IBM DB2 10.1
for Linux, UNIX, and Windows

データベースのモニタリング ガイドおよびリファレンス

2013 年 1 月更新版

SA88-4663-01
(英文原典：SC27-3887-01)

IBM DB2 10.1
for Linux, UNIX, and Windows

データベースのモニタリング ガイドおよびリファレンス

2013 年 1 月更新版

SA88-4663-01
(英文原典：SC27-3887-01)

ご注意

本書および本書で紹介する製品をご使用になる前に、 1891 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

IBM 資料は、オンラインでご注文いただくことも、ご自分の国または地域の IBM 担当員を通してお求めいただくこともできます。

- オンラインで資料を注文するには、IBM Publications Center (http://www.ibm.com/shop/publications/order) をご利用ください。
- ご自分の国または地域の IBM 担当員を見つけるには、IBM Directory of Worldwide Contacts (http://www.ibm.com/planetwide/) をお調べください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC27-3887-01
IBM DB2 10.1
for Linux, UNIX, and Windows
Database Monitoring Guide and
Reference
Updated January, 2013

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.12

目次

本書について

「システム・モニター *ガイドおよびリファレンス*」では、データベースおよびデータベース・マネージャーに関するさまざまな種類の情報を収集する方法について説明しています。

本書では、収集した情報を使用してデータベース・アクティビティーを理解したり、パフォーマンスを向上させたり、問題の原因を判別したりする方法についても説明しています。

第 1 部 データベース・モニターのインターフェース

データベース内の操作をモニターするには、2 つの方法があります。特定の時点におけるデータベースの状態のさまざまな側面を示す情報を参照できます。または、イベント・モニターをセットアップして、特定のタイプのデータベース・イベントが発生したときに履歴情報をキャプチャーすることができます。

モニター表関数を使用すると、データベース操作をリアルタイムでモニターできます。例えば、モニター表関数を使用して、表スペースの合計使用量を調べることができます。これらの表関数により、 SQL を使用してデータベース操作のほぼすべての側面についてレポートするモニター・エレメント およびメトリックを調べることができます。モニター表関数は、バージョン 9.7 で導入された軽量で高速な新しいモニター・インフラストラクチャーを使用しています。表関数に加えて、スナップショット・モニター・ルーチンも用意されています。 DB2® のスナップショット・モニター機能は、バージョン 9.7 より前に存在したモニター・インフラストラクチャーを使用しています。一般的に、今後本製品でスナップショット・モニター機能が拡張されることはありません。参照するデータを取得する際には、可能であればモニター表関数を使用してください。

イベント・モニターは、ある期間において特定のタイプのイベントが発生したときに、データベース操作に関する情報をキャプチャーします。例えば、ロックおよびデッドロックがシステムで発生したときに、それらの情報をキャプチャーするイベント・モニターを作成できます。または、指定したしきい値 (アプリケーションまたはワークロードで使用される合計プロセッサー時間など) が超過された時刻を記録するイベント・モニターを作成することもできます。イベント・モニターはさまざまな形式で出力を生成します。すべてのイベント・モニターは、イベント・データを通常の表に書き込めます。追加の出力オプションを持つイベント・モニターもあります。

IBM® InfoSphere® Optim™ Performance Manager は、データベースの典型的なパフォーマンス上の問題を特定して分析するために使用できる Web インターフェースを提供します。データベースの正常性の要約を表示して調べることもできます。詳細については、Optim Performance Manager を使用したモニター (http://publib.boulder.ibm.com/infocenter/idm/docv3/topic/com.ibm.datatools.perfmgmt.monitor.doc/p_monitor.html) を参照してください。

第 1 章 データベース・モニター

データベース・モニター という用語は、データベースの運用状況の調査に関連する作業を指します。

データベースのモニターは、データベース管理システムのパフォーマンスと正常性を保守するためにきわめて重要なアクティビティーです。モニターを容易にするために、DB2 はデータベース・マネージャー、データベース、および他の接続されているアプリケーションから情報を収集します。この情報を使用して、以下のタイプの作業などを行うことができます。

- データベース使用パターンに基づいたハードウェア要件の予測
- 個々のアプリケーションまたは SQL 照会のパフォーマンスの分析
- 索引および表の使用量の追跡
- システム・パフォーマンス低下の原因の正確な指摘
- 最適化アクティビティー (データベース・マネージャー構成パラメーターの変更、索引の追加、SQL 照会の変更など) の影響の評価

第 2 章 モニターする表関数

DB2 バージョン 9.7 からは、従来のシステム・モニターに代わる軽量の代替手段により、モニター・データにアクセスできるようになりました。モニター表関数を使用して、システム、アクティビティー、またはデータ・オブジェクトのデータを収集および表示します。

モニター・エレメントのデータは、メモリー内に継続的に累積され、照会に使用することができます。単一オブジェクト (例えば、サービス・クラス A または表 TABLE1) についてのデータまたはすべてのオブジェクトについてのデータを受け取ることを選択できます。

データベース・パーティション環境内でこれらの表関数を使用すると、単一のパーティションについてのデータまたはすべてのパーティションについてのデータを受け取ることを選択できます。すべてのパーティションのデータを受け取ることを選択した場合、表関数は各パーティションについて 1 行を返します。 SQL を使用すると、すべてのパーティションの値を合計して、すべてのパーティションのモニター・エレメントの値を取得することができます。

表関数を使用したシステム情報のモニター

システム・モニター・パースペクティブは、アプリケーション要求を処理するためにデータ・サーバーによって費やされた作業量および労力全体を対象としています。このパースペクティブから、データ・サーバーの実行内容を全体として、およびアプリケーション要求の特定のサブセットごとに判別することができます。

要求モニター・エレメントと呼ばれる、このパースペクティブのモニター・エレメントは、要求の処理と関連するデータ・サーバー操作の全範囲を対象とします。

要求モニター・エレメントはメモリー内に継続的に累積されて集約されるため、それらは即時に照会に使用可能です。要求モニター・エレメントは、ワークロード管理 (WLM) オブジェクト階層のさまざまなレベル (作業単位別、ワークロード別、サービス・クラス別) の要求全体について集約されます。それらは接続別でも集約されます。

以下の表関数を使用して、現在のシステム・モニター情報にアクセスします。

- MON_GET_SERVICE_SUBCLASS および MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_WORKLOAD および MON_GET_WORKLOAD_DETAILS
- MON_GET_CONNECTION および MON_GET_CONNECTION_DETAILS
- MON_GET_UNIT_OF_WORK および MON_GET_UNIT_OF_WORK_DETAILS

この一連の表関数により、特定の集約レベルの要求モニター・エレメントをドリルダウンしたり、それに焦点を当てたりすることができます。表関数は対で提供されています。1 つは一般に使用されるデータへのリレーショナル・アクセス用、もう 1 つは使用可能なモニター・エレメントの完全セットへの XML アクセス用です。

システム・モニター情報は、これらの表関数により、デフォルトでは新規データベースを対象に収集されます。デフォルト設定は、以下のいずれかまたは両方の設定を使用して変更できます。

- データベース構成パラメーター **mon_req_metrics** は、すべてのサービス・クラスにおける最小レベルの収集を指定します。
- CREATE/ALTER SERVICE CLASS ステートメントの COLLECT REQUEST METRICS 節は、サービス・スーパークラスの収集のレベルを指定します。この設定を使用して、すべてのサービス・クラスに設定されている最小レベルの収集をオーバーライドして、特定のサービス・クラスの収集レベルを上げることができます。

各設定に指定可能な値は、以下のとおりです。

None 要求モニター・エレメントは収集されません。

BASE すべての要求モニター・エレメントが収集されます。

例えば、サービス・クラスのサブセットのみのシステム・モニター情報を収集するには、以下を実行します。

1. データベース構成パラメーター **mon_req_metrics** を NONE に設定します。
2. 対象とする各サービス・クラスについて、CREATE/ALTER SERVICE CLASS ステートメントの COLLECT REQUEST METRICS 節を BASE に設定します。

表関数を使用したアクティビティーのモニター

アクティビティー・モニター・パースペクティブは、アクティビティーの実行に関係した、データ・サーバー処理のサブセットに焦点を当てています。 SQL ステートメントのコンテキストでは、「アクティビティー」という語は、SQL ステートメントのセクションの実行を表します。

アクティビティー・モニター・エレメントと呼ばれる、このパースペクティブのモニター・エレメントは、要求モニター・エレメントのサブセットです。アクティビティー・モニター・エレメントは、ステートメント・セクションの実行で行われた作業の各局面を測定します。アクティビティー・モニターには、アクティビティーの SQL ステートメント・テキストなどの他の情報も含まれます。

進行中のアクティビティーの場合、アクティビティー・メトリックはメモリーに累積されます。 SQL ステートメントのアクティビティーの場合、アクティビティー・メトリックはパッケージ・キャッシュにも累積します。パッケージ・キャッシュには、各 SQL ステートメント・セクションのすべての実行に対するアクティビティー・メトリックが集約されます。

以下の表関数を使用して、アクティビティーの現行データにアクセスします。

MON_GET_ACTIVITY_DETAILS
表関数が呼び出されるときに、進行中の個々のアクティビティーに関するデータを返します。データはリレーショナル形式で返されますが、詳細メトリックは結果表にある DETAILS 列に XML 文書形式で返されます。

MON_GET_PKG_CACHE_STMT
データベース・パッケージ・キャッシュの静的および動的の両方の SQL ステートメントのポイント・イン・タイム・ビューを戻します。データはリレーショナル形式で返されます。

MON_GET_PKG_CACHE_STMT_DETAILS
1 つ以上のパッケージ・キャッシュ項目の詳細メトリックを戻します。データはリレーショナル形式で返されますが、詳細メトリックは結果表にある DETAILS 列に XML 文書形式で返されます。

アクティビティー・モニター情報は、デフォルトでは新規データベースを対象に収集されます。デフォルト設定は、以下のいずれかまたは両方の設定を使用して変更できます。

- **mon_act_metrics** データベース構成パラメーターは、すべてのワークロードにおける最小レベルの収集を指定します。
- CREATE/ALTER WORKLOAD ステートメントの COLLECT ACTIVITY METRICS 節は、すべてのワークロードに設定される最小レベルの収集をオーバーライドして、特定のワークロードの収集レベルを指定します。

各設定に指定可能な値は、以下のとおりです。

None アクティビティー・モニター・エレメントは収集されません。

BASE すべてのアクティビティー・モニター・エレメントが収集されます。

例えば、選択したワークロードのみのアクティビティー・モニター・エレメントを収集するには、以下を実行します。

1. **mon_act_metrics** データベース構成パラメーターを NONE に設定します。
2. CREATE/ALTER WORKLOAD ステートメントの COLLECT ACTIVITY METRICS 節を BASE に設定します。デフォルトでは、その他のワークロードの値は NONE です。

表関数を使用したデータ・オブジェクトのモニター

データ・オブジェクト・モニター・パースペクティブは、表、索引、バッファー・プール、表スペース、およびコンテナーなどのデータ・オブジェクトに対して実行された操作に関する情報を提供します。

各オブジェクト・タイプに対してさまざまなモニター・エレメントのセットが使用できます。データ・オブジェクトのモニター・エレメントは、要求が対象オブジェクトの処理に関与するたびに増分されます。例えば、特定の表からの行の読み取りに関与する要求を処理する場合、その表の行読み取りのメトリックが増分されます。

以下の表関数を使用して、データ・オブジェクトの現行の詳細にアクセスします。

- MON_GET_BUFFERPOOL
- MON_GET_TABLESPACE
- MON_GET_CONTAINER
- MON_GET_TABLE
- MON_GET_INDEX

これらの表関数は、リレーショナル形式でデータを返します。

データ・オブジェクトの履歴データにはアクセスできません。

データ・オブジェクト・モニター・エレメントは、デフォルトでは新規データベースを対象に収集されます。 **mon_obj_metrics** データベース構成パラメーターを使用して、表関数により収集されるデータの量を削減することができます。

この構成パラメーターに指定可能な値は、以下のとおりです。

None データ・オブジェクト・モニター・エレメントは収集されません。

BASE 一部のデータ・オブジェクト・モニター・エレメントが収集されます。

Extended
すべてのデータ・オブジェクト・モニター・エレメントが収集されます。

以下の表関数により報告されるデータ・オブジェクト・モニター・エレメントの収集を停止するには、**mon_obj_metrics** 構成パラメーターを NONE に設定します。

- MON_GET_BUFFERPOOL
- MON_GET_TABLESPACE
- MON_GET_CONTAINER

オブジェクトの使用

SQL ステートメントを実行すると、表や索引など様々なデータベース・オブジェクトが使用されます。ステートメントがアクセスするデータベース・オブジェクトや、ステートメントがどのように影響を及ぼすのかを把握すると、モニターやパフォーマンス調整のターゲットを識別するのに役立ちます。

以下の表には、データベース・オブジェクトとステートメントの関係を調べるために使用できるエンティティーが示されています。

表 1. オブジェクトの使用法を識別する手段

メカニズム	定義	使用法
使用リスト	使用リストは、特定の表または索引を参照する各 DML ステートメント・セクションを記録し、セクションの実行時にセクションに関する統計をキャプチャーするデータベース・オブジェクトです。	表または索引に影響を及ぼしたステートメントを識別します。データベース・オブジェクトのモニター時にメトリックで異常な値に気付く場合、使用リストを用いて、そのメトリックに関係する特定のステートメントを判別します。また、オブジェクトに影響を及ぼした各ステートメントの統計も表示できます。

表 1. オブジェクトの使用法を識別する手段 (続き)

メカニズム	定義	使用法
実行時統計が含まれるセクション Explain	セクション Explain とは、SQL ステートメントでオプティマイザーが選択するアクセス・プランについての一連の情報のことです。 Explain の一部としてセクション実行時統計をキャプチャーできます。セクション実行時統計とは、セクションを実行したときに収集されるランタイムの統計のことです。	ステートメントが影響を及ぼす表または索引を識別します。表または索引ごとに統計を表示し、それらの統計を使用して、ステートメントが各オブジェクトに及ぼす影響や、調整が必要な場所を判別できます。

使用リスト、または実行時統計を持つセクション Explain の情報を、パフォーマンス調整用のベースライン・データとして使用できます。ステートメントまたはデータベース構成パラメーターにおける調整を行う前に、オブジェクトの使用に関する情報を収集してください。調整後もう一度情報を収集し、調整によってパフォーマンスが改善されたことを検証します。

表に影響を及ぼすステートメントの識別

使用リストを使用して、DML ステートメント・セクションの実行時に特定の表に影響を及ぼすステートメント・セクションを識別します。各ステートメントの統計を表示し、それらの統計を使用してさらにモニタリングや調整が必要となる箇所を判別できます。

始める前に

次のタスクを実行します。

- オブジェクトの使用統計を表示する対象となる表を識別します。MON_GET_TABLE 表関数を使用すると、1 つ以上の表のモニター・メトリックを表示できます。
- 必要なステートメントを実行するために、各ステートメントの許可 ID によって保持されている特権として DBADM 権限または SQLADM 権限が含まれていることを確認します。
- MON_GET_TABLE_USAGE_LIST 表関数または MON_GET_USAGE_LIST_STATUS 表関数に関する EXECUTE 特権があることを確かめます。

このタスクについて

MON_GET_TABLE 表関数の出力を表示すると、モニター・エレメントに異常値がある場合があります。使用リストを使用して、いずれかの DML ステートメントがその値に関係しているかどうかを判別できます。

使用リストには、特定の時間フレームにおいて表に影響を及ぼしたそれぞれのステートメントに関する、ロックやバッファー・プールの使用などの要因についての統計が含まれています。特定のステートメントが表に悪影響を与えていると判断する

場合、こうした統計を使用して、さらにモニタリングが必要な箇所や、ステートメントの調整方法を判別します。

手順

表に影響を及ぼすステートメントを識別するには、以下のようにします。

1. 以下のコマンドを発行して、**mon_obj_metrics** 構成パラメーターを EXTENDED に設定します。

   ```
   DB2 UPDATE DATABASE CONFIGURATION USING MON_OBJ_METRICS EXTENDED
   ```

   この構成パラメーターを EXTENDED に設定すると、使用リスト内の項目ごとに統計が収集されるようになります。

2. CREATE USAGE LIST ステートメントを使用して、表の使用リストを作成します。 例えば、SALES.INVENTORY 表の INVENTORYUL 使用リストを作成する場合、次のコマンドを発行します。

   ```
   CREATE USAGE LIST INVENTORYUL FOR TABLE SALES.INVENTORY
   ```

3. SET USAGE LIST STATE ステートメントを使用して、オブジェクト使用統計の収集をアクティブにします。 例えば、INVENTORYUL 使用リストの収集をアクティブにする場合、次のコマンドを発行します。

   ```
   SET USAGE LIST INVENTORYUL STATE = ACTIVE
   ```

4. オブジェクト統計の収集中に、MON_GET_USAGE_LIST_STATUS 表関数を使用して、使用リストの状態がアクティブで、使用リストに割り振られているメモリー量が十分であることを確認します。 例えば、INVENTORYUL 使用リストの状況を確認する場合、次のコマンドを発行します。

   ```
   SELECT MEMBER,
          STATE,
          LIST_SIZE,
          USED_ENTRIES,
          WRAPPED
   FROM TABLE(MON_GET_USAGE_LIST_STATUS('SALES', 'INVENTORYUL', -2))
   ```

5. オブジェクト使用統計を収集する期間が経過したら、SET USAGE LIST STATE ステートメントを使用して、使用リストのデータ収集を非アクティブにします。 例えば、INVENTORYUL 使用リストの収集を非アクティブにする場合、次のコマンドを発行します。

   ```
   SET USAGE LIST SALES.INVENTORYUL STATE = INACTIVE
   ```

6. MON_GET_TABLE_USAGE_LIST 関数を使用して収集された情報を表示します。 統計を収集した期間中に表に影響を及ぼしたステートメントの一部またはそのすべての統計を表示できます。 例えば、表から読み取った行の数が多い上位 10 個のステートメントだけを表示する場合、以下のコマンドを発行します。

   ```
   SELECT MEMBER,
          EXECUTABLE_ID,
          NUM_REFERENCES,
          NUM_REF_WITH_METRICS,
          ROWS_READ,
          ROWS_INSERTED,
          ROWS_UPDATED,
          ROWS_DELETED
   FROM TABLE(MON_GET_TABLE_USAGE_LIST('SALES', 'INVENTORYUL', -2))
   ORDER BY ROWS_READ DESC
   FETCH FIRST 10 ROWS ONLY
   ```

7. 表に影響を及ぼしたステートメントのテキストを表示する場合、MON_GET_TABLE_USAGE_LIST 出力の **executable_id** エレメントの値を MON_GET_PKG_CACHE_STMT 表関数の入力として使用します。 例えば、次のコマンドを発行して、特定のステートメントのテキストを表示します。

```
SELECT STMT_TEXT
FROM TABLE
(MON_GET_PKG_CACHE_STMT(NULL,
x'0100000000000007C000000000000000000000000200200811261717207289 97',
NULL, -2))
```

8. ステートメントのリストと、それらのステートメント用に提供されている統計を使用して、追加のモニタリングや調整が必要な場合にはその箇所を判別します。例えば、**pool_writes** モニター・エレメントの値が **direct_writes** モニター・エレメント値と比較して低いステートメントには、注意が必要なバッファー・プールの問題がある可能性があります。

次のタスク

使用リストの情報が不要な場合には、SET USAGE LIST STATE ステートメントを使用して使用リストに関連するメモリーを解放します。例えば、INVENTORYUL 使用リストに関連するメモリーを解放するには、次のコマンドを発行します。

```
SET USAGE LIST SALES.INVENTORYUL STATE = RELEASED
```

ステートメントがデータベース・オブジェクトに及ぼす影響の調査

セクション実行時統計情報が含まれるセクション Explain を使用して、ステートメントがデータベース・オブジェクトに及ぼす影響を調査することができます。それぞれの表や索引にステートメント・セクションがどのように影響を及ぼしたかについての統計を使用して、追加のモニタリングや調整が必要かどうかを判断できます。

始める前に

次のタスクを実行します。

- オブジェクトの使用統計を表示する対象となるステートメントを識別します。
- Explain 表を DB2 バージョン 10.1 にマイグレーション済みであることを確認します。
- 自動統計プロファイル生成が無効になっていることを確認します。
- EXPLAIN_FROM_ACTIVITY プロシージャーの呼び出しに必要な特権を持っていることを確かめます。

このタスクについて

オブジェクトの使用統計を表示する対象となるステートメントを識別してから、セクションの実行時統計情報が含まれるセクション Explain を取得できます。セクションの実行時統計情報には、ステートメントが実行時に使用したそれぞれの表または索引にどのように影響を及ぼしたかが示されています。

実行時統計情報には、それぞれの表または索引についてのロックおよびバッファー・プールの使用などの要因についての実行時統計が含まれています。それらの統

計をベースライン・データと比較して、さらにモニタリングや調整が必要となる箇所があるかどうかを判断できます。

手順

データベース・オブジェクトがステートメントによってどのように影響を受けているかを判別するには、以下のようにします。

1. 次のコマンドを発行して、データベース・レベルにおけるセクション実行時統計の収集を有効にします。

   ```
   DB2 UPDATE DATABASE CONFIGURATION USING SECTION_ACTUALS BASE
   ```

2. ステートメントを発行するアプリケーションによって処理依頼されるアクティビティーについてのセクション実行時統計情報を収集するワークロードを作成します。 例えば、TEST アプリケーションによって処理依頼されるアクティビティーについての ACTWORKLOAD ワークロードを作成し、それらのアクティビティーに関して収集を有効にするには、次のコマンドを発行します。

   ```
   CREATE WORKLOAD ACTWORKLOAD APPLNAME ('TEST')
   COLLECT ACTIVITY DATA ON ALL WITH DETAILS,SECTION INCLUDE ACTUALS BASE
   ```

   セクション実行時統計の収集の有効化は、以下の方法でも実行できます。

   - CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメント
   - CREATE WORK ACTION SET または ALTER WORK ACTION SET ステートメント
   - WLM_SET_CONN_ENV プロシージャー
   - **section_actuals** 構成パラメーター

3. CREATE EVENT MONITOR ステートメントを使用して、アクティビティー・イベント・モニターを作成します。 例えば、ACTEVMON アクティビティー・イベント・モニターを作成するには、次のコマンドを発行します。

   ```
   CREATE EVENT MONITOR ACTEVMON
          FOR ACTIVITIES
          WRITE TO TABLE
          CONTROL (TABLE CONTROL_ACTEVMON ),
          ACTIVITY (TABLE ACTIVITY_ACTEVMON ),
          ACTIVITYSTMT (TABLE ACTIVITYSTMT_ACTEVMON ),
          ACTIVITYVALS (TABLE ACTIVITYVALS_ACTEVMON ),
          ACTIVITYMETRICS (TABLE ACTIVITYMETRICS_ACTEVMON )
   ```

4. SET EVENT MONITOR STATE ステートメントを使用して作成したアクティビティー・イベント・モニターをアクティブにします。 例えば、ACTEVMON アクティビティー・イベント・モニターをアクティブにするには、次のコマンドを発行します。

   ```
   SET EVENT MONITOR ACTEVMON STATE 1
   ```

5. オブジェクト統計を表示する対象となるステートメントを発行するアプリケーションを実行します。
6. 次のコマンドを使用してアクティビティー・イベント・モニター表を照会して、ステートメント・セクションの ID 情報を検索します。

```
SELECT APPL_ID,
       UOW_ID,
       ACTIVITY_ID,
       STMT_TEXT
FROM ACTIVITYSTMT_ACTEVMON
```

7. アクティビティー ID 情報を EXPLAIN_FROM_ACTIVITY プロシージャーの入力として使用して、実行時統計を含むセクション Explain を取得します。 例えば、アプリケーション ID *N2.DB2INST1.0B5A12222841、作業単位 ID 16、アクティビティー ID 4 のセクションのセクション Explain を取得するには、次のコマンドを発行します。

   ```
   CALL EXPLAIN_FROM_ACTIVITY( '*N2.DB2INST1.0B5A12222841', 16, 4, 'ACTEVMON',
    'MYSCHEMA', ?, ?, ?, ?, ? )
   ```

   以下の出力例のような出力が戻されます。

   ```
   Value of output parameters
   --------------------------
   Parameter Name : EXPLAIN_SCHEMA
   Parameter Value : MYSCHEMA

   Parameter Name : EXPLAIN_REQUESTER
   Parameter Value : GSDBUSER3

   Parameter Name : EXPLAIN_TIME
   Parameter Value : 2010-11-23-10.51.09.631945

   Parameter Name : SOURCE_NAME
   Parameter Value : SQLC2J21

   Parameter Name : SOURCE_SCHEMA
   Parameter Value : NULLID

   Parameter Name : SOURCE_VERSION
   Parameter Value :

   Return Status = 0
   ```

8. **db2exfmt** コマンドを使用して、Explain データをフォーマット設定します。 EXPLAIN_FROM_ACTIVITY プロシージャーからの出力の **explain_requester**、**explain_time**、**source_name**、**source_schema**、および **source_version** の各パラメーターの値をこのコマンドの入力として使用します。
9. Explain 出力を表示して、セクションの実行時に使用されたデータベース・オブジェクトに対して、セクションがどのような影響を及ぼしたかを判別します。 出力にある統計によって、さらにモニタリングや調整が必要となる場合があります。 例えば、セクションが使用した表についての **lock_wait** モニター・エレメントの値が高い場合には、ロック管理が必要となることがあります。
10. ステートメントを調整する場合には、ステップ 5 (12 ページ) から 9 までを繰り返して、パフォーマンスが向上したことを確かめます。

次のタスク

SET EVENT MONITOR STATE ステートメントを使用して、アクティビティー・イベント・モニターを非アクティブにします。例えば、ACTEVMON アクティビティー・イベント・モニターを非アクティブにするには、次のコマンドを発行します。

```
SET EVENT MONITOR ACTEVMON STATE 0
```

表関数を使用したロックのモニター

表関数を使用して、ロックに関する情報を取得できます。要求、アクティビティー、データ・オブジェクトのモニター・エレメントとは異なり、ロックに関する情報はいつでもデータベース・マネージャーから入手できます。この情報の収集を有効にする必要はありません。

システム内のロックに関する現行情報にアクセスするには、以下のモニター表関数を使用します。

- MON_GET_LOCKS
- MON_GET_APPL_LOCKWAIT

どちらの表関数も、リレーショナル形式でデータを返します。

表関数を使用したシステム・メモリーのモニター

表関数を使用したシステム・メモリーの使用量に関する情報を取得することができます。

オペレーティング・システムからのメモリー割り振りである、メモリー・セットのレベルのメモリー使用量を調べることができます。指定されたメモリー・セット内の特定のメモリー・プールによるメモリーの使用量を調べることもできます。メモリーの使用量に関する現行情報にアクセスするには、以下のモニター関数を使用します。

- MON_GET_MEMORY_SET
- MON_GET_MEMORY_POOL

表関数を使用したルーチンのモニター

ルーチンについての情報を取得するために、表関数を使用することができます。

表関数を使用すると、ルーチンをモニターしたり、以下の情報を提供したりできます。

- 対象ルーチンの総コストをレポートする集約メトリック。メトリックは、ルーチンのすべての呼び出しにおいて集約され、対象ルーチンによって実行されるすべての子ステートメントと要求に関するメトリックが含まれます。
- ルーチンによって実行されたステートメントのリスト。ドリルダウンや問題判別に役立ちます。
- ステートメントによって呼び出される可能性のあるルーチンのリスト。ルーチン関連の詳細をさらにドリルダウンするのに役立ちます。

以下のモニター関数を使用して、ルーチンに関する情報を入手します。

- MON_GET_ROUTINE
- MON_GET_ROUTINE_DETAILS
- MON_GET_ROUTINE_EXEC_LIST
- MON_GET_SECTION_ROUTINE

例: CPU 消費量が最も多いルーチンの識別

ルーチン・モニターを使用すると、最もコストの高いルーチンを識別できます。

シナリオ

この例では、データベース管理者 (DBA) が最も多くの CPU 合計を消費するデータベース・ルーチンを識別しようとしています。 DBA は次の照会を発行し、データベースが活動化されて以降に実行されたすべてのルーチンを表示します。

```
SELECT ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME,
   SPECIFICNAME, SUM(TOTAL_CPU_TIME) AS TOTAL_CPU
FROM TABLE(MON_GET_ROUTINE(NULL,NULL,NULL,NULL,-2)) AS T
GROUP BY ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, SPECIFICNAME
ORDER BY TOTAL_CPU DESC
```

結果は、ルーチンの CPU 消費量合計の順になります。

```
ROUTINESCHEMA     ROUTINEMODULENAME  ROUTINENAME                SPECIFICNAME        TOTAL_CPU
----------------- ------------------ -------------------------- ------------------- --------------------
SYSIBMINTERNAL    -                  COMPILED_ANON_BLOCK_INVOKE  SQL120801135416210              8942414
DRICARD           -                  PROC1                      PROC1                             23444
SYSIBMSUBROUTINE  -                  PROC1_66613_101877843      -                                  4213
DRICARD           -                  MYPROC                     SQL120801135351900                 1838
DRICARD           -                  TRIG1                      SQL120801135519200                  467

  5 record(s) selected.
```

これで DBA は、最も多くの CPU 合計を消費するルーチンのチューニング作業に集中できます。

例: ルーチンによって実行されたステートメントに費やされた時間のリスト作成

ルーチン・モニターを使用すると、ルーチンによって実行された各ステートメントにかかった時間をリスト表示できます。

シナリオ

この例では、データベース管理者 (DBA) は TEST.PROC1 という重要なストアード・プロシージャーのパフォーマンスを調査しています。 MON_GET_ROUTINE 表関数によって返される TOTAL_ROUTINE_COORD_EXEC_TIME モニター・エレメントは、このストアード・プロシージャーの実行経過時間が長いことを示しています。 DBA は、MON_RTN_EXECLIST データベース構成パラメーターを使用して、ルーチンごとにステートメント情報を追跡するようにデータベースを構成してあります。 DBA は以下の照会を発行し、TEST.PROC1 によって実行されたステートメントのリストを作成します。

```
SELECT B.EXECUTABLE_ID,
   100*B.COORD_STMT_EXEC_TIME / A.TOTAL_ROUTINE_COORD_EXEC_TIME
AS PERCENT_EXEC_TIME,(SELECT SUBSTR(C.STMT_TEXT,1,120)
FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL,B.EXECUTABLE_ID,NULL,-2)) AS C) AS STMT_TEXT
FROM TABLE(MON_GET_ROUTINE('P','TEST',NULL,'PROC1', -2)) AS A,
   TABLE(MON_GET_ROUTINE_EXEC_LIST('P','TEST',NULL,'PROC1', -1)) AS B
WHERE A.TOTAL_ROUTINE_COORD_EXEC_TIME<>0
ORDER BY PERCENT_EXEC_TIME DESC
```

結果は、ルーチンの実行経過時間のパーセンテージ順になります。

```
EXECUTABLE_ID                                                                    PERCENT_EXEC_TIME    STMT_TEXT
-------------------------------------------------------------------------------- -------------------- ----------------------------------
x'0100000000000000C0000000000000000000000020020120801145618138490'                    0 SELECT TABNAME FROM SYSCAT.TABLES
                                                                                           WHERE TABNAME='MY_TABLE' AND  TABSCHEMA='MYSCHEMA'
x'0100000000000000B0000000000000000000000020020120801145618127528'                    0 SELECT TABNAME FROM SYSCAT.TABLES
                                                                                           WHERE TABNAME='MY_TABLE_2' AND  TABSCHEMA='MYSCHEMA'
x'0100000000000000070000000000000000000000020020120801145618004432'                   0 SELECT COLNAME, TYPENAME FROM  SYSCAT.COLUMNS
                                                                                           WHERE TABNAME='MY_TABLE' AND TABSCHEMA='MYSCHEMA'

  3 record(s) selected.
```

示された、ルーチンで実行時間の最も長いステートメントに関して、DBA はさらに調査を行い、一部のステートメントが長期間実行されている原因を判別できます。DBA は対象ルーチンにおけるそのステートメントのすべての実行に関して集約されたエンベロープ・メトリックを調べるか、ステートメントの **executable_id** を MON_GET_PKG_CACHE_STMT 表関数と一緒に使用してそのステートメントのすべての実行に関する詳細メトリックを検索できます。

例: ルーチンによる CPU 高消費量の調査

ルーチン・モニターを使用すると、ルーチンが消費する CPU リソース量が予想よりも多い理由を調査できます。

シナリオ

この例では、データベース管理者 (DBA) は、TEST.PROC1 という重要なストアード・プロシージャーのパフォーマンスを調査しています。 MON_GET_ROUTINE 表関数によって返される TOTAL_ROUTINE_CPU_TIME モニター・エレメントは、このストアード・プロシージャーが予想されていた以上の CPU リソースを使用していることを示しています。 DBA は、MON_RTN_EXECLIST データベース構成パラメーターを使用して、ルーチンごとにステートメント情報を追跡するようにデータベースを構成してあります。 DBA は以下の照会を発行し、すべてのメンバーにおいてこのルーチンが実行する各ステートメントの CPU 使用量の合計を求めます。

```
WITH TOTAL_STMT_CPU (EXEC_ID, TOTAL_CPU, NUM_ROUTINES, MIN_MEMBER) AS
  (SELECT
     EXECUTABLE_ID,
     SUM(TOTAL_CPU_TIME),
     MAX(NUM_ROUTINES),
     MIN(MEMBER)
   FROM
     TABLE(MON_GET_ROUTINE_EXEC_LIST('P', 'TEST', NULL, 'PROC1', -2)) AS T
   GROUP BY
     EXECUTABLE_ID
  ),
  TOTAL_RTN_CPU (TOTAL_RTN_CPU) AS
  (SELECT
     SUM(TOTAL_CPU_TIME)
   FROM
     TABLE(MON_GET_ROUTINE('P', 'TEST', NULL, 'PROC1', -2)) AS R
  )
SELECT
  B.EXEC_ID,
  100*B.TOTAL_CPU / A.TOTAL_RTN_CPU AS PERCENT_CPU,
  B.NUM_ROUTINES,
  C.STMT_TEXT
FROM
  TOTAL_RTN_CPU AS A,
  TOTAL_STMT_CPU AS B,
  TABLE(MON_GET_PKG_CACHE_STMT(NULL, NULL, NULL, -2)) AS C
WHERE
  B.EXEC_ID = C.EXECUTABLE_ID AND
```

```
  B.MIN_MEMBER = C.MEMBER AND
  A.TOTAL_RTN_CPU<>0
ORDER BY
  TOTAL_CPU DESC
```

結果には、ストアード・プロシージャーによって実行されたステートメントが、CPU 合計に対する、ルーチンで使用された CPU 消費量パーセンテージによってランク付けされてリスト表示されます。

```
EXEC_ID                                                                  PERCENT_CPU  NUM_ROUTINES   STMT_TEXT
------------------------------------------------------------------------ ------------ -------------- ----------------------------------------
x'0100000000000000560100000000000002000000010020120801142628005514'           10              0 WITH GET_UPDATE_LIST (COL1, COL2STATS) AS AF
x'0100000000000000560100000000000002000000010020120801142628005514'            1              0 insert into T1 values(3,'d','d','d')
x'0100000000000000560100000000000001000000010020120801142628005514'            0              1 call SYSIBMSUBROUTINE.P1_66613_1157394573()

  3 record(s) selected.
```

MON_GET_ROUTINE_EXEC_LIST 表関数によってレポートされる各ステートメントの CPU 消費量には、子ステートメントが消費した CPU 量は含まれません。MON_GET_ROUTINE_EXEC_LIST が生成するレポートには、対象ルーチンが直接呼び出したステートメントによって消費された CPU 量のパーセンテージだけが示されます。 DBA は次のようにして調査を続行できます。

- 直接の子ステートメントが CPU の大部分を使用している場合、MON_GET_ROUTINE_EXEC_LIST レポートには問題を引き起こしている可能性のあるステートメントが示され、DBA はそのステートメントを調査できます。例えば、MON_GET_PKG_CACHE_STMT 表関数を使用して、CPU 消費量が多いステートメントをドリルダウンし、そのステートメントのすべての実行に関するメトリックの全セットをリストします。

  ```
  SELECT * FROM TABLE(MON_GET_PKG_CACHE_STMT
  (NULL, '<high_cpu_consuming_exec_id>', NULL, -2)) AS T
  ```

- 出力に、ルーチンによって直接実行されているステートメントに大量の CPU を使用しているものがないことが示される場合、DBA は実行されたステートメントごとにリストされている **num_routines** モニター・エレメントを調べることができます。
  - 実行された各ステートメントの **num_routines** がゼロの場合、CPU 消費量が多いのはルーチン自体での処理に原因があります。
  - **num_routines** がゼロではない場合、DBA は MON_GET_SECTION_ROUTINE 表関数を使用して、そのステートメントが呼び出したルーチンを判別し、そうしたルーチンが TEST.PROC1 全体 の CPU 消費の原因になっているかどうかを調べることができます。例えば、

    ```
    SELECT
       ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, SPECIFICNAME
       FROM TABLE(MON_GET_SECTION_ROUTINE('<exec_id>')) AS T
    ```

    返される結果は、次のとおりです。

    ```
    ROUTINESCHEMA     ROUTINEMODULENAME  ROUTINENAME                SPECIFICNAME
    ----------------- ------------------ -------------------------- -------------
    DRICARD           -                  PROC1                      PROC1
    SYSIBMSUBROUTINE  -                  PROC1_66613_101877843      -

      2 record(s) selected
    ```

    DBA は CPU の高消費量についてこうしたルーチンを調べることができますが、こうしたルーチンは TEST.PROC1 以外の別のコンテキストから呼び出された可能性もあることに留意してください。つまり、こうしたルーチンに関し

て MON_GET_SECTION_ROUTINE から返されるメトリックは、TEST.PROC1 に関するメトリックよりも大きい可能性があります。

例: 無名ブロックの集約ルーチン・メトリックのリスト作成

ルーチン・モニターを使用すると、無名ブロックで実行されるステートメントの集約メトリックのリストを作成できます。

シナリオ

この例では、データベース管理者 (DBA) がステートメント・テキスト 'BEGIN ... END' の無名ブロックによって実行されるすべてのステートメントに関する集約メトリックを表示しようとしています。 DBA は以下の照会を発行し、パッケージ・キャッシュ内で無名ブロックの **executable_id** を検索します。

```
SELECT EXECUTABLE_ID
   FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL, NULL,NULL,-1)) AS T
   WHERE STMT_TEXT = 'BEGIN BLAH BLAH END'
```

返される結果は、次のとおりです。

```
EXECUTABLE_ID
--------------------------------------------------------------------
x'0100000000000007A000000000000000000000000200201208011538417899 93'

  1 record(s) selected.
```

DBA は、返された **executable_id** を使用して以下の照会を発行し、対象の無名ブロックの集約メトリックを検索します。

```
SELECT * FROM TABLE(MON_GET_ROUTINE('A', NULL,NULL,NULL,-2)) AS T
   WHERE DYN_COMPOUND_EXEC_ID =  x'0100000000000007A000000000000000000000000200201208011538417899 93'
```

返される結果は、次のとおりです。

```
ROUTINETYPE ROUTINESCHEMA   ROUTINEMODULENAME  ROUTINENAME                SPECIFICNAME         DYN_COMPOUND_EXEC_ID
----------- --------------- ------------------ -------------------------- -------------------- ------------------------
C           SYSIBMINTERNAL  -                  COMPILED_ANON_BLOCK_INVOKE  SQL120801153841490   x'01000000000000007A00000
                                                                                                  000000000000000000
                                                                                                  2002012080115384178 9993'

  1 record(s) selected.
```

または、DBA は MON_GET_SECTION_ROUTINE 表関数を使用して無名ブロックに対応する内部プロシージャーとスキーマ名を検索し、それらの値を MON_GET_ROUTINE 表関数への入力として渡すこともできます。この方法の場合に戻されるのは無名ブロックに関する情報だけなので、WHERE 節を使用して出力をフィルター処理する必要はありません。

例: ルーチンのステートメント・テキストの取得

ルーチン・モニターを使用すると、ルーチンによって実行されたステートメントを取得できます。

シナリオ

この例では、データベース管理者 (DBA) はルーチンによって実行されたコストの高いステートメントを手動で調査しています。調査の一環として、MON_GET_ROUTINE_EXEC_LIST 内の行をルーチンの特定のステートメントまた

は行にリンクすると役立つことがあります。無名ブロック、動的 SQL ステートメント、または外部ルーチンなどの存続期間の短いルーチンのステートメント・テキストを取得するには、パッケージ・キャッシュのステートメントを取得します。以下の照会は、MON_GET_ROUTINE_EXEC_LIST 内の行を特定のステートメントにリンクします。

```
SELECT
   A.ROUTINETYPE, A.ROUTINESCHEMA, A.ROUTINENAME,
   A.SECTION_TYPE, A.SECTION_NUMBER, A.STMTNO,
   SUBSTR(B.STMT_TEXT,1,160)
FROM
   TABLE(MON_GET_ROUTINE_EXEC_LIST('P','DRICARD',NULL,'PROC1',-1)) AS A,
   TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,NULL,-1)) AS B
WHERE
   A.EXECUTABLE_ID=B.EXECUTABLE_ID
```

返される結果は、次のようになります。

```
STMT_TEXT
 ------------------------------------------
WITH GET_UPDATE_LIST (COL1, COL2STATS) AS AF
insert into T1 values(3,'d','d','d')
call SYSIBMSUBROUTINE.P1_66613_1157394573()

  3 record(s) selected.
```

注: 動的 SQL ステートメントまたは外部ルーチン・ステートメントが MON_GET_ROUTINE_EXEC_LIST によって返され、関連付けられた **executable_id** がパッケージ・キャッシュにもうない場合には、イベント・モニターを使用してその情報をログに記録していない限りは、対象のステートメント・テキストをリカバリーすることはできません。 InfoSphere Optim Performance Manager 製品のリフレッシュ・サイクルでは、ほとんどの場合、この情報を取得できます。

コンパイル済み SQL ステートメントおよびインライン化されたルーチンの場合、MON_GET_ROUTINE_EXEC_LIST によって返されるパッケージおよびステートメントの情報を使用してステートメント・テキストを検出できます。例えば、

```
SELECT  RS.ROUTINETYPE, RS.ROUTINESCHEMA, RS.ROUTINENAME,
   RS.SECTION_NUMBER, RS.STMTNO, SUBSTR(SS.TEXT,1,160)
FROM
   TABLE(MON_GET_ROUTINE_EXEC_LIST('F','DRICARD','','MYFUNC',-1)) AS RS,
   SYSIBM.SYSSTMT SS
WHERE
   RS.SECTION_TYPE = 'S'
    AND SS.PLNAME = RS.PACKAGE_SCHEMA
    AND SS.PLCREATOR = RS.PACKAGE_NAME
    AND SS.STMTNO = RS.STMTNO
    AND SS.SECTNO = RS.SECTION_NUMBER
```

返される結果は、次のようになります。

```
STMT_TEXT
 ------------------------------------------
insert into MYTABLE values('1')

  1 record(s) selected.
```

その他のモニター表関数

表関数には、システム、アクティビティー、ロック、またはデータ・オブジェクトの情報を返すものだけでなく、それ以外のさまざまなタイプの情報を返すものがあります。例えば、高速コミュニケーション・マネージャー (FCM) に関連した情報や、表スペースのエクステントの移動の状況に関する情報を返す表関数です。

以下のそれぞれの表関数はいつでも使用可能です。要求メトリック (システム・モニター・パースペクティブ)、アクティビティー・メトリック (アクティビティー・モニター・パースペクティブ)、またはデータ・オブジェクトに関連したメトリック (データ・オブジェクト・モニター・パースペクティブ) を返す表関数とは異なり、以下の関数が返すモニター・エレメントの収集を最初に有効にする必要はありません。

- MON_GET_FCM
- MON_GET_FCM_CONNECTION_LIST
- MON_GET_EXTENT_MOVEMENT_STATUS

モニター・データを XML 文書で返すインターフェース

DB2 バージョン 9.7 以降、一部のモニター・データが XML 文書のエレメントとして報告されるようになりました。

XML を使用してモニター情報を報告するという方法は、拡張性と柔軟性を高めます。出力表に新しい列を追加する必要なしに、新しいモニター・エレメントを追加できます。また、必要に応じて XML 文書をさまざまに処理できます。例えば、

- XQuery を使用して、XML 文書に対して照会を実行できます。
- XSLTRANSFORM スカラー関数を使用して、文書を他のフォーマットに変換できます。
- 組み込み MON_FORMAT_XML_* フォーマット関数または XMLTABLE 表関数を使用して、内容をフォーマット済みテキストとして表示できます。

モニター・エレメントを含んだ XML 文書は、いくつかのモニター・インターフェースによって生成されます。以下のセクションでは、結果がどのように XML 文書として返されるかを説明します。

- 『名前の終わりが「_DETAILS」のモニター表関数』
- 23 ページの『イベント・モニターによって返される XML データ』.

名前の終わりが「_DETAILS」のモニター表関数

これらの表関数の例として、以下の関数があります。

- MON_GET_PKG_CACHE_STMT_DETAILS
- MON_GET_WORKLOAD_DETAILS
- MON_GET_CONNECTION_DETAILS
- MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_ACTIVITY_DETAILS
- MON_GET_UNIT_OF_WORK_DETAILS

これらの表関数は、システムとアクティビティーのモニターの観点からのモニター・エレメントを返します。これらの関数が返すモニター・エレメントのほとんどは、XML 文書に含められます。例えば、MON_GET_CONNECTION_DETAILS 表関数は、以下の列を返します。

- APPLICATION_HANDLE
- MEMBER
- DETAILS

各行の DETAILS 列には、モニター・エレメント・データのある XML 文書が含まれます。この XML 文書は、モニター・エレメントに対応するいくつかの文書エレメントで構成されます。 22 ページの図 1 は、XML 文書が含まれる DETAILS 列を図示したものです。さらに、DETAILS 列の XML 文書で返されるモニター・エレメントも示しています。

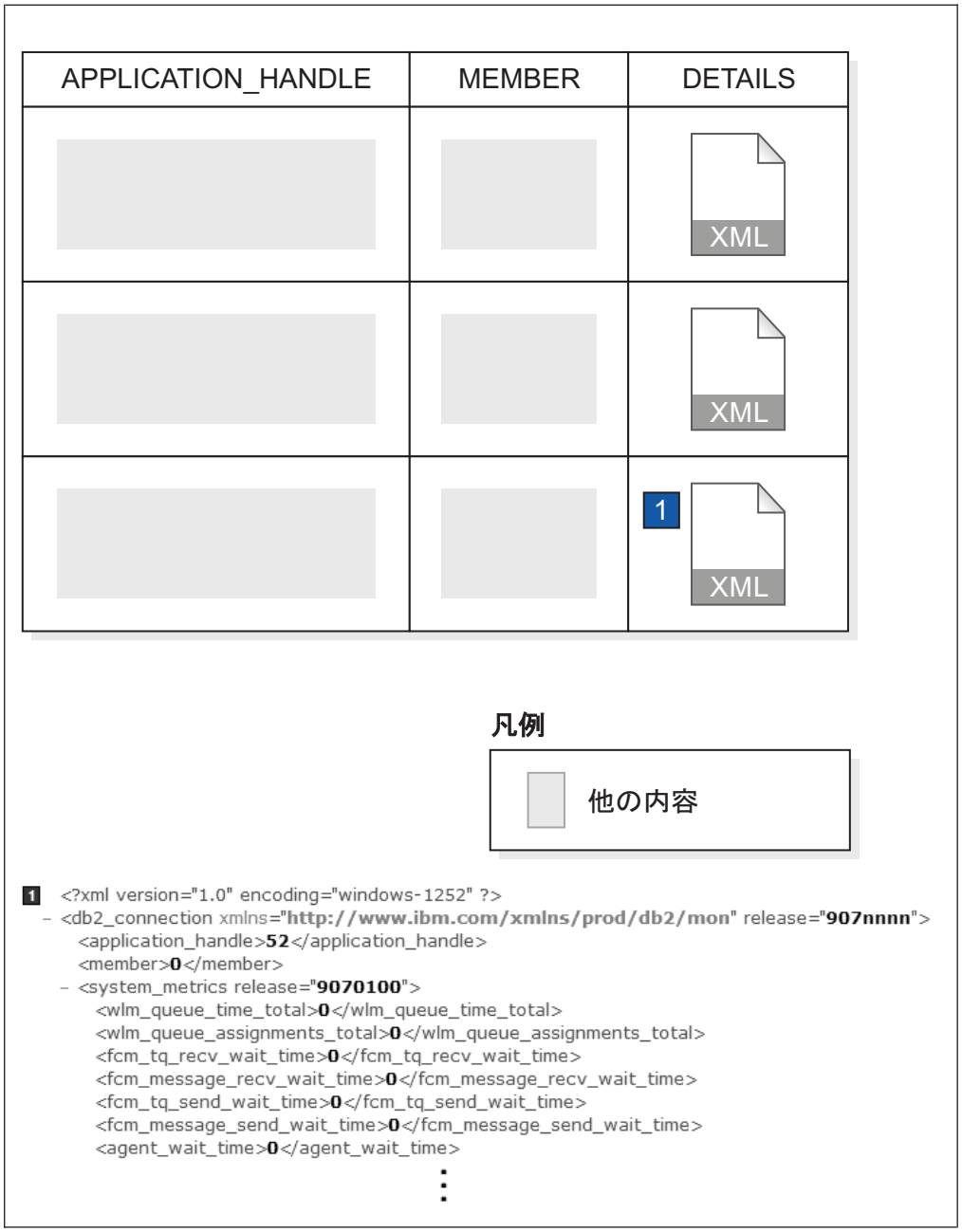


図 1. MON_GET_CONNECTION_DETAILS によって返される表における XML 文書を含んだ DETAILS 列: 第 3 行の XML 文書の内容 (**1**) を表に続いて示しています。

この例では、XML 文書の <agent_wait_time> エレメントは、**agent_wait_time** モニター・エレメントに対応します。

DETAILS 列で返される XML 文書のスキーマは、`sqllib/misc/DB2MonRoutines.xsd` ファイルにあります。さらに詳細なものが、`sqllib/misc/DB2MonCommon.xsd` ファイルにあります。

DETAILS 列の文書に含まれるモニター・エレメントのうちのいくつかは、より上位の文書エレメントにグループ化される場合があります。例えば、アクティビティー関連のメトリックについて報告する各モニター・エレメントは、**activity_metrics**

エレメントの一部です。同様に、システム・レベルのメトリックは、**system_metrics** エレメントの一部です。

イベント・モニターによって返される XML データ

いくつかのイベント・モニターは、XML 形式でデータを返します。これについては、表 2で要約されています。以下のセクションでは、さまざまなイベント・モニターによって返される XML 文書の詳細について説明します。

表 2. さまざまなイベント・モニターによって返される XML 文書

イベント・モニター	イベント・モニター出力形式	返される XML 文書(注: この一連のトピックでは、小文字の details_xml は XML 文書 details_xml を表します。大文字の DETAILS_XML はリレーショナル表の DETAILS_XML という列を表し、この列には details_xml 文書が含まれます。)
『統計イベント・モニター』	リレーショナル表 ファイル 名前付きパイプ	**metrics** この文書に報告されるメトリックは、統計が最後に収集された時からの各メトリックの値の変化を示します。 **details_xml** この文書で報告されるメトリックは、データベースが非活動化されるまで累積されます。
25 ページの『アクティビティー・イベント・モニター』	リレーショナル表 ファイル 名前付きパイプ	details_xml
25 ページの『パッケージ・キャッシュ・イベント・モニター』	未フォーマット・イベント (UE) 表	メトリック この文書は、 UE 表を XML またはリレーショナル表のいずれかに変換しないと表示できません。
26 ページの『作業単位イベント・モニター』	未フォーマット・イベント (UE) 表	メトリック この文書は、 UE 表を XML またはリレーショナル表のいずれかに変換しないと表示できません。

統計イベント・モニター

統計イベント・モニターは、イベント・モニター出力に以下の 2 つの論理データ・グループのいずれかが含まれる場合には、XML 形式でメトリックを記録します。

- EVENT_SCSTATS
- EVENT_WLSTATS

これらのグループのうち、いずれかのグループのモニター・エレメントについて報告する統計イベント・モニターを作成すると、いくつかのシステム・メトリックが 2 つの XML 文書の一部として収集されます。 **details_xml** モニター・エレメントおよび **metrics** モニター・エレメントのそれぞれの XML 文書です。どちらの

XML 文書にも、同じセットのモニター・エレメントが含まれています。metrics 文書の場合、エレメントの値は、統計が最後に収集された時からの各エレメントの値の変化を示します。details_xml に含まれているエレメントの値は、間隔ごとにリセットされることはありません。リセットされるのは、データベースが再活動化される場合だけです。 データがファイルまたは名前付きパイプに書き込まれると、これらのエレメントは自己記述型データ・ストリームの一部になります。イベント・モニター・データが表に書き込まれる場合には、metrics 文書は METRICS という列に、details_xml は DETAILS_XML という列に保管されます。図 2 は、統計イベント・モニターによって生成された SCSTATS 表に表示される METRICS 列と DETAILS_XML 列の XML 文書を示しています。

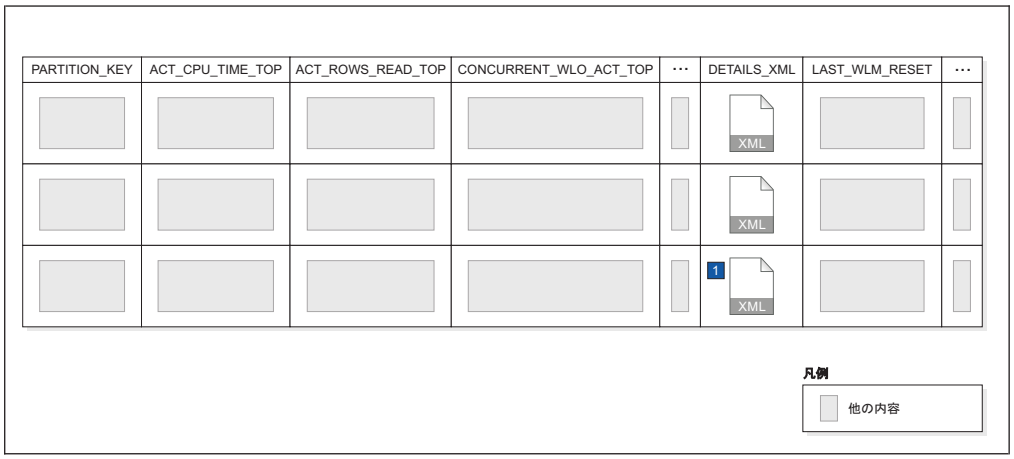


図 2. 表に書き込まれた統計イベント・モニター出力における DETAILS_XML および METRICS 列： 第 3 行の XML 文書の内容 (**1**) を表に続いて示しています。

これらの列に含まれている各文書には最上位エレメントとして **system_metrics** が含まれており、このエレメントには、システム関連メトリックについて報告するいくつかのモニター・エレメントが含まれています。

```
1  <?xml version="1.0" encoding="windows-1252" ?>
  - <db2_connection xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="907nnnn">
      <application_handle>52</application_handle>
      <member>0</member>
    - <system_metrics release="9070100">
        <wlm_queue_time_total>0</wlm_queue_time_total>
        <wlm_queue_assignments_total>0</wlm_queue_assignments_total>
        <fcm_tq_recv_wait_time>0</fcm_tq_recv_wait_time>
        <fcm_message_recv_wait_time>0</fcm_message_recv_wait_time>
        <fcm_tq_send_wait_time>0</fcm_tq_send_wait_time>
        <fcm_message_send_wait_time>0</fcm_message_send_wait_time>
        <agent_wait_time>0</agent_wait_time>
                                    ⋮
```

metrics モニター・エレメントの XML 文書でシステム・メトリックを確認できる他、EVENT_SCMETRICS 論理データ・グループおよび EVENT_WLMETRICS 論理データ・グループに関連する出力から直接、個々のメトリックを確認することもできます。

注:

- 統計イベント・モニターによって生成される DETAILS_XML 列に含まれる details_xml 文書で報告される **system_metrics** エレメントは、MON_GET_SERVICE_SUBCLASS_DETAILS および MON_GET_WORKLOAD_DETAILS 表関数が返す DETAILS 列にある XML 文書の一部でもあります。details_xml 文書で報告されるメトリック同様、DETAILS 列に含まれる文書で報告されるメトリックの値は、データベースが非活動化されるまで累積されます。
- 統計イベント・モニターからの XML 出力のスキーマについては、374 ページの『system_metrics および activity_metrics モニター・エレメント用に XML に書き込まれる情報』を参照してください。

アクティビティー・イベント・モニター

event_activity 論理データ・グループ (55 ページの『event_activity 論理データ・グループ』を参照) のモニター・エレメントについて報告するアクティビティー・イベント・モニターを作成した場合、生成される列の 1 つは DETAILS_XML になります。イベント・モニターが表に書き込まれる場合、DETAILS_XML は 1 つの列になります。ファイルまたは Named PIPE に書き込まれる場合、DETAILS_XML は自己記述型データ・ストリームの一部になります。いずれにしても、文書には **activity_metrics** モニター・エレメントが含まれ、そのエレメント自体に、アクティビティー関連のメトリックについて報告するいくつかのモニター・エレメントが含まれます。アクティビティー・イベント・モニターからの XML 出力のスキーマについては、374 ページの『system_metrics および activity_metrics モニター・エレメント用に XML に書き込まれる情報』を参照してください。

注: アクティビティー・イベント・モニターが生成する DETAILS_XML 列の XML 文書で報告される activity_metrics は、MON_GET_ACTIVITY_DETAILS 表関数が返す DETAILS 列にある XML 文書の一部でもあります。

パッケージ・キャッシュ・イベント・モニター

パッケージ・キャッシュ・イベント・モニターは、その出力を未フォーマット・イベント (UE) 表に書き込みます。この表のデータを EVMON_FORMAT_UE_TO_TABLES 表関数を使用して変換した場合、生成される表の 1 つは PKGCACHE_EVENT です。この表には、METRICS 列が含まれます。各行のこの列には、パッケージ・キャッシュ・イベント・モニター・エレメントに関連したエレメントのある XML 文書が含まれます。

注: DB2 バージョン 9.7 フィックスパック 1 以降、EVMON_FORMAT_UE_TO_TABLES は、このイベント・モニターによって収集されたメトリックを対象とした別個の表 (PKGCACHE_METRICS) も作成するようになりました。この表には、PKGCACHE_EVENT 表の METRICS 列で報告されるその同じ情報が含まれます。したがって、PKGCACHE_METRICS 表の列からメトリックを取得することも、PKGCACHE_EVENT 表の METRICS 列にある XML 文書を使用することもできます。詳しくは、290 ページの『パッケージ・キャッシュ・イベント・モニターの場合に EVMON_FORMAT_UE_TO_TABLES によってリレーショナル表に書き込まれる情報』を参照してください。

EVMON_FORMAT_UE_TO_XML 関数は、パッケージ・キャッシュ・イベント・モニター・エレメントに関連したエレメントのある XML 文書も生成します。例え

ば、XML 文書エレメント <num_executions> は、**num_executions** モニター・エレメントに対応します。パッケージ・キャッシュ・イベント・モニターからの XML 出力のスキーマについては、 300 ページの『パッケージ・キャッシュ・イベント・モニター用に XML に書き込まれる情報』を参照してください。

作業単位イベント・モニター

作業単位イベント・モニターは、その出力を未フォーマット・イベント (UE) 表に書き込みます。この表のデータを EVMON_FORMAT_UE_TO_TABLES 表関数を使用して変換した場合、生成される表の 1 つは UOW_EVENT です。この表には、作業単位イベント・モニター・エレメントに関連したエレメントのある XML 文書を含んだ METRICS 列があります。

注: DB2 バージョン 9.7 フィックスパック 1 以降、EVMON_FORMAT_UE_TO_TABLES は、このイベント・モニターによって収集されたメトリックを対象とした別個の表 (UOW_METRICS) も作成するようになりました。この表には、UOW_EVENT 表の METRICS 列で報告されるその同じ情報が含まれます。したがって、UOW_METRICS 表の列からメトリックを取得することも、UOW_EVENT 表の METRICS 列にある XML 文書を使用することもできます。詳しくは、 236 ページの『作業単位イベント・モニターの場合に EVMON_FORMAT_UE_TO_TABLES によってリレーショナル表に書き込まれる情報』を参照してください。

EVMON_FORMAT_UE_TO_XML 関数は、作業単位イベント・モニター・エレメントに関連したエレメントのある XML 文書も生成します。例えば、XML 文書エレメント <workload_name> は、**workload_name** モニター・エレメントに対応します。作業単位イベント・モニターからの XML 出力のスキーマについては、 248 ページの『作業単位イベント・モニターの場合に EVMON_FORMAT_UE_TO_XML によって XML に書き込まれる情報』を参照してください。

XML モニター情報をフォーマット済みテキストとして表示するためのインターフェース

モニター・インターフェースによって生成された XML 文書に含まれるデータを、データをどう表示するかやどう使用するかに応じて、さまざまに表示できます。

XQuery を使用して、モニター・インターフェースが返した XML 文書を照会および操作できます。表関数を使用して、XML 文書を読みやすいフォーマットにすることもできます。

XQuery は、XML データを照会および操作するための強力で柔軟なインターフェースを備えています。一方、エレメント・データをテキスト・ベースのフォーマットで表示した方がよい場合もあります。 XML 文書に含まれるモニター・エレメントを、必要に応じて列指向または行指向のフォーマットで表示できます。前者は、調べるモニター・エレメントが分かっている場合に便利です。後者は、例えば待機時間の上位 5 タイプを知りたいときなど、調べるモニター・エレメントが前もって分かっていない場合に便利です。以下のセクションでは、XML 文書に含まれるモニター・データをフォーマット済みテキストとして表示するための 2 つの方法を説明します。

- 27 ページの『列指向フォーマットでモニター・エレメントを表示する』

• 28 ページの『行指向フォーマットでモニター・エレメントを表示する』

列指向フォーマットでモニター・エレメントを表示する

XMLTABLE 表関数は XML 文書を入力として取り、それをリレーショナル表に変換することによって、選択された XML 文書エレメントのそれぞれが列に見えるようにします。このアプローチは、表示したいモニター・エレメントが分かっている場合に便利です。例えば、event_scstats 論理データ・グループから情報を収集するための DBSTATS という統計イベント・モニターを作成したとします。 (この論理データ・グループに関連付けられるモニター・エレメントについて詳しくは、 89 ページの『event_scstats 論理データ・グループ』を参照してください。) この論理グループのモニター・エレメントには **details_xml** が含まれます。details_xml は実際は XML 文書であり、これ自体に **system_metrics** モニター・エレメントを構成するメトリックが含まれます。(system_metrics モニター・エレメントに関連付けられるモニター・エレメントについて詳しくは、 374 ページの『system_metrics』を参照してください。) rows_returned や total_section_time、total_cpu_time など、details_xml に含まれる特定の **system_metrics** モニター・エレメントを表示するには、XMLTABLE 表関数を使用して、統計イベント・モニターが返す details_xml 文書から選択したモニター・エレメントをフォーマット設定します。この例を、以下に示します。 (説明の目的上、SQL は特定のサービス・クラスの結果のみを返しています。)

```
SELECT partition_number,
       service_class_id,
       statistics_timestamp,
       event.rows_returned,
       event.total_section_time,
       event.total_cpu_time
 FROM  SCMETRICS_DBSTATS as DBSTATS,
 XMLTABLE( XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon' ),
           '$metrics/system_metrics' PASSING XMLPARSE( DOCUMENT DBSTATS.METRICS ) as "metrics"
           COLUMNS
           rows_returned         BIGINT      PATH 'rows_returned',
           total_section_time    BIGINT      PATH 'total_section_time',
           total_cpu_time        BIGINT      PATH 'total_cpu_time'
         ) AS EVENT
 WHERE service_class_id = 12;
```

以下の出力は、この照会の結果を示しています。

```
PARTITION_NUMBER SERVICE_CLASS_ID STATISTICS_TIMESTAMP       ROWS_RETURNED        TOTAL_SECTION_TIME   TOTAL_CPU_TIME
---------------- --------------- -------------------------- -------------------- -------------------- --------------------
               0              12 2010-01-05-12.14.37.001717                  402                  990              1531250
               0              12 2010-01-05-12.15.00.035409                  402                  990              1531250
               0              12 2010-01-05-12.20.00.021884                  412                 1064              1609375
               0              12 2010-01-05-12.25.00.039175                  422                 1075              1687500
               0              12 2010-01-05-12.29.59.950137                  432                 1104              1765625
               0              12 2010-01-05-12.34.59.948979                  442                 1130              1796875
               0              12 2010-01-05-12.39.59.903928                  452                 1149              1890625
               0              12 2010-01-05-12.44.59.953596                  462                 1178              1953125
               0              12 2010-01-05-12.49.59.970059                  473                 1207              2062500
               0              12 2010-01-05-12.54.59.971990                  483                 1230              2109375

  10 record(s) selected.
```

1. 注: この一連のトピックでは、小文字の details_xml は XML 文書 details_xml を表します。大文字の DETAILS_XML はリレーショナル表の DETAILS_XML という列を表し、この列には details_xml 文書が含まれます。

この場合、最初の 3 つの列は、統計イベント・モニターが生成した表 SCSTATS_DBSTATS から直接表示されたものです。終わりの 3 つの列は、この表の DETAILS_XML 列の XML 文書から抽出されたメトリック・モニター・エレメントです。

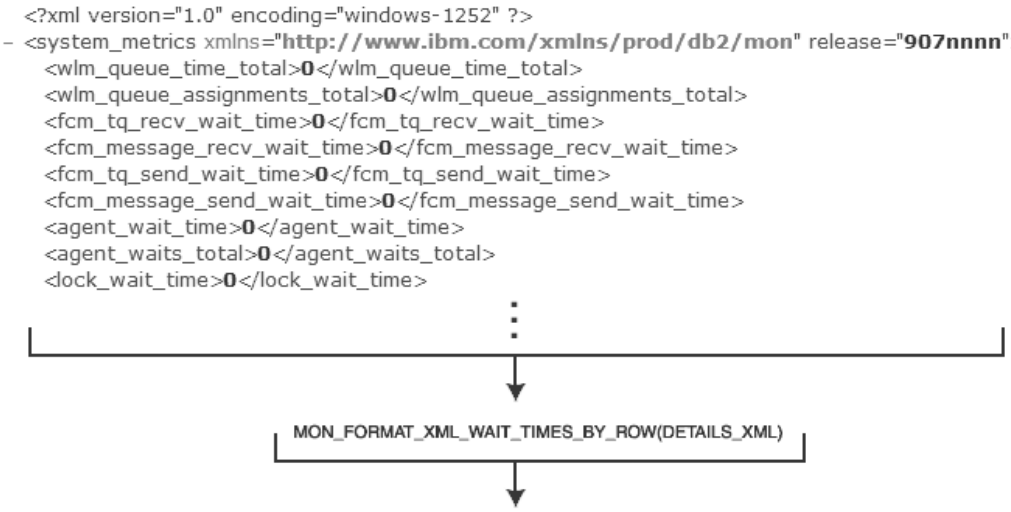
XMLTABLE の使用法について詳しくは、XMLTABLE 関数に関する資料を参照してください。 MON_GET_*_DETAILS 各種関数の資料にも、XMLTABLE を使用してモニター・エレメントを表示する方法の例が示されています。

行指向フォーマットでモニター・エレメントを表示する

DB2 バージョン 9.7 フィックスパック 1 で導入された、名前が MON_FORMAT_XML_* _BY_ROW の形式の表関数を、XML 文書に含まれるメトリック・モニター・エレメントを表示するための手っ取り早い方法として使用できます。これらは行ベースのフォーマットでメトリックを報告し、各モニター・エレメントが単独の行で表示されます。このグループには、以下の関数が含まれます。

- MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW
- MON_FORMAT_XML_TIMES_BY_ROW
- MON_FORMAT_XML_WAIT_TIMES_BY_ROW
- MON_FORMAT_XML_METRICS_BY_ROW

例えば、統計イベント・モニターから返される XML 文書 DETAILS_XML が、29 ページの図 3 の最初の部分に示すようなものだったとします。
MON_FORMAT_XML_WAIT_TIMES_BY_ROW 関数を使用して DETAILS_XML の内容をフォーマット設定すると、出力はダイアグラムの下部に示す表のようになります。




```
METRIC_NAME                         TOTAL_TIME_VALUE     COUNT                PARENT_METRIC_NAME
----------------------------------- -------------------- -------------------- ----------------------------
WLM_QUEUE_TIME_TOTAL                                  0                    0 TOTAL_WAIT_TIME
FCM_TQ_RECV_WAIT_TIME                                 0                    0 FCM_RECV_WAIT_TIME
FCM_MESSAGE_RECV_WAIT_TIME                            0                    0 FCM_RECV_WAIT_TIME
FCM_TQ_SEND_WAIT_TIME                                 0                    0 FCM_SEND_WAIT_TIME
FCM_MESSAGE_SEND_WAIT_TIME                            0                    0 FCM_SEND_WAIT_TIME
AGENT_WAIT_TIME                                       0                    0 TOTAL_WAIT_TIME
LOCK_WAIT_TIME                                        0                    0 TOTAL_WAIT_TIME
DIRECT_READ_TIME                                      0                    0 TOTAL_WAIT_TIME
DIRECT_WRITE_TIME                                     0                    0 TOTAL_WAIT_TIME
LOG_BUFFER_WAIT_TIME                                  0                    0 TOTAL_WAIT_TIME
LOG_DISK_WAIT_TIME                                    0                    0 TOTAL_WAIT_TIME
```

⋮

図 3. モニター・データを含んだ XML ファイルをいずれかの MON_FORMAT_XML_* 関数で処理する： この例は、MON_FORMAT_XML_WAIT_TIMES_BY_ROW 関数を使用した場合を示しています。待機時間のみが返され、wlm_queue_assignments_total など XML ファイルに含まれる他のメトリックは、この特定の関数によって除外されます。

返される列の数は、使用する具体的な関数によって異なります。例えば、MON_FORMAT_XML_METRICS_BY_ROW は、以下のように 2 つの列を返します。1 つはメトリック名の列、もう 1 つはそれに対応する値の列です。

```
METRIC_NAME               VALUE
------------------------- --------------------
WLM_QUEUE_TIME_TOTAL                         0
WLM_QUEUE_ASSIGNMENTS_TOT                    0
FCM_TQ_RECV_WAIT_TIME                        0
FCM_MESSAGE_RECV_WAIT_TIM                    0
FCM_TQ_SEND_WAIT_TIME                        0
```

⋮

それに対して MON_FORMAT_XML_TIMES_BY_ROW は、以下のように 4 つの列を返します。

```
METRIC_NAME                 TOTAL_TIME_VALUE     COUNT      PARENT_METRIC_NAME
--------------------------- -------------------- ---------- ----------------------
WLM_QUEUE_TIME_TOTAL                         0          0 TOTAL_WAIT_TIME
FCM_TQ_RECV_WAIT_TIME                        0          0 FCM_RECV_WAIT_TIME
FCM_MESSAGE_RECV_WAIT_TIME                   0          0 FCM_RECV_WAIT_TIME
FCM_TQ_SEND_WAIT_TIME                        0          0 FCM_SEND_WAIT_TIME
FCM_MESSAGE_SEND_WAIT_TIME                   0          0 FCM_SEND_WAIT_TIME
```

⋮

MON_FORMAT_XML_* _BY_ROW 関数は、表示したいエレメントが分かっていないときに便利です。例えば、CLPWORKLOAD というワークロードの待機時間モニター・エレメントの上位 10 エレメントを調べることもできます。この情報を収集するには、DBSTATS (event_wlstats 論理データ・グループ) という統計イベント・モニターを作成します。このイベント・モニターを表に書き込むようにセットアップしたとすると、メトリックは DETAILS_XML という列に記録されます。以下のように、イベント・モニターからの出力表にモニター・データを設定したら、MON_FORMAT_XML_WAIT_TIMES_BY_ROW 関数を使用する照会を構成することによって、調べたいモニター・エレメントを抽出できます。

```
SELECT SUBSTR(STATS.WORKLOAD_NAME,1,15) AS WORKLOAD_NAME,
       SUBSTR(METRICS.METRIC_NAME,1,30) AS METRIC_NAME,
       SUM(METRICS.TOTAL_TIME_VALUE) AS TOTAL_TIME_VALUE
FROM   WLSTATS_DBSTATS AS STATS,
       TABLE(MON_FORMAT_XML_WAIT_TIMES_BY_ROW(STATS.DETAILS_XML)) AS METRICS
WHERE  WORKLOAD_NAME='CLPWORKLOAD' AND (PARENT_METRIC_NAME='TOTAL_WAIT_TIME')
GROUP  BY WORKLOAD_NAME,METRIC_NAME
ORDER  BY TOTAL_TIME_VALUE DESC
FETCH FIRST 10 ROWS ONLY
```

要確認: 消費時間モニター・エレメントは、階層的に編成されています。この例では、待機時間の二重カウントを避けるために、**total_wait_time** に累積されるモニター・エレメントのみを含めています (上記 SQL ステートメントの WHERE 節を参照)。そうしないと、個別の待機時間をいくつも含んでいる **total_wait_time** 自体が結果に入ってしまうことになります。
この照会の結果は、例えば以下のような出力になります。

```
WORKLOAD_NAME   METRIC_NAME                    TOTAL_TIME_VALUE
--------------- ------------------------------ --------------------
CLPWORKLOAD     LOCK_WAIT_TIME                        15138541
CLPWORKLOAD     DIRECT_READ_TIME                       6116231
CLPWORKLOAD     POOL_READ_TIME                         6079458
CLPWORKLOAD     DIRECT_WRITE_TIME                       452627
CLPWORKLOAD     POOL_WRITE_TIME                         386208
CLPWORKLOAD     IPC_SEND_WAIT_TIME                      283172
CLPWORKLOAD     LOG_DISK_WAIT_TIME                      103888
CLPWORKLOAD     DIAGLOG_WRITE_WAIT_TIME                  78198
CLPWORKLOAD     IPC_RECV_WAIT_TIME                       15612
CLPWORKLOAD     TCPIP_SEND_WAIT_TIME                      3291

  10 record(s) selected.
```

注: MON_FORMAT_XML_*_BY_ROW 関数は、測定つまりメトリック を追跡するモニター・エレメントのみを返します。これには、カウンターだけでなく、待機時間およびコンポーネント時間を追跡するモニター・エレメントも含まれます。uow_id や activity_id など、XML 文書に含まれる非メトリック・モニター・エレメントは返しません。

XMLTABLE 関数を使用することによって、XML 文書に含まれる*任意の* エレメント (非メトリック・エレメントを含む) を表示できます。ただし、最も頻繁に使用される非メトリック・モニター・エレメントは、MON_GET_UNIT_OF_WORK や MON_GET_CONNECTION など、MON_GET_* で始まるモニター関数で列として返されます。 XML をよくご存じない方は、これらの関数を使用して照会を作成したほうが、XMLTABLE 関数を使用して XML 文書からモニター・エレメントを抽出するよりも、速くて簡単かもしれません。

要約すると、次のようになります。非メトリック・モニター・エレメントを表示したい場合は、XMLTABLE 関数の代わりに MON_GET_* シリーズの表関数を使用したほうが望ましいことがあります。メトリック・モニター・エレメントを表示したい場合は、MON_FORMAT_XML_*_BY_ROW 表関数がニーズに適していることがあります。

XML 文書にあるメトリック・モニター・エレメントを表の行として表示する

イベント・モニターから返された XML 文書に含まれるメトリック関連の情報を表示する方法の 1 つに、各モニター・エレメントが単独の行で表示されるフォーマットに情報を変換するという方法があります。

このフォーマットは、テキスト・ベースのフォーマットで情報を表示したいが、調べるモニター・エレメントが具体的に分かっていない場合に便利です。

このタスクについて

さまざまなモニター・インターフェースから返される XML 文書にあるメトリック情報を行ベースのフォーマットで表示するには、MON_FORMAT_XML_*_BY ROW 表関数を使用します。これらの関数は、DB2 バージョン 9.7 フィックスパック 1 で導入されたものです。

手順

このタスクで示す例では、MON_FORMAT_XML_TIMES_BY_ROW 表関数を使用して、パッケージ・キャッシュ・イベント・モニターによって追跡されたステートメントのコンポーネント時間を表示します。 PKGCACHEEVENTS というパッケージ・キャッシュ・イベント・モニターを作成済みであり、活動化されているとします。パッケージ・キャッシュ・イベント・モニターは、その出力を未フォーマット・イベント (UE) 表に書き込みます。この出力情報を使用するには、UE 表のデータを EVMON_FORMAT_UE_TO_TABLES ストアード・プロシージャーを使用してリレーショナル表に変換するか、または EVMON_FORMAT_UE_TO_XML 表関数を使用して XML に変換する必要があります。このタスクでは、2 つのアプローチのうちの最初のアプローチを示します。

1. まず、EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用して、パッケージ・キャッシュ・イベント・モニターが書き込んだ未フォーマット・イベント (UE) 表を、リレーショナル表に変換します。

   ```
   call EVMON_FORMAT_UE_TO_TABLES ('PkgCache',NULL,NULL,NULL,NULL,NULL,
        NULL,0,'SELECT * FROM PKGCACHEEVENTS')
   ```

   このプロシージャーは、次の 2 つの表を作成します。

   - 1 つは PKGCACHE_EVENT という表であり、この表には METRICS という列があります。この列自体に、メトリック・モニター・エレメントのある XML 文書が含まれます。
   - もう 1 つは PKGCACHE_METRICS という表です。

     注: PKGCACHE_EVENT 表の METRICS 列からメトリックを抽出するのではなく、PKGCACHE_METRICS の列にあるメトリックを直接表示することもできます。ただし、PKGCACHE_METRICS を調べる場合は、行ではなく列にメ

トリックが表示されます。したがって、例えば値が大きいメトリックのランキングを把握するのは、それほど容易なことではありません。

2. 実行時間の点で最もコストがかかっているステートメントを判別するために、前のステップで生成された 2 つの表に対して次の照会を実行します。

```
SELECT EVENTS.EXECUTABLE_ID,
       SUM(METRICS.STMT_EXEC_TIME) AS TOTAL_STMT_EXEC_TIME
FROM   PKGCACHE_EVENT AS EVENTS,
       PKGCACHE_METRICS AS METRICS
WHERE  EVENTS.XMLID = METRICS.XMLID
GROUP  BY EVENTS.EXECUTABLE_ID
ORDER  BY TOTAL_STMT_EXEC_TIME DESC
FETCH  FIRST 5 ROWS ONLY
```

この照会では、ステップ 1 (31 ページ) で生成された 2 つの表が結合されます。その結果、PKGCACHE_EVENT 表にある各ステートメント ID が、PKGCACHE_METRICS 表にあるそれぞれの実行時間と関連付けられ、以下のようになります。

```
EXECUTABLE_ID                                                                    TOTAL_STMT_EXEC_TIME
-------------------------------------------------------------------------------- --------------------
x'0100000000000001A0300000000000000000000020020091215115933859000'                  250
x'0100000000000001503000000000000000000002002009121511585032800 0'                  191
x'0100000000000002102000000000000000000002002009121511581834300 1'                  129
x'0100000000000000C4020000000000000000000020020091215115838578000'                   41
x'0100000000000000B0020000000000000000000020020091215115838203000'                   38

  5 record(s) selected.
```

結果の最初の項目が、全体実行時間が最大のステートメントを表しています。

3. オプション: 必要であれば、そのステートメントのテキストを、次の SQL を使用することによって表示できます。

```
SELECT SUBSTR(STMT_TEXT,1,60) AS STMT_TEXT
FROM PKGCACHE_EVENT
WHERE EXECUTABLE_ID = x'0100000000000001A0300000000000000000000020020091215115933859000'
```

結果:

```
STMT_TEXT
------------------------------------------------------------
DROP XSROBJECT MYSCHEMA.EVMON_PKGCACHE_SCHEMA_SQL09070

  1 record(s) selected.
```

4. ステップ 2 で識別したステートメントの消費時間モニター・エレメントのリストを、次のように MON_FORMAT_XML_TIMES_BY_ROW 表関数を使用して表示します。

```
SELECT SUBSTR(XMLMETRICS.METRIC_NAME,1,30) AS METRIC_NAME,
       XMLMETRICS.TOTAL_TIME_VALUE,
       SUBSTR(XMLMETRICS.PARENT_METRIC_NAME,1,30) AS PARENT_METRIC_NAME
FROM PKGCACHE_EVENT AS EVENTS,
     TABLE(MON_FORMAT_XML_TIMES_BY_ROW(EVENTS.METRICS)) AS XMLMETRICS
WHERE EVENTS.EXECUTABLE_ID=
x'0100000000000001A0300000000000000000000020020091215115933859000'
 AND PARENT_METRIC_NAME='STMT_EXEC_TIME'
ORDER BY XMLMETRICS.TOTAL_TIME_VALUE DESC
```

注:

- 消費時間モニター・エレメントは階層的に編成されていることに注意してください。二重カウントをなくすために、**stmt_exec_time** に累積されるメトリッ

クのみを結果に含めています。そうしないと、個別のコンポーネント時間をいくつも含んでいる stmt_exec_time 自体が結果に入ってしまうことになります。

- PARENT_METRIC_NAME (MON_FORMAT_XML_TIMES_BY_ROW が返す列の 1 つ) を使用しているのは、例示が目的です。

実行すると、この照会から次の結果が返されます。

```
METRIC_NAME                  TOTAL_TIME_VALUE     PARENT_METRIC_NAME
---------------------------- -------------------- ----------------------------
TOTAL_ACT_WAIT_TIME                           234 STMT_EXEC_TIME
TOTAL_SECTION_PROC_TIME                        15 STMT_EXEC_TIME
```

これから、合計処理時間が 249 ms になることが分かります。この時間をステップ 2 (32 ページ) で示された合計時間の 250 と比較した場合の余分のミリ秒は、**stmt_exec_time** に含まれていない他の時間 (例えば待機) と見なすことができます。

タスクの結果

前の例の結果は、メトリックが特別な配置になっているのが分かります。つまり、行指向フォーマットになっており、1 行につき 1 つのメトリックが表示されています。このアプローチを使用することのメリットは、調べるメトリックつまりモニター・エレメントを前もって分かっておく必要がないという点にあります。時間を消費するメトリックの内の上位 5 個のメトリック、または特定の値範囲に入るメトリックを確認する場合は、知りたい結果を返す照会を簡単に作成できます。一方、XMLTABLE 関数を使用してモニター・エレメントを列として表示する場合は、表示するモニター・エレメントを指定する (またはすべてを表示する) 必要があります。

例

MON_GET_*_DETAILS 表関数によって生成された DETAILS 列の内容を表示する

MON_FORMAT_XML_*_BY_ROW 関数を使用して、MON_GET_*_DETAILS 関数のいずれかが返した DETAILS 列の内容を表示することもできます。例えば MON_GET_CONNECTION_DETAILS は、データベース接続に関連したメトリックのある XML 文書を含んだ DETAILS 列を返します。

例えば、全メンバーの各接続のゼロ以外のコンポーネント時間を表示するには、次の照会を使用できます。

```
SELECT CONDETAILS.APPLICATION_HANDLE,
       SUBSTR(XMLMETRICS.METRIC_NAME,1,30) AS METRIC_NAME,
       SUM(XMLMETRICS.TOTAL_TIME_VALUE) AS TOTAL_TIME_VALUE,
       SUBSTR(XMLMETRICS.PARENT_METRIC_NAME,1,30) AS PARENT_METRIC_NAME
FROM TABLE(MON_GET_CONNECTION_DETAILS(NULL,-1)) AS CONDETAILS,
     TABLE(MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW(CONDETAILS.DETAILS))AS XMLMETRICS
WHERE TOTAL_TIME_VALUE > 0 AND XMLMETRICS.PARENT_METRIC_NAME='TOTAL_RQST_TIME'
GROUP BY CONDETAILS.APPLICATION_HANDLE,
         XMLMETRICS.PARENT_METRIC_NAME,
         XMLMETRICS.METRIC_NAME
ORDER BY CONDETAILS.APPLICATION_HANDLE ASC, TOTAL_TIME_VALUE DESC
```

注:

- 二重カウントをなくすために、**total_rqst_time** に累積されるメトリックのみを結果に含めています (WHERE ....

XMLMETRICS.PARENT_METRIC_NAME='TOTAL_RQST_TIME')。そうしないと、個別のコンポーネント時間をいくつも含んでいる total_rqst_time 自体が結果に入ってしまうことになります。

- PARENT_METRIC_NAME
  (MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW が返す列の 1 つ) を使用しているのは、例示が目的です。

この照会は、以下の結果を返します。

```
APPLICATION_HANDLE   METRIC_NAME                   TOTAL_TIME_VALUE     PARENT_METRIC_NAME
-------------------- ------------------------------ -------------------- ------------------------------
                  52 TOTAL_SECTION_TIME                             3936 TOTAL_RQST_TIME
                  52 TOTAL_COMPILE_TIME                              482 TOTAL_RQST_TIME
                  52 TOTAL_COMMIT_TIME                                15 TOTAL_RQST_TIME
                  52 TOTAL_ROLLBACK_TIME                               1 TOTAL_RQST_TIME
                 496 TOTAL_COMPILE_TIME                              251 TOTAL_RQST_TIME
                 496 TOTAL_SECTION_TIME                               46 TOTAL_RQST_TIME
                 496 TOTAL_IMPLICIT_COMPILE_TIME                      5 TOTAL_RQST_TIME

  7 record(s) selected.
```

この例が示すとおり、**total_rqst_time** を構成するメトリックのみが含まれています。仮に WHERE .... XMLMETRICS.PARENT_METRIC_NAME='TOTAL_RQST_TIME' 節がこの照会に含まれていなかったとすると、結果は次のようなものになります。

```
APPLICATION_HANDLE   METRIC_NAME                   TOTAL_TIME_VALUE     PARENT_METRIC_NAME
-------------------- ------------------------------ -------------------- ------------------------------
                  52 TOTAL_RQST_TIME                                4603 -
                  52 TOTAL_SECTION_TIME                             3942 TOTAL_RQST_TIME
                  52 TOTAL_COMPILE_TIME                              537 TOTAL_RQST_TIME
                  52 TOTAL_SECTION_SORT_TIME                         299 TOTAL_SECTION_TIME
                  52 TOTAL_COMMIT_TIME                                15 TOTAL_RQST_TIME
                  52 TOTAL_ROLLBACK_TIME                               1 TOTAL_RQST_TIME
                 496 TOTAL_RQST_TIME                                 341 -
                 496 TOTAL_COMPILE_TIME                              251 TOTAL_RQST_TIME
                 496 TOTAL_SECTION_TIME                               46 TOTAL_RQST_TIME
                 496 TOTAL_IMPLICIT_COMPILE_TIME                      5 TOTAL_RQST_TIME
                 496 TOTAL_SECTION_SORT_TIME                          2 TOTAL_SECTION_TIME

  11 record(s) selected.
```

この場合は、各接続の **total_rqst_time** の値が結果に含まれていますが、親であるこの **total_rqst_time** は、子である他のすべてのエレメントの値を含んでいます。同様に、イタリック で示した項目の値は **total_section_time** に累積されています。これらは、WHERE 節で除外されていなければ、結果で三重カウントされることになります。**total_section_time** 自体が **total_rqst_time** に累積されているためです。

第 3 章 イベント・モニター

モニター表関数およびスナップショット・ルーチンは、ルーチンが実行された特定の時点におけるモニター・エレメントの値を戻します。これは、システムの現行状態を確認する場合に有用です。ただし、時間が経過する間、常にモニターすることが望ましくない場合があります。

特定のイベントが発生したときにだけ、システムの状態についての情報をキャプチャーすることが必要な場合は多くあります。イベント・モニターは、この目的に応えます。

イベント・モニターを作成して、システムで発生するさまざまな種類のイベント に関する、特定時点の情報をキャプチャーできます。例えば、定義した特定のしきい値が超過されたときに情報をキャプチャーするイベント・モニターを作成できます。キャプチャーされる情報には、しきい値が超過されたときに実行されていたアプリケーションの ID などがあります。また、イベント・モニターを作成して、ロック・イベントが発生したときに実行されていたステートメントを調べることもできます。

イベント・モニターがデータをキャプチャーするイベントのタイプ

イベント・モニターを使用すると、システムで発生する多様な種類のイベントに関する情報をキャプチャーできます。

次の表に、システムで発生するイベントで、イベント・モニターを使用してモニターできるイベントのタイプをリストします。また、さまざまなイベントに関して収集されるデータのタイプと、そのモニター・データが収集されるタイミングについても記載しています。2 列目に示しているイベント・モニター名は、そのタイプのイベント・モニターを CREATE EVENT MONITOR ステートメントによって作成するときに使用するキーワードに対応しています。

表 3. イベント・タイプ

モニターするイベントのタイプ	イベント・モニター名	イベント・モニターの特性	詳細
ロックおよびデッドロック	LOCKING	*このイベント・モニターの用途*	ロックまたはデッドロックがいつ発生するか、および関与しているアプリケーションを調べます。非推奨の DEADLOCKS イベント・モニターではなく LOCKING イベント・モニターを使用する利点には、ロック・イベントとデッドロック・イベントの両方に関する統合的なレポートが行われる点や、ロック待機およびロック・タイムアウトの情報も含んでいる点などがあります。
		収集されるデータ	関係するアプリケーションについての広範囲の情報。関係するステートメント (およびステートメント・テキスト) の識別や保持されているロックなど。
		イベント・データが生成されるとき[1]	イベント・モニターの構成方法に応じて、次のいずれかのイベント・タイプが検出されたとき • ロック・タイムアウト • デッドロック • 指定の期間を超えるロック待機
SQL ステートメント、またはデータベース・アクティビティーを生成するその他の操作の実行	ACTIVITIES	*このイベント・モニターの用途*	システムで実行されているアクティビティーを把握するために、個々のステートメントおよびその他のアクティビティーの実行を追跡します。また、診断目的でアクティビティーをキャプチャーしたり、 SQL のリソース消費量を調べたりもします。
		収集されるデータ	アクティビティー・レベルのデータ。一般的には、ワークロード管理オブジェクトに関与するアクティビティーに関するデータ。 • ワークロード管理オブジェクトに対する CREATE または ALTER ステートメントの COLLECT ACTIVITY DATA 節の一部として WITH DETAILS を指定した場合、収集される情報には、対象のアクティビティーに関するステートメントとコンパイル環境の情報が含まれます。 WITH SECTION も指定すると、ステートメント、コンパイル環境、セクション環境データ、およびセクション実行時統計もキャプチャーされます。 • また、ワークロード管理オブジェクトに対する CREATE または ALTER ステートメントに AND VALUES も指定した場合、収集される情報には、対象のアクティビティーの入力データ値も含まれます。
		イベント・データが生成されるとき[1]	• COLLECT ACTIVITY DATA オプションがオンになっているサービス・クラス、ワークロード、または作業クラス内で実行されたアクティビティーが完了したとき。 • COLLECT ACTIVITY DATA オプションが有効になっているアクティビティーがしきい値に違反したとき。 • WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーが実行された瞬間。 • WLM_SET_CONN_ENV ストアード・プロシージャーを使用してアクティビティー収集が有効にされた接続で、アクティビティーが実行されるとき。

表 3. イベント・タイプ (続き)

モニターするイベントのタイプ	イベント・モニター名	イベント・モニターの特性	詳細
SQL ステートメントの実行	STATEMENTS	*このイベント・モニターの用途*	SQL ステートメントの実行の結果としてデータベースに対して要求された内容を調べます。
		収集されるデータ	ステートメントの開始/停止時刻、使用されている CPU、動的 SQL のテキスト、 SQLCA (SQL ステートメントの戻りコード)、 およびその他のメトリック (フェッチ・カウントなど)。パーティション・データベースの場合: 使用された CPU、実行時間、表および表キューの情報。 **注:** • ステートメント・イベント・モニターを使用して SQL プロシージャーの実行をモニターする場合、 INSERT、SELECT、DELETE、UPDATE などのデータ操作言語 (DML) ステートメントは、イベントを生成します。変数割り当てや制御構造 (例えば WHILE や IF) などのプロシージャー・ステートメントは、決定論的にイベントを生成しません。 • ステートメントの開始/停止時刻は、 TIMESTAMP スイッチが OFF のときは使用できません。
		イベント・データが生成されるとき	SQL ステートメントの終了時[2]、パーティション・データベースの場合はサブセクションの終了時[2]
作業単位 (トランザクション) の完了	UNIT OF WORK	*このイベント・モニターの用途*	システムで実行された作業単位についてのリソース使用量情報およびパフォーマンス・メトリックを収集します。この情報の用途は、アプリケーションで使用されたシステム・リソースに関して請求またはチャージバックを行うためにレポートを生成したり、実行の遅いルーチンによるパフォーマンス問題をトラブルシューティングしたりと多岐にわたります。 TRANSACTIONS イベント・モニターに代わって推奨されます。
		収集されるデータ	作業単位 (トランザクション) についての情報。開始時刻、終了時刻、実行元のワークロードおよびサービス・クラスなど。オプションで、作業単位の一部として実行されたステートメントのパッケージまたは実行可能 ID に関する情報を、要求メトリックと一緒に含めることができます。
		イベント・データが生成されるとき[1]	作業単位の完了時
パッケージ・キャッシュからのセクションの追い出し	PACKAGE CACHE	*このイベント・モニターの用途*	パッケージ・キャッシュからなくなったステートメントの履歴 (および関連するメトリック) をキャプチャーします。この情報は、メモリーからなくなったステートメントのパフォーマンス・メトリックを調べる必要がある場合に使用できます。
		収集されるデータ	ステートメント・テキストと、そのセクションのすべての実行について集約したメトリックが含まれます。
		イベント・データが生成されるとき[1]	パッケージ・キャッシュから項目が追い出されたとき

表 3. イベント・タイプ (続き)

モニターするイベントのタイプ	イベント・モニター名	イベント・モニターの特性	詳細
アプリケーションによるデータベースへの接続	CONNECTIONS	*このイベント・モニターの用途*	アプリケーションによるデータベースへの接続ごとに、メトリックおよびその他のモニター・エレメントをキャプチャーします。
		収集されるデータ	すべてアプリケーション・レベルのカウンターです。例えば、アプリケーションがデータベースに接続した時刻、アプリケーションがデータベースから切断した時刻、アプリケーションが関与したロック・エスカレーションの数などがあります。
		イベント・データが生成されるとき	接続の終了時[2]
データベースの非アクティブ化	DATABASE	*このイベント・モニターの用途*	活動化以降のデータベース全体についての情報を示すメトリックおよびその他のモニター・エレメントをキャプチャーします。
		収集されるデータ	すべてのデータベース・レベルのカウンター。例えば、活動化以降にデータベースに確立された接続数、ロック待機した時間、挿入されたデータの行数などがあります。
		イベント・データが生成されるとき	データベースの非アクティブ化時[2]
	BUFFERPOOLS TABLESPACES	*このイベント・モニターの用途*	バッファー・プールおよび表スペースに関するメトリックをキャプチャーします。
		収集されるデータ	バッファー・プール、プリフェッチャー、ページ・クリーナー、および個々のバッファー・プールの直接 I/O のカウンター。
		イベント・データが生成されるとき	データベースの非アクティブ化時[2]
	TABLES	*このイベント・モニターの用途*	データベースの活動化以降に変更された表に関するメトリックをキャプチャーします。
		収集されるデータ	表レベルのカウンター。読み取られた行数、書き込まれた行数、データ、LOB、または索引オブジェクトに使用されているディスク・ページ数などがあります。
		イベント・データが生成されるとき	データベースの非アクティブ化時[2]
ワークロード管理オブジェクトについての統計およびメトリック	STATISTICS	*このイベント・モニターの用途*	データベース内のワークロード管理オブジェクト (サービス・スーパークラス、ワークロードなど) についての処理メトリックをキャプチャーします。例えば、統計イベント・モニターを使用して、特定のワークロードの CPU 使用率の時間変化を確認できます。
		収集されるデータ	システム上の個々のサービス・クラス、ワークロード、または作業クラス中で実行されたアクティビティーから統計が計算されます。
		イベント・データが生成されるとき	統計は一定間隔で自動的に収集できます。この間隔は、**wlm_collect_int** データベース構成パラメーターで定義します。 WLM_COLLECT_STATS ストアード・プロシージャーを使用して手動でデータを収集することもできます。 **注:** どちらの収集メカニズムでも、統計モニター・エレメントの値は、収集が実行された後に 0 にリセットされます。

表 3. イベント・タイプ (続き)

モニターするイベントのタイプ	イベント・モニター名	イベント・モニターの特性	詳細
ワークロード管理しきい値の超過	THRESHOLD VIOLATIONS	*このイベント・モニターの用途*	設定した特定のしきい値が、データベース運用において、いつ超過されるのかを調べます。しきい値の設定対象は、CPU 時間からデータベース接続数、特定のステートメントの実行など、多岐にわたります。収集されたデータは、潜在的な問題がないか (TEMPORARY 表スペースの限度への接近など) モニターするためなどの、さまざまな目的に使用できます。
		収集されるデータ	しきい値違反情報。
		イベント・データが生成されるとき	しきい値違反の検出時。しきい値は、CREATE THRESHOLD ステートメントを使用して定義します。
データベースまたはデータベース・マネージャー構成への変更	CHANGE HISTORY	*このイベント・モニターの用途*	データベースおよびデータベース・マネージャーの構成変更、レジストリー設定変更、DDL ステートメントの実行、ならびにユーティリティーの実行をキャプチャーします。
		収集されるデータ	データベースおよびデータベース・マネージャーの構成パラメーターの変更、レジストリー変数の変更、DDL ステートメントの実行、特定の DB2 ユーティリティーおよびコマンドの実行、ならびに変更履歴イベント・モニターの開始。 **注:** 一般的に、変更履歴イベント・モニターが非アクティブである間、またはデータベースがオフラインの間に発生したイベントに関連する情報は、キャプチャーされません。ただし、レジストリー変数および構成パラメーターの変更は記録されます。
		イベント・データが生成されるとき[1]	モニターの開始時、パラメーターまたは変数の変更時、あるいはコマンド、DDL、またはユーティリティーの完了時。

注:

1. アクティビティー・イベント・モニターがアクティブなままデータベースを非アクティブ化すると、キューにバックログされているアクティビティー・レコードは廃棄されます。アクティビティー・イベント・モニター・レコードをすべて取得し、どれも破棄されないようにするには、データベースを非アクティブ化する前にまず、アクティビティー・イベント・モニターを非アクティブ化してください。アクティビティー・イベント・モニターを明示的に非アクティブ化する場合、イベント・モニターは、キュー内のすべてのバックログ・アクティビティー・レコードが処理されてから非アクティブ化されます。
2. データ収集が自動的に行われる規定のタイミングに加えて、 FLUSH EVENT MONITOR SQL ステートメントを使用してイベントを生成することもできます。この方式によって生成されたイベントは、フラッシュされたイベント・モニターに関連したすべてのモニター・タイプ (DEADLOCKS および DEADLOCKS WITH DETAILS を除く) に関する現行のデータベース・モニター値とともに書き込まれます。

表 4. 非推奨イベント・モニターのイベント・タイプ

モニターするイベントのタイプ	イベント・モニター名	イベント・モニターの特性	詳細
デッドロック	DEADLOCKS[2]	*このイベント・モニターの用途*	デッドロックがいつ発生するか、および関与しているアプリケーションを調べます。
		収集されるデータ	関係しているアプリケーション、および競合しているロック。
		イベント・データが生成されるとき	デッドロック検出時
	DEADLOCKS WITH DETAILS[2]	*このイベント・モニターの用途*	デッドロックがいつ発生するか、および関与しているアプリケーションを調べます。
		収集されるデータ	関係するアプリケーションについての広範囲の情報。関係するステートメント (およびステートメント・テキスト) の識別や保持されているロックなど。 DEADLOCKS イベント・モニターではなく DEADLOCKS WITH DETAILS イベント・モニターを使用すると追加の情報が収集されるため、デッドロックが発生したときにパフォーマンス・コストの負担になります。
		イベント・データが生成されるとき	デッドロック検出時
	DEADLOCKS WITH DETAILS HISTORY[2]	*このイベント・モニターの用途*	デッドロックがいつ発生するか、および関与しているアプリケーションを調べます。
		収集されるデータ	DEADLOCKS WITH DETAILS イベント・モニターで報告されるすべての情報。ならびにデータベース・パーティションのデッドロック・シナリオにかかわるロック (このデータベース・パーティションで保持されるロック) を所有する、各アプリケーションの現行作業単位のステートメント履歴。 DEADLOCKS WITH DETAILS HISTORY イベント・モニターを使用すると、ステートメント履歴を追跡することになるので、活動化したときに若干モニター・パフォーマンスの負担になります。
		イベント・データが生成されるとき	デッドロック検出時
	DEADLOCKS WITH DETAILS HISTORY VALUES[2]	*このイベント・モニターの用途*	
		収集されるデータ	デッドロックの詳細およびデッドロックの詳細履歴で報告されるすべての情報。ならびにステートメントの実行時にパラメーター・マーカーに提供されるあらゆる値。 DEADLOCKS WITH DETAILS HISTORY VALUES イベント・モニターを使用すると、さらにデータ値をコピーすることになるため、活動化したときにパフォーマンス・コストの大きな負担になります。
		イベント・データが生成されるとき	デッドロック検出時
作業単位 (トランザクション) の完了	TRANSACTIONS[3]	*このイベント・モニターの用途*	
		収集されるデータ	作業単位の作業開始/停止時刻、直前の作業単位時間、使用されている CPU、ロッキング、およびロギングのメトリック。 XA で実行している場合は、トランザクション・レコードは生成されません。
		イベント・データが生成されるとき	作業単位の完了時[1]

表 4. 非推奨イベント・モニターのイベント・タイプ (続き)

モニターするイベントのタイプ	イベント・モニター名	イベント・モニターの特性	詳細
注:			
1. データ収集が自動的に行われる規定のタイミングに加えて、 FLUSH EVENT MONITOR SQL ステートメントを使用してイベントを生成することもできます。この方式によって生成されたイベントは、フラッシュされたイベント・モニターに関連したすべてのモニター・タイプ (DEADLOCKS および DEADLOCKS WITH DETAILS を除く) に関する現行のデータベース・モニター値とともに書き込まれます。			
2. このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。			
3. このイベント・モニターは推奨されなくなりました。このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 トランザクション・イベントをモニターするには、 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用してください。			

注: 「デッドロックの詳細」イベント・モニターが、新規に作成されるデータベースごとに作成されます。このイベント・モニターは DB2DETAILDEADLOCK という名前で、データベースが活動化されると開始され、データベース・ディレクトリー内のファイルに書き込みます。このイベント・モニターに必要な追加のプロセッサー時間は、これをドロップすることによって回避することができます。 DB2DETAILDEADLOCK イベント・モニターは使用すべきではありません。このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

イベント・モニターの操作

一般的に、イベント・モニターを作成し使用して、特定のイベントが発生したときにシステムに関する情報をキャプチャーするプロセスは、すべてのイベント・モニター・タイプで類似しています。まずイベント・モニターを作成し、次にデータ収集を有効にして、最後に、収集されたデータにアクセスします。

このタスクについて

このトピックでは、イベント・モニターを操作する際に実行する一般的な手順を大まかに説明します。

手順

イベント・モニターを使用してイベント情報をキャプチャーするには、次のようにします。

1. イベント・モニターを作成します。 イベント・モニターを作成するには、適切なバージョンの CREATE EVENT MONITOR ステートメントを使用します。イベント・モニターを作成するときに、イベント・モニターが収集するデータをどのように記録するかを選択する必要があります。すべてのイベント・モニターは、その出力をリレーショナル表に書き込むことができますが、目的によっては、別の方法のほうが適している場合があります。

2. イベント・モニターを活動化します。 イベント・モニターを活動化するには、SET EVENT MONITOR STATE ステートメントを使用します。 例えば **capturestats** というイベント・モニターの場合は、次のコマンドを使用します。

   ```
   SET EVENT MONITOR capturestats STATE 1
   ```

   イベント・モニターによるデータ収集をオフにするには、次のステートメントを使用します。

   ```
   SET EVENT MONITOR capturestats STATE 0
   ```

   デフォルトでは、一部のイベント・モニターはデータベースが活動化されると自動的に活動化されますが、それ以外のイベント・モニターは手動で活動化しなければなりません。ただし、AUTOSTART オプションを使用して作成されたイベント・モニターは、次にデータベースが活動化されるまで自動的に活動化されません。最近作成されたイベント・モニターを強制的にアクティブ状態にするには、SET EVENT MONITOR STATE ステートメントを使用します。イベント・モニターが自動的に始動するかどうかを決める方法は、関連の CREATE EVENT MONITOR ステートメントについての参照情報をご覧ください。

3. データの収集を有効にします。 (LOCKING、ACTIVITIES、STATISTICS、UNIT OF WORK、および PACKAGE CACHE の各イベント・モニターの場合のみ) データの収集を有効にするには、イベント・モニターによって記録する特定のタイプのデータを収集するように、データベース・マネージャーを構成する必要があります。

   すべてのイベント・モニターでデータ収集を有効にする必要があるわけではありません。 TABLE イベント・モニターなど、有効にしなくともよいイベント・モニターでは、作成と活動化さえ行えば、データを収集するのには十分です。しきい値違反イベント・モニターも、データの収集を自動的に開始します。しかしこの場合、CREATE THRESHOLD ステートメントを使用して、データをキャプチャーする際のしきい値も定義する必要があります。

   データの収集を有効にする必要のあるイベント・モニターの場合、さまざまなオプションが使用可能です。扱っているイベント・モニターのタイプによっては、データベース全体にわたるデータ収集を有効にするようにデータベース構成パラメーターを設定することもできます。あるいは、特定のタイプのワークロード・オブジェクトに関する特定の種類のデータの収集を有効にするよう選択することもできます。 例えば、システム内のある作業単位が終了したときに、作業単位イベント・モニターに関する基本情報を収集するよう構成するためには、**mon_uow_data** パラメーターを BASE に設定することができます。あるいは、特定のワークロードに関してのみ作業単位情報をキャプチャーするためには、CREATE WORKLOAD ステートメントまたは ALTER WORKLOAD ステートメントの一部として COLLECT UNIT OF WORK DATA BASE 節を指定することができます。

4. アプリケーションまたは照会を実行します。 イベント・モニターを作成し、活動化してデータの収集を有効にした後、データを収集する対象のアプリケーションまたは照会を実行します。

5. オプション: イベント・モニターを非アクティブ化します。 データを収集する対象のアプリケーションまたは照会の実行後に、SET EVENT MONITOR STATE ステートメントを使用してイベント・モニターを非アクティブ化することができ

ます。 (ステップ 2 (42 ページ) を参照。) 次のステップに進む前に、イベント・モニターを非アクティブ化することは必須ではありませんが、イベント・モニターをアクティブのままにしておくと、見る必要のないデータのためにディスク・スペースが使用されることになります。

6. イベント・モニターによるデータ収集の結果を調べます。 イベント・モニターが作成した出力のタイプに応じて、収集したデータにアクセスするためのさまざまなオプションがあります。データが直接リレーショナル表に書き込まれた場合は、SQL を使用して、表の列に含まれるデータにアクセスできます。一方、イベント・モニターが未フォーマット・イベント (UE) 表に書き込んだ場合は、イベント・データを表示できるようにするために、**db2evmonfmt** のようなコマンドまたは EVMON_FORMAT_UE_TO_TABLES のようなプロシージャーを使用して、UE 表を後処理する必要があります。
7. オプション: 不要になったデータを、イベント・モニター表から除去します。 定期的に使用するイベント・モニターでは、表から不要なデータを除去するとよいでしょう。例えば、作業単位イベント・モニターを使用して、別のアプリケーションが使用しているシステム・リソースに関する日次会計レポートを生成している場合、レポートの生成後にイベント・モニター表からその日のデータを削除するとよいでしょう。

   ヒント: イベント・モニターの出力を定期的に整理する必要がある場合は、未フォーマット・イベント (UE) 表を使用してイベント・モニターの出力を記録することを検討してください。 DB2 バージョン 10.1 以降、データを通常表に転送した後で、UE 表を自動的に整理できるようになりました。

イベント・モニターの作成

CREATE EVENT MONITOR ステートメントのバリエーションを使用して、さまざまなタイプのイベント・モニターを作成します。このステートメントのオプションを使用すると、イベント・モニターで収集するデータのタイプ、およびイベント・モニターの出力の生成方法を指定できます。

この後の各セクションでは、各種の出力オプションと、これらのタイプの出力を生成するイベント・モニターの作成方法について説明していきます。

始める前に

イベント・モニターを作成する前に、イベント・モニターで生成できる出力のさまざまなオプションを理解しておくことが大切です。多くのイベント・モニターは、少なくとも 2 つの形式の出力を生成できます。また一部のイベント・モニターでは、最大で 4 つの形式から選べるものもあります。

手順

イベント・モニターを作成するには、以下のようにします。

1. どの種類のイベント・モニターが必要なのかを判別します。
2. イベント・モニターにどのようなタイプの出力をさせるかを決定します。データの書き込み先を、通常表、未フォーマット・イベント表、ファイル、またはパイプのどれにするのかということです。
3. CREATE EVENT MONITOR ステートメントを発行します。

4. オプション: 作成したイベント・モニターのタイプで活動化が必要な場合は、SET EVENT MONITOR STATE ステートメントを発行して活動化します。

イベント・モニターの出力オプション

イベント・モニターは、収集したデータをいくつかの方法でレポートできます。すべてのイベント・モニターは、収集したデータを表に書き込めます。一部のイベント・モニターは、未フォーマット・イベント (UE) 表に書き込むことが可能で、これはパフォーマンスの向上に役立ちます。ファイルまたは名前付きパイプに直接書き込めるイベント・モニターもあります。

イベント・モニターによって収集する情報の用途、およびイベント・モニターのタイプに応じて、イベント・モニターの収集した出力を、さまざまな方法で生成するように選択することができます。選択可能な出力タイプには以下があります。

通常表 DB2 バージョン 10.1 以降、すべてのイベント・モニターは、SQL を使用して直接照会できる通常の表に書き込めるようになりました。すべてのイベントにおいて、イベントに関して収集されたモニター・エレメントまたはメトリックは、それぞれ表の対応する列に書き込まれます。したがって、SELECT ステートメントを使用して出力を照会して、特定のモニター・エレメントの値を調べることが可能です。

表に書き込むイベント・モニターを作成するには、WRITE TO TABLE 節を CREATE EVENT MONITOR ステートメントに指定します。イベント・モニターに応じて、出力を格納するための表が 1 つ以上作成されます。各表には、単一の論理グループに属するモニター・エレメントが格納されます。各論理グループ用に生成される特定の表の詳細については、134 ページの『ターゲット表、コントロール表、およびイベント・モニター表の管理』を参照してください。

表は、選択した表スペース内に保管できます。ただし、 CREATE EVENT MONITOR ステートメントのターゲット表は、パーティション表以外の表でなければなりません。

注: 表に書き込むイベント・モニターには、2 つのタイプがあります。1 つ目のタイプは、バージョン 9.7 以降のリリースで作成されるイベント・モニターを含むタイプです。これには、作業単位、パッケージ・キャッシュ、ロック、変更履歴イベント・モニターなどがあります。DB2 バージョン 10.1 以降、これらのイベント・モニターのうち最初の 3 つは、 UE 表の代わりに通常の表にも出力を書き込めるようになりました。変更履歴イベント・モニターは、通常の表にのみ書き込みます。

2 つ目のタイプは、 DB2 バージョン 9.7 より前に実装されたイベント・モニターです。これには、他のすべてのイベント・モニターが含まれます。

概して、どちらのタイプのイベント・モニターも、作成した後は、ほとんど同じように機能します。つまり、生成された表のデータに SQL を使用して直接アクセスできます。ただし、2 つ目のカテゴリーに入る古い方のイベント・モニターの場合、イベント・モニターの作成時に指定できる追加のオプションがあります。また、2 つ目のカテゴリーに入るイベント・モニターのみ、ファイルおよび名前付きパイプにも書き込み可能です。

未フォーマット・イベント (UE) 表

UE 表は、DB2 バージョン 9.7 で、このリリースで追加された新しいイベント・モニター用に導入されました。UE 表はリレーショナル表ですが、限られた数の列しかありません。各イベントに関連付けられているデータのほとんどは、インライン・バイナリー (BLOB) オブジェクトを格納する列に書き込まれます。イベント・データをバイナリー・フォーマットで書き込むことによって、各レコードを表に書き込む時間が削減されます。このため、UE 表は、イベント・モニターのパフォーマンスが重要である場合に特に有用です。例えば、入出力や CPU の制約が厳しいシステムのケースなどが考えられます。

ただし、イベント・データがバイナリー・フォーマットで書き込まれるため、 SQL を使用して判読可能なデータを抽出することはできません。バイナリー・フォーマットで保管されているデータを抽出するには、 UE 表に対して後処理を実行する必要があります。UE 表を使用するもう 1 つの利点は、後処理時に自動的に UE 表データを除去できる点です。EVMON_FORMAT_UE_TO_TABLES プロシージャーには、正常にデータを抽出した場合に、そのデータを UE 表から削除するオプションがあります。

未フォーマット・イベント表に書き込むイベント・モニターを作成するには、 WRITE TO UNFORMATTED EVENT TABLE 節を CREATE EVENT MONITOR ステートメントに指定します。UE 表は、イベント・モニターごとに 1 つのみ作成されます。

ファイル

一部のイベント・モニターは、ファイル・システムによって保守されるファイルに、出力を直接送ることができます。イベント・モニターの出力にデータベース内での管理に伴う余分な処理時間の影響が及ばないようにしたい場合や、データベースがオフラインのときにデータを参照したい場合に、このタイプの出力が有用です。ファイルに書き込むイベント・モニターを作成するには、WRITE TO FILE 節を CREATE EVENT MONITOR ステートメントに指定します。

名前付きパイプ

アプリケーション・プロセスのイベント・データを、生成された直後に取得するには、名前付きパイプのイベント・モニターを使用します。このタイプのイベント・モニターは、名前付きパイプに直接出力を送るため、別のアプリケーションでただちにデータを使用することができます。イベント・データをリアルタイムで処理する必要がある場合に有用であると考えられます。

名前付きパイプに書き込むイベント・モニターを作成するには、WRITE TO PIPE 節を CREATE EVENT MONITOR ステートメントに指定します。

要件によっては、あるタイプのイベント・モニター出力が、他のタイプより適している場合があります。 46 ページの表 5 に、特定の出力タイプが特に有用となる状況について要約します。

表 5. イベント・モニターのさまざまな出力タイプの要約

出力タイプ	この出力タイプが有用なシナリオ
通常表	• モニター・データを後ほど参照する場合 • CPU、ログ・ファイル、またはディスク・ストレージの最大容量に接近していないシステム • SQL を使用したデータへの即時アクセスが望ましい場合
未フォーマット・イベント (UE) 表	• モニター・データを後ほど参照する場合 • イベント・モニターのパフォーマンスが優先されるシステム、あるいは CPU、ログ・ファイル、またはディスク使用量に制約があるシステム • データ後処理の追加ステップが問題にならない場合
ファイル	• モニター・データをデータベースの一部として管理することが望ましくない、または管理する必要がないシステム (ロギング、挿入、整合性維持のための追加の処理時間が不要になります)。 • モニター対象のデータベースの外にデータを保管する場合 • データを後ほどオフラインのときに参照する場合
パイプ	• イベント・データをただちに処理するアプリケーションにストリーミングする場合 • イベント・データに後ほどアクセスする必要がない場合

すべてのイベント・モニターがすべての出力タイプをサポートしているわけではありません。例えば、 UE 表を生成できるのは、作業単位、パッケージ・キャッシュ、およびロックのイベント・モニターのみです。 表 6 に、さまざまなタイプのイベント・モニターでどの出力オプションを使用できるのかを示します。

表 6. イベント・モニターの出力オプション

イベント・モニター・タイプ	通常の表	未フォーマット・イベント表	ファイル	名前付きパイプ
アクティビティー	はい		はい	はい
バッファー・プール	はい		はい	はい
変更履歴	はい			
接続	はい		はい	はい
データベース	はい		はい	はい
デッドロック*(すべてのバリエーション)	はい		はい	はい
ロッキング	はい	はい		
パッケージ・キャッシュ	はい	はい		
ステートメント	はい		はい	はい
統計	はい		はい	はい
表スペース	はい		はい	はい

表 6. イベント・モニターの出力オプション (続き)

イベント・モニター・タイプ	通常の表	未フォーマット・イベント表	ファイル	名前付きパイプ
表	はい		はい	はい
しきい値違反	はい		はい	はい
トランザクション*	はい		はい	はい
作業単位	はい	はい		
* 推奨されないイベント・モニターです。				

表に書き込むイベント・モニター

DB2 バージョン 10.1 以降、すべてのイベント・モニターは、SQL を使用して直接照会できる通常の表に出力を書き込めるようになりました。

また、DB2 バージョン 10.1 以降、プロシージャー EVMON_UPGRADE_TABLES を使用して、前のリリースのイベント・モニターによって生成された表をアップグレードすることができます。この機能により、 DB2 製品をアップグレードする際にイベント・モニター・データを保持することが、より簡単になります。

表に書き込むイベント・モニターの作成:

イベント・モニターを作成するには、CREATE EVENT MONITOR STATEMENT を使用します。このステートメントは、モニターしようとするイベントのタイプに応じて、さまざまな形で使用されます。

始める前に

- 表イベント・モニターを作成するには、SQLADM 権限または DBADM 権限が必要です。
- CREATE EVENT MONITOR ステートメントのターゲット表 (つまり、イベント・モニターがその出力を書き込む表) は、非パーティション表でなければなりません。

このタスクについて

表イベント・モニターの各種オプションは、CREATE EVENT MONITOR ステートメントで設定されます。表書き込みイベント・モニター用の CREATE EVENT MONITOR SQL ステートメントの生成をより簡単に行うために、 **db2evtbl** コマンドを使用することができます。イベント・モニターの名前および目的のイベント・タイプ (1 つ以上) を指定するだけで、 CREATE EVENT MONITOR ステートメントが生成され、すべてのターゲット表のリストが完成します。次に、生成されたステートメントをコピーして変更を加え、ステートメントをコマンド行プロセッサーから実行することができます。

手順

出力を通常表に書き込むイベント・モニターを作成するには、以下のステップを実行します。

1. WRITE TO TABLE 節を使用した CREATE EVENT MONITOR ステートメントを作成して、イベント・モニター・データが表 (または表集合) に収集されることを指定します。

   ```
   CREATE EVENT MONITOR evmon-name FOR eventtype
      WRITE TO TABLE
   ```

   evmon-name はイベント・モニターの名前、eventtype は以下の値のいずれかです。

   - ACTIVITIES
   - BUFFERPOOLS
   - CHANGE HISTORY
   - CONNECTIONS
   - DATABASE
   - DEADLOCKS
   - LOCKING
   - PACKAGE CACHE
   - STATEMENTS
   - STATISTICS
   - TABLE
   - TABLESPACE
   - THRESHOLD VIOLATIONS
   - TRANSACTIONS
   - UNIT OF WORK

   例えば、myevmon という作業単位イベント・モニターを作成するには、次のようなステートメントを使用します。

   ```
   CREATE EVENT MONITOR myevmon FOR UNIT OF WORK
      WRITE TO TABLE
   ```

   前述のステートメントでは、収集したモニター・エレメントの論理グループ、対応する出力表の名前、およびその表のターゲット表スペースに対してデフォルトを使用する作業単位イベント・モニターが作成されます。これらのデフォルトについて詳しくは、該当する CREATE EVENT MONITOR ステートメントの資料を参照してください。

2. オプション: データ収集の対象とする論理グループを指定します。 デフォルトでは、そのイベント・モニター・タイプのすべての論理データ・グループに関するイベント・データが収集されます。(詳しくは、 134 ページの『ターゲット表、コントロール表、およびイベント・モニター表の管理』を参照してください。) 選択した論理グループのデータのみを収集しようとする場合は、CREATE EVENT MONITOR ステートメントの中で、その論理グループの名前を指定することができます。例えば、ロッキング・イベント・モニターで、LOCK 論理グループと PARTICIPANT 論理グループに関連した情報のみを収集するとします。これらの論理グループのみを含めるためには、次のようなステートメントを使用することができます。

   ```
   CREATE EVENT MONITOR mylocks FOR LOCKING
      WRITE TO TABLE
      LOCK, PARTICIPANTS
   ```

3. オプション: 出力表に使用する表名を指定します。 他の表名を指定しなければ、モニター・エレメントの各論理グループに対する表に、デフォルトの名前が使用されます。デフォルトの名前は、論理グループ名をイベント・モニターの名前と

連結することによって導出されます。例えば、前述のステップでステートメントにより作成されたロッキング・イベント・モニターの場合、生成される表の非修飾名は LOCK_MYLOCKS と PARTICIPANTS_MYLOCKS です。デフォルトの名前をオーバーライドするには、次のようにして、論理グループを指定するときに使用する表名を含めます。

```
CREATE EVENT MONITOR mylocks FOR LOCKING
   WRITE TO TABLE
   LOCK(TABLE LOCKDATA), PARTICIPANTS(TABLE PARTICIP)
```

前述の例では、LOCK 論理グループと PARTICIPANTS 論理グループの表には、LOCKDATA_MYLOCKS と PARTICIP_MYLOCKS という名前が使用されます。

また次のように、使用する表スペースの名前を含めることによって、それぞれの表に使用される表スペースをオーバーライドすることもできます。

```
CREATE EVENT MONITOR mylocks FOR LOCKING
  WRITE TO TABLE
  LOCK(TABLE LOCKDATA IN EVMONSPACE), PARTICIPANTS(TABLE PARTICIP IN EVMONSPACE)
```

前述の例では、両方の出力表に EVMONSPACE 表スペースが使用されます。

追加のオプション

さまざまなイベント・モニターにより、さまざまな構成オプションが提供されています。特定のタイプのイベント・モニターで使用可能なオプションについて詳しくは、使用するイベント・モニターのタイプに関する CREATE EVENT MONITOR ステートメントの資料を参照してください。次に示す例では、各種のイベント・モニターで選択できる構成オプションの一部を紹介しています。

1 つのイベント・モニターによる複数のイベント・タイプのキャプチャー

一部のタイプ [2] のイベント・モニターでは、1 つのイベント・モニターでさまざまなタイプのイベントをキャプチャーできます。このイベント・モニターで複数のタイプのイベントをキャプチャーしようとする場合は、eventtype に追加の値をコンマで区切って指定します。例えば、1 つのイベント・モニターの中で、バッファー・プールと表スペース・モニタリングを組み合わせることができます。

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                    WRITE TO TABLE
```

このイベント・モニターは、BUFFERPOOL イベント・タイプと TABLESPACE イベント・タイプについてモニターします。上記のステートメントがユーザー dbadmin によって発行されたとすると、ターゲット表の派生名および表スペースは以下のようになります。

- DBADMIN.BUFFERPOOL_MYEVMON
- DBADMIN.TABLESPACE_MYEVMON
- DBADMIN.CONTROL_,MYEVMON

イベント・モニター出力バッファーのサイズ調整

一部のタイプ[2] のイベント・モニターでは、表イベント・モニター・バッフ

2. BUFFERPOOLS、CONNECTIONS、DATABASE、DEADLOCKS、STATEMENTS、TABLES、および TABLESPACES のイベント・モニターは、このオプションをサポートしています。

ァーのサイズは、BUFFERSIZE 値を調整することにより、(4K ページ単位で) 変更できます。例えば、次のステートメントの場合、

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                    WRITE TO TABLE BUFFERSIZE 8
```

8 は、2 つのイベント表バッファーを合わせた容量です (4K ページ単位)。それぞれのバッファーが 16K で、最大 32K までバッファー・スペースが追加されます。

各バッファーのデフォルト・サイズは 4 ページです (16K バッファーが 2 個割り振られます)。最小サイズは 1 ページです。バッファーはモニター・ヒープから割り振られるので、バッファーの最大サイズはこのヒープのサイズによって制限されます。パフォーマンス上の理由で、アクティブ率の高いイベント・モニターには、アクティブ率が比較的低いものよりも大きなバッファーが必要です。

イベント・モニターの出力をブロック化するか非ブロック化するかの制御

一部のイベント・モニター[2] では、イベント・モニターの出力バッファーが満杯になったときに、その後の処理をどのようにするか制御できます。ブロック化イベント・モニターの場合、イベントを生成する各エージェントは、イベント・バッファーが満杯であれば、それが表に書き込まれるまで待機します。これは、データベースのパフォーマンスを低下させることがあります。なぜなら、バッファーがクリアされるまで、延期されたエージェントおよび従属エージェントが実行できなくなるからです。イベント・データが脱落しないようにするには、BLOCKED 節を使用します。

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                    WRITE TO TABLE BUFFERSIZE 8 BLOCKED
```

すべての単一イベント・レコードを収集することよりも、データベース・パフォーマンスのほうが重要な場合には、非ブロック化イベント・モニターを使用してください。その場合、イベントを生成する各エージェントは、イベント・バッファーが満杯のときに、それが表に書き込まれるのを待機しません。結果として、非ブロック化イベント・モニターは、アクティブ率の高いシステムではデータの脱落という影響を受けます。イベント・モニターによって生じる処理時間の増加を最小限に抑えるには、NONBLOCKED 節を使用します。

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                    WRITE TO TABLE BUFFERSIZE 8 NONBLOCKED
```

注: 廃棄されたイベントに関する情報がイベント・モニターのコントロール表に書き込まれる方法についての追加情報は、 134 ページの『ターゲット表、コントロール表、およびイベント・モニター表の管理』および 156 ページの『表書き込みイベント・モニターおよびファイル・イベント・モニターのバッファー方式』を参照してください。

どのデータに関するどのモニター・エレメントを収集するかの制御

どのモニター・エレメントに関するデータを収集しますか。少数のモニター・エレメントにしか興味がない場合、一部のイベント・モニター[2] では、CREATE EVENT MONITOR ステートメントにエレメント名を指定することによって、どのモニター・エレメントを収集するかを指定することができます。

```
CREATE EVENT MONITOR myevmon FOR DATABASE, BUFFERPOOLS, TABLESPACES
             WRITE TO TABLE DB, DBMEMUSE,
             BUFFERPOOL (EXCLUDES(db_path, files_closed)),
             TABLESPACE (INCLUDES
               (tablespace_name, direct_reads, direct_writes))
             BUFFERSIZE 8 NONBLOCKED
```

DB 論理データ・グループおよび DBMEMUSE 論理データ・グループのすべてのモニター・エレメントがキャプチャーされます (これがデフォルトの動作です)。 BUFFERPOOL の場合は、**db_path** および **files_closed** 以外のすべてのモニター・エレメントがキャプチャーされます。最後に、TABLESPACE の場合は、モニター・エレメント **tablespace_name**、**direct_reads** および **direct_writes** だけがキャプチャーされます。

イベント・モニターを非アクティブ化する際の、使用する表スペースに基づくしきい値の設定

すべてのイベント・モニターで、表スペースがどの程度満杯になれば、イベント・モニターが自動的に非アクティブ化するのかを指定するオプションが提供されています。

```
CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                     PCTDEACTIVATE 90
```

表スペースが 90% の容量に達すれば、myevmon イベント・モニターが自動的にシャットオフします。 DMS 表スペースには PCTDEACTIVATE 節のみを使用できます。ターゲット表スペースで自動サイズ変更が有効な場合は、PCTDEACTIVATE 文節を 100 に設定してください。

次のタスク

デフォルトでは、バージョン 9.7 以降に導入されたイベント・モニターは、AUTOSTART イベント・モニターとして作成されます。データベースが次に活動化されるとき、およびその後データベースが活動化されるときに、これらのイベント・モニターが自動的に活動化されます。データベースが次に活動化される前にイベント・モニターをただちに活動化しようとするときは、SET EVENT MONITOR STATE ステートメントを使用してイベント・モニターを手動で開始します。ロッキング、作業単位、およびパッケージ・キャッシュの各イベント・モニターだけでなく、データ収集も有効化する必要があります。

イベント・モニターの論理データ・グループおよびモニター・エレメント:

一緒に調べると有用であることが多いモニター・エレメントは、*論理データ・グループ* にグループ化されます。

すべてのイベント・モニターは、何らかの形で論理データ・グループを使用します。一部のイベント・モニター・タイプでは、情報を記録する論理データ・グループを指定することによって、収集する情報を指定できます。論理データ・グループは、イベント・モニターが生成する出力の、データをグループ化するときにも使用されます。例えば、表に書き込むイベント・モニターは、一般的にはモニター・エレメントの論理データ・グループごとに 1 つの表を作成します。

次の表は、論理データ・グループと、イベント・モニターによって戻されるモニター・エレメントの一覧表です。

- 53 ページの『changesummary 論理データ・グループ』

- 54 ページの『dbdbmcfg 論理データ・グループ』
- 54 ページの『ddlstmtexec 論理データ・グループ』
- 54 ページの『dllock 論理データ・グループ』
- 55 ページの『event_activity 論理データ・グループ』
- 58 ページの『event_activitymetrics 論理データ・グループ』
- 62 ページの『event_activitystmt 論理データ・グループ』
- 63 ページの『event_activityvals 論理データ・グループ』
- 63 ページの『event_bufferpool 論理データ・グループ』
- 65 ページの『event_conn 論理データ・グループ』
- 68 ページの『event_connheader 論理データ・グループ』
- 69 ページの『event_connmemuse 論理データ・グループ』
- 69 ページの『event_data_value 論理データ・グループ』
- 70 ページの『event_db 論理データ・グループ』
- 74 ページの『event_dbheader 論理データ・グループ』
- 75 ページの『event_dbmemuse 論理データ・グループ』
- 75 ページの『event_deadlock 論理データ・グループ』
- 75 ページの『event_detailed_dlconn 論理データ・グループ』
- 77 ページの『event_dlconn 論理データ・グループ』
- 78 ページの『event_histogrambin 論理データ・グループ』
- 78 ページの『event_log_header 論理データ・グループ』
- 78 ページの『event_overflow 論理データ・グループ』
- 79 ページの『event_qstats 論理データ・グループ』
- 79 ページの『event_scmetrics 論理データ・グループ』
- 89 ページの『event_scstats 論理データ・グループ』
- 91 ページの『event_start 論理データ・グループ』
- 91 ページの『event_stmt 論理データ・グループ』
- 93 ページの『event_stmt_history 論理データ・グループ』
- 93 ページの『event_subsection 論理データ・グループ』
- 94 ページの『event_table 論理データ・グループ』
- 95 ページの『event_tablespace 論理データ・グループ』
- 96 ページの『event_thresholdviolations 論理データ・グループ』
- 97 ページの『event_wcstats 論理データ・グループ』
- 98 ページの『event_wlmetrics 論理データ・グループ』
- 108 ページの『event_wlstats 論理データ・グループ』
- 110 ページの『event_xact 論理データ・グループ』
- 111 ページの『evmonstart 論理データ・グループ』
- 111 ページの『lock 論理データ・グループ』
- 113 ページの『lock_participants 論理データ・グループ』
- 112 ページの『lock_participant_activities 論理データ・グループ』
- 111 ページの『lock_activity_values 論理データ・グループ』

- 115 ページの『pkgcache 論理データ・グループ』
- 116 ページの『pkgcache_metrics 論理データ・グループ』
- 121 ページの『pkgcache_stmt_args 論理データ・グループ』
- 122 ページの『regvar 論理データ・グループ』
- 122 ページの『sqlca 論理データ・グループ』
- 122 ページの『txncompletion 論理データ・グループ』
- 123 ページの『uow 論理データ・グループ』
- 125 ページの『uow_metrics 論理データ・グループ』
- 133 ページの『uow_package_list 論理データ・グループ』
- 124 ページの『uow_executable_list 論理データ・グループ』
- 133 ページの『utillocation 論理データ・グループ』
- 133 ページの『utilphase 論理データ・グループ』
- 134 ページの『utilstart 論理データ・グループ』
- 134 ページの『utilstop 論理データ・グループ』

changesummary 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

939 ページの『coord_member - コーディネーター・メンバー : モニター・エレメント』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

1729 ページの『utility_type ユーティリティー・タイプ』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

853 ページの『application_handle - アプリケーション・ハンドル : モニター・エレメント』

1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

900 ページの『client_port_number - クライアント・ポート番号 : モニター・エレメント』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

896 ページの『client_hostname - クライアント・ホスト名 : モニター・エレメント』

903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』

893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』

902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』

894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』

869 ページの『backup_timestamp - バックアップ・タイム・スタンプ』

dbdbmcfg 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

888 ページの『cfg_name - 構成名』

890 ページの『cfg_value - 構成値』

890 ページの『cfg_value_flags - 構成値フラグ』

889 ページの『cfg_old_value - 古い構成値』

889 ページの『cfg_old_value_flags - 古い構成値のフラグ』

888 ページの『cfg_collection_type - 構成収集タイプ』

980 ページの『deferred - 据え置き』

ddlstmtexec 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1061 ページの『global_transaction_id - グローバル・トランザクション ID のモニター・エレメント』

1126 ページの『local_transaction_id - ローカル・トランザクション ID のモニター・エレメント』

1474 ページの『savepoint_id - セーブポイント ID』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

975 ページの『ddl_classification - DDL 種別』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

dllock 論理データ・グループ

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1129 ページの『lock_count ロック・カウント : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1144 ページの『lock_object_name ロック対象名』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

1148 ページの『lock_status - ロック状況 : モニター・エレメント』

1208 ページの『node_number ノード番号』

1554 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

注: この論理データ・グループの基礎となる実装は、スナップショット・モニターの LOCK 論理データ・グループです。ファイルおよびパイプ出力オプションで使用される自己記述型ストリームで、この論理グループの出力を参照すると、 LOCK グループを使用して出力が生成されていることを確認できます。

event_activity 論理データ・グループ

812 ページの『act_exec_time アクティビティー実行時間 : モニター・エレメント』

817 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

820 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

821 ページの『activity_type アクティビティー・タイプ : モニター・エレメント』

823 ページの『address - 接続の開始元となった IP アドレス』

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

855 ページの『arm_correlator アプリケーション応答測定相関関係子 : モニター・エレメント』

940 ページの『coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント』

971 ページの『db_work_action_set_id データベース作業アクション・セット ID : モニター・エレメント』

972 ページの『db_work_class_id データベース作業クラス ID : モニター・エレメント』

982 ページの『details_xml - 詳細 XML : モニター・エレメント』

1110 ページの『intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント』

1220 ページの『num_remaps 再マップ数 : モニター・エレメント』

1256 ページの『parent_activity_id 親アクティビティー ID : モニター・エレメント』

1256 ページの『parent_uow_id 親作業単位 ID : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』

1474 ページの『sc_work_action_set_id サービス・クラス作業アクション・セット ID : モニター・エレメント』

1475 ページの『sc_work_class_id サービス・クラス作業クラス ID : モニター・エレメント』

1476 ページの『section_actuals - セクション実行時統計 : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1510 ページの『sqlca SQL 連絡域 (SQLCA)』

1599 ページの『time_completed 完了時刻 : モニター・エレメント』

1599 ページの『time_created 作成時刻 : モニター・エレメント』

1600 ページの『time_started 開始時刻 : モニター・エレメント』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1740 ページの『workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1423 ページの『prep_time 準備時間 : モニター・エレメント』

1432 ページの『query_card_estimate 照会行数の見積もり』

1433 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

1463 ページの『rows_fetched フェッチ行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

1553 ページの『system_cpu_time システム CPU 時間 : モニター・エレメント』

1721 ページの『user_cpu_time ユーザー CPU 時間 : モニター・エレメント』

1731 ページの『wl_work_action_set_id - ワークロード作業アクション・セット ID : モニター・エレメント』

1732 ページの『wl_work_class_id - ワークロード作業クラス ID : モニター・エレメント』

event_activitymetrics 論理データ・グループ

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1657 ページの『total_routine_non_sect_proc_time - 非セクション処理時間 : モニター・エレメント』

1658 ページの『total_routine_non_sect_time - 非セクション・ルーチンの実行時間 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

event_activitystmt 論理データ・グループ

817 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

820 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

908 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

1005 ページの『eff_stmt_text - 有効なステートメント・テキスト : モニター・エレメント』

1020 ページの『executable_id 実行可能 ID : モニター・エレメント』

1110 ページの『intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1220 ページの『num_routines - ルーチン数のモニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』

1460 ページの『routine_id - ルーチン ID : モニター・エレメント』

1477 ページの『section_env セクション環境 : モニター・エレメント』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1528 ページの『stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ : モニター・エレメント』

1530 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』

1530 ページの『stmt_isolation ステートメント分離』

1531 ページの『stmt_last_use_time - ステートメント最終使用時タイム・スタンプ : モニター・エレメント』

1531 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』

1532 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1535 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』

1537 ページの『stmt_source_id ステートメント・ソース ID』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

1546 ページの『stmtno - ステートメント番号のモニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

event_activityvals 論理データ・グループ

817 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

820 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1110 ページの『intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント』

1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』

1543 ページの『stmt_value_data 値データ』

1543 ページの『stmt_value_index 値索引』

1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

event_bufferpool 論理データ・グループ

874 ページの『bp_id バッファー・プール ID : モニター・エレメント』

874 ページの『bp_name - バッファー・プール名 : モニター・エレメント』

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1012 ページの『event_time イベント時刻』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1057 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1271 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1278 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

1279 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

1280 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

1281 ページの『pool_async_write_time - バッファー・プール非同期書き込み時間 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

871 ページの『block_ios - ブロック入出力要求数 : モニター・エレメント』

1254 ページの『pages_from_block_ios - ブロック入出力によって読み取られたページ数の合計 : モニター・エレメント』

1254 ページの『pages_from_vectored_ios - ベクトル化入出力によって読み取られたページ数の合計 : モニター・エレメント』

1277 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1286 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

1343 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

1730 ページの『vectored_ios - ベクトル化入出力要求数 : モニター・エレメント』

event_conn 論理データ・グループ

808 ページの『acc_curs_blk 受け入れられたブロック・カーソル要求』

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

848 ページの『appl_priority アプリケーション・エージェント優先順位』

849 ページの『appl_priority_type アプリケーション優先順位タイプ』

849 ページの『appl_section_inserts セクション挿入数 : モニター・エレメント』

850 ページの『appl_section_lookups - セクション検索』

866 ページの『authority_bitmap ユーザー許可レベル : モニター・エレメント』

867 ページの『authority_lvl ユーザー許可レベル : モニター・エレメント』

870 ページの『binds_precompiles 試行されたバインド/プリコンパイル』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

883 ページの『cat_cache_overflows カタログ・キャッシュ・オーバーフロー数』

907 ページの『commit_sql_stmts - 試行されたコミット・ステートメント』

976 ページの『ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1002 ページの『disconn_time データベース非アクティブ化タイム・スタンプ』

1004 ページの『dynamic_sql_stmts 試行された動的 SQL ステートメント』

1024 ページの『failed_sql_stmts 失敗したステートメント操作』

1078 ページの『hash_join_overflows ハッシュ結合のオーバーフロー』

1079 ページの『hash_join_small_overflows ハッシュ結合の短精度オーバーフロー』

1102 ページの『int_auto_rebinds 内部自動再バインド』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1105 ページの『int_deadlock_rollbacks デッドロックによる内部ロールバック』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1238 ページの『olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1109 ページの『int_rows_updated 更新された内部行数』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1425 ページの『priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数』

1426 ページの『priv_workspace_section_inserts 専用ワークスペース・セクション挿入』

1427 ページの『priv_workspace_section_lookups 専用ワークスペース・セクション検索』

1427 ページの『priv_workspace_size_top 専用ワークスペースの最大サイズ』

1444 ページの『rej_curs_blk リジェクトされたブロック・カーソル要求』

1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1471 ページの『rows_selected 選択行数』

1472 ページの『rows_written 書き込み行数』

1479 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1489 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

1490 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

1490 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

1491 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1523 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

1553 ページの『system_cpu_time システム CPU 時間 : モニター・エレメント』

1634 ページの『total_hash_joins ハッシュ結合の合計』

1635 ページの『total_hash_loops ハッシュ・ループの合計』

1644 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

1669 ページの『total_sec_cons 2 次接続』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1708 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

1721 ページの『user_cpu_time ユーザー CPU 時間 : モニター・エレメント』

1742 ページの『x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント』

1744 ページの『xquery_stmts - 試行された XQuery ステートメント』

850 ページの『appl_status アプリケーション状況 : モニター・エレメント』

884 ページの『cat_cache_size_top - カタログ・キャッシュの最高水準点 : モニター・エレメント』

940 ページの『coord_node コーディネーター・ノード』

1007 ページの『elapsed_exec_time ステートメント実行経過時間』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1462 ページの『rows_deleted - 削除行数 : モニター・エレメント』

1463 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

1471 ページの『rows_updated - 更新行数 : モニター・エレメント』

879 ページの『cat_cache_heap_full - カタログ・キャッシュ・ヒープ・フルのモニター・エレメント』

event_connheader 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

866 ページの『auth_id 許可 ID』

895 ページの『client_db_alias アプリケーションで使用するデータベース別名』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

904 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

924 ページの『conn_time データベース接続時刻 : モニター・エレメント』

941 ページの『corr_token DRDA 相関トークン』

1023 ページの『execution_id ユーザー・ログイン ID』

1208 ページの『node_number ノード番号』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1592 ページの『territory_code データベース・テリトリー・コード』

898 ページの『client_nname - クライアント名モニター・エレメント』

event_connmemuse 論理データ・グループ

1208 ページの『node_number ノード番号』

1287 ページの『pool_config_size メモリー・プールの構成済みサイズ』

1288 ページの『pool_cur_size メモリー・プールの現行サイズ』

1324 ページの『pool_id メモリー・プール ID』

1375 ページの『pool_secondary_id メモリー・プール 2 次 ID』

1390 ページの『pool_watermark メモリー・プール水準点』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

POOL_LIST_ID

POOL_MAX_SIZE

event_data_value 論理データ・グループ

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1258 ページの『participant_no デッドロック内の参加者』

1529 ページの『stmt_history_id ステートメント履歴 ID』

1543 ページの『stmt_value_data 値データ』

1543 ページの『stmt_value_index 値索引』

1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

event_db 論理データ・グループ

818 ページの『active_hash_joins - アクティブ・ハッシュ結合』

849 ページの『appl_section_inserts セクション挿入数 : モニター・エレメント』

850 ページの『appl_section_lookups - セクション検索』

856 ページの『async_runstats - 非同期 RUNSTATS 要求の合計数 : モニター・エレメント』

870 ページの『binds_precompiles 試行されたバインド/プリコンパイル』

872 ページの『blocks_pending_cleanup クリーンアップ保留中のロールアウト済みブロック : モニター・エレメント』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

883 ページの『cat_cache_overflows カタログ・キャッシュ・オーバーフロー数』

884 ページの『cat_cache_size_top - カタログ・キャッシュの最高水準点 : モニター・エレメント』

885 ページの『catalog_node カタログ・ノード番号』

885 ページの『catalog_node_name カタログ・ノード・ネットワーク名』

907 ページの『commit_sql_stmts - 試行されたコミット・ステートメント』

925 ページの『connections_top 同時接続の最大数』

966 ページの『db_heap_top 割り振られた最大データベース・ヒープ』

976 ページの『ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1002 ページの『disconn_time データベース非アクティブ化タイム・スタンプ』

1004 ページの『dynamic_sql_stmts 試行された動的 SQL ステートメント』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1024 ページの『failed_sql_stmts 失敗したステートメント操作』

1057 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

1078 ページの『hash_join_overflows ハッシュ結合のオーバーフロー』

1079 ページの『hash_join_small_overflows ハッシュ結合の短精度オーバーフロー』

1102 ページの『int_auto_rebinds 内部自動再バインド』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1170 ページの『log_held_by_dirty_pages ダーティー・ページによって占有されるログ・スペースの量』

1171 ページの『log_read_time ログ読み取り時間』

1172 ページの『log_reads 読み取られたログ・ページの数』

1172 ページの『log_to_redo_for_recovery リカバリーの場合に再実行されるログの量』

1173 ページの『log_write_time ログ書き込み時間』

1174 ページの『log_writes 書き込まれたログ・ページの数』

1216 ページの『num_log_read_io ログ読み取り数』

1217 ページの『num_log_write_io ログ書き込み数』

1221 ページの『num_threshold_violations しきい値違反の回数 : モニター・エレメント』

1238 ページの『olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

1268 ページの『pkg_cache_num_overflows パッケージ・キャッシュ・オーバーフロー』

1268 ページの『pkg_cache_size_top - パッケージ・キャッシュの最高水準点』

1271 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1425 ページの『priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数』

1426 ページの『priv_workspace_section_inserts 専用ワークスペース・セクション挿入』

1427 ページの『priv_workspace_section_lookups 専用ワークスペース・セクション検索』

1427 ページの『priv_workspace_size_top 専用ワークスペースの最大サイズ』

1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』

1462 ページの『rows_deleted - 削除行数 : モニター・エレメント』

1463 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1471 ページの『rows_selected 選択行数』

1471 ページの『rows_updated - 更新行数 : モニター・エレメント』

1475 ページの『sec_log_used_top 使用された最大 2 次ログ・スペース』

1479 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

1483 ページの『server_platform サーバーのオペレーティング・システム』

1489 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

1490 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

1490 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

1491 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1523 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

1524 ページの『stats_cache_size - 統計キャッシュのサイズ : モニター・エレメント』

1525 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間 : モニター・エレメント』

1526 ページの『stats_fabrications - 統計作成の合計数 : モニター・エレメント』

1551 ページの『sync_runstats - 同期 RUNSTATS アクティビティーの合計数 : モニター・エレメント』

1552 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント』

1601 ページの『tot_log_used_top 使用された最大合計ログ・スペース』

1617 ページの『total_cons データベース活動化以降の接続』

1634 ページの『total_hash_joins ハッシュ結合の合計』

1635 ページの『total_hash_loops ハッシュ・ループの合計』

1644 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1708 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

1742 ページの『x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント』

1744 ページの『xquery_stmts - 試行された XQuery ステートメント』

1007 ページの『elapsed_exec_time ステートメント実行経過時間』

1216 ページの『num_log_part_page_io 部分ログ・ページ書き込み数』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1286 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1506 ページの『sort_shrheap_top ソート共有ヒープの最高水準点』

879 ページの『cat_cache_heap_full - カタログ・キャッシュ・ヒープ・フルのモニター・エレメント』

LOG_FILE_ARCHIVE

LOG_FILE_NUM_CURR

LOG_FILE_NUM_FIRST

LOG_FILE_NUM_LAST

NUM_LOG_BUFF_FULL

NUM_LOG_DATA_IN_BUFF

event_dbheader 論理データ・グループ

924 ページの『conn_time データベース接続時刻 : モニター・エレメント』

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

event_dbmemuse 論理データ・グループ

1208 ページの『node_number ノード番号』

1287 ページの『pool_config_size メモリー・プールの構成済みサイズ』

1288 ページの『pool_cur_size メモリー・プールの現行サイズ』

1324 ページの『pool_id メモリー・プール ID』

1390 ページの『pool_watermark メモリー・プール水準点』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1375 ページの『pool_secondary_id メモリー・プール 2 次 ID』

POOL_MAX_SIZE

event_deadlock 論理データ・グループ

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1003 ページの『dl_conns - デッドロックに関係している接続 : モニター・エレメント』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1458 ページの『rolled_back_agent_id ロールバックされたエージェント』

1458 ページの『rolled_back_appl_id ロールバック・アプリケーション』

1459 ページの『rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント』

1459 ページの『rolled_back_sequence_no ロールバックされたシーケンス番号』

1523 ページの『start_time イベント開始時刻』

event_detailed_dlconn 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

845 ページの『appl_id_holding_lk ロックを保持しているアプリケーション ID』

872 ページの『blocking_cursor ブロック・カーソル』

926 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

958 ページの『cursor_name カーソル名』

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』
1140 ページの『lock_mode - ロック・モード : モニター・エレメント』
1142 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』
1144 ページの『lock_node ロック・ノード』
1144 ページの『lock_object_name ロック対象名』
1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』
1153 ページの『lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント』
1163 ページの『locks_held - ロック保持数 : モニター・エレメント』
1165 ページの『locks_in_list 報告されたロックの回数』
1248 ページの『package_name - パッケージ名 : モニター・エレメント』
1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』
1258 ページの『participant_no デッドロック内の参加者』
1258 ページの『participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者』
1477 ページの『section_number - セクション番号 : モニター・エレメント』
1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』
1481 ページの『sequence_no_holding_lk ロックを保持しているシーケンス番号』
1523 ページの『start_time イベント開始時刻』
1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』
1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』
1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』
1555 ページの『table_name - 表名 : モニター・エレメント』
1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』
1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』
1128 ページの『lock_attributes ロック属性 : モニター・エレメント』
1129 ページの『lock_count ロック・カウント : モニター・エレメント』
1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』
1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』
1143 ページの『lock_name ロック名 : モニター・エレメント』
1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』
1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

event_dlconn 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

845 ページの『appl_id_holding_lk ロックを保持しているアプリケーション ID』

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1129 ページの『lock_count ロック・カウント : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1142 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1144 ページの『lock_node ロック・ノード』

1144 ページの『lock_object_name ロック対象名』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

1153 ページの『lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1258 ページの『participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1481 ページの『sequence_no_holding_lk ロックを保持しているシーケンス番号』

1523 ページの『start_time イベント開始時刻』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

event_histogrambin 論理データ・グループ

870 ページの『bin_id ヒストグラム・ビン ID : モニター・エレメント』

873 ページの『bottom ヒストグラム・ビンの最下位 : モニター・エレメント』

1080 ページの『histogram_type ヒストグラム・タイプ : モニター・エレメント』

1223 ページの『number_in_bin ビン内の数 : モニター・エレメント』

1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1600 ページの『top ヒストグラム・ビンの最上位 : モニター・エレメント』

1736 ページの『work_action_set_id 作業アクション・セット ID : モニター・エレメント』

1737 ページの『work_class_id 作業クラス ID : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_log_header 論理データ・グループ

878 ページの『byte_order イベント・データのバイト・オーダー』

904 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

1011 ページの『event_monitor_name イベント・モニター名』

1219 ページの『num_nodes_in_db2_instance パーティション内のノード数』

1482 ページの『server_instance_name サーバー・インスタンス名』

1483 ページの『server_prdid - サーバー製品/バージョン ID』

1592 ページの『territory_code データベース・テリトリー・コード』

1730 ページの『version モニター・データのバージョン』

event_overflow 論理データ・グループ

942 ページの『count - イベント・モニター・オーバーフロー数』

1058 ページの『first_overflow_time 最初のイベント・オーバーフロー時刻』

1120 ページの『last_overflow_time 最後のイベント・オーバーフロー時刻』

1208 ページの『node_number ノード番号』

event_qstats 論理データ・グループ

1123 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

1435 ページの『queue_assignments_total キュー割り当ての合計 : モニター・エレメント』

1436 ページの『queue_size_top キュー・サイズの最上位 : モニター・エレメント』

1436 ページの『queue_time_total キュー時間の合計 : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1595 ページの『threshold_domain しきい値ドメイン : モニター・エレメント』

1596 ページの『threshold_name しきい値名 : モニター・エレメント』

1597 ページの『threshold_predicate しきい値述部 : モニター・エレメント』

1598 ページの『thresholdid しきい値 ID : モニター・エレメント』

1736 ページの『work_action_set_name 作業アクション・セット名 : モニター・エレメント』

1737 ページの『work_class_name 作業クラス名 : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_scmetrics 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

1260 ページの『partition_number パーティション番号』

1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』

831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』

1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』

897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』

1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』

1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』

1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』

1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』

1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』

1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』

860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』

862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』

864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』

986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』

1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』

1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』

841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』

1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』

1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』

1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』

1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』

1664 ページの『total_rqst_mapped_in - マッピングの際の要求の合計 : モニター・エレメント』

1664 ページの『total_rqst_mapped_out - マッピングの際に除外された要求の合計 : モニター・エレメント』

813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』

809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』

1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』

1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』

1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』

1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』

1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』

1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』

1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』

1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』

1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』

1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』

1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』

1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』

1642 ページの『total_loads - ロード合計 : モニター・エレメント』

1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』

1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』

1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』

1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』

1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』

1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』

1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』

1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』

1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』

1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』

1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』

1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』

1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』

1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』

1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』

1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』

1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』

1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』

1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』

1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』

1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』

1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』

1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』

1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』

1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』

1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』

1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』

1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』

1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』

1357 ページの『pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』

1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』

1366 ページの『pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求のモニター・エレメント』

1353 ページの『pool_queued_async_other_reqs - プリフェッチャーが処理したその他の要求モニター・エレメント』

1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』

1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』

1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』

1355 ページの『pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求のモニター・エレメント』

1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』

1364 ページの『pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求のモニター・エレメント』

1308 ページの『pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求のモニター・エレメント』

1310 ページの『pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求のモニター・エレメント』

1322 ページの『pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求のモニター・エレメント』

1314 ページの『pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』

1317 ページの『pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント』

1319 ページの『pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求のモニター・エレメント』

1313 ページの『pool_failed_async_other_reqs - 失敗した非プリフェッチの要求のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

1647 ページの『total_peds - partial early distinct の合計回数のモニター・エレメント』

1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』

1416 ページの『post_threshold_peds - partial early distinct しきい値のモニター・エレメント』

1645 ページの『total_peas - partial early aggregation の合計回数のモニター・エレメント』

1413 ページの『post_threshold_peas - partial early aggregation しきい値のモニター・エレメント』

1703 ページの『tq_sort_heap_requests - 表キュー・ソート・ヒープ要求のモニター・エレメント』

1700 ページの『tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否のモニター・エレメント』

1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』

1622 ページの『total_connect_requests - 接続要求またはユーザー切り替え要求のモニター・エレメント』

1623 ページの『total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント』

1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』

1619 ページの『total_connect_authentications - 実行された接続またはユーザー切り替え認証のモニター・エレメント』

1620 ページの『total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1422 ページの『prefetch_waits - プリフェッチャーの待機カウントのモニター・エレメント』

1288 ページの『pool_data_gbp_indep_pages_found
_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』

1326 ページの『pool_index_gbp_indep_pages
_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』

1393 ページの『pool_xda_gbp_indep_pages
_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』

905 ページの『comm_exit_wait_time - 通信バッファー出口待機時間のモニター・エレメント』

906 ページの『comm_exit_waits - 通信バッファー出口待機回数のモニター・エレメント』

FCM_TQ_RECV_WAITS_TOTAL

FCM_MESSAGE_RECV_WAITS_TOTAL

FCM_TQ_SEND_WAITS_TOTAL

FCM_MESSAGE_SEND_WAITS_TOTAL

FCM_SEND_WAITS_TOTAL

FCM_RECV_WAITS_TOTAL

1092 ページの『ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント』

1093 ページの『ida_sends_total - データ送信回数 : モニター・エレメント』

1090 ページの『ida_send_volume - 送信された合計データ量 : モニター・エレメント』

1087 ページの『ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント』

1088 ページの『ida_recvs_total - データ受信回数 : モニター・エレメント』

1085 ページの『ida_recv_volume - 受信した合計データ量 : モニター・エレメント』

event_scstats 論理データ・グループ

811 ページの『act_cpu_time_top - アクティビティーの CPU 時間の最上位 : モニター・エレメント』

814 ページの『act_remapped_in - 再マッピングするアクティビティー : モニター・エレメント』

814 ページの『act_remapped_out - 再マッピングの際に除外されるアクティビティー : モニター・エレメント』

814 ページの『act_rows_read_top - アクティビティーの読み取り行数の最上位 : モニター・エレメント』

816 ページの『act_throughput - アクティビティー・スループット : モニター・エレメント』

835 ページの『agg_temp_tablespace_top 集約 TEMPORARY 表スペースの最上位 : モニター・エレメント』

911 ページの『concurrent_act_top 並行アクティビティーの最上位 : モニター・エレメント』

913 ページの『concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント』

912 ページの『concurrent_connection_top 並行接続の最上位 : モニター・エレメント』

930 ページの『coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント』

930 ページの『coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント』

931 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

932 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

933 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

934 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

935 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

936 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

937 ページの『coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント』

942 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

951 ページの『cpu_utilization - CPU 使用率 : モニター・エレメント』

982 ページの『details_xml - 詳細 XML : モニター・エレメント』

1123 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

1203 ページの『metrics - メトリック : モニター・エレメント』.)

1455 ページの『request_exec_time_avg 要求の平均実行時間 : モニター・エレメント』

1470 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1592 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1710 ページの『uow_completed_total - 完了済みの合計作業単位 : モニター・エレメント』

1712 ページの『uow_lifetime_avg - 作業単位の平均存続期間 : モニター・エレメント』

1716 ページの『uow_throughput - 作業単位スループット : モニター・エレメント』

1716 ページの『uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_start 論理データ・グループ

1523 ページの『start_time イベント開始時刻』

event_stmt 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

834 ページの『agents_top 作成されたエージェントの数』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

872 ページの『blocking_cursor ブロック・カーソル』

926 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

958 ページの『cursor_name カーソル名』

1056 ページの『fetch_count 成功したフェッチの数』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1509 ページの『sql_req_id SQL ステートメントの要求 ID』

1510 ページの『sqlca SQL 連絡域 (SQLCA)』

1523 ページの『start_time イベント開始時刻』

1525 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間 : モニター・エレメント』

1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

1547 ページの『stop_time イベント停止時刻』

1552 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント』

1553 ページの『system_cpu_time システム CPU 時間 : モニター・エレメント』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1721 ページの『user_cpu_time ユーザー CPU 時間 : モニター・エレメント』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

event_stmt_history 論理データ・グループ

908 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1528 ページの『stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ : モニター・エレメント』

1529 ページの『stmt_history_id ステートメント履歴 ID』

1530 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』

1530 ページの『stmt_isolation ステートメント分離』

1531 ページの『stmt_last_use_time - ステートメント最終使用時タイム・スタンプ : モニター・エレメント』

1531 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』

1532 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1535 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』

1537 ページの『stmt_source_id ステートメント・ソース ID』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

event_subsection 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

1208 ページの『num_agents ステートメントで作動しているエージェントの数』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1519 ページの『ss_exec_time サブセクション実行経過時間』

1519 ページの『ss_node_number サブセクション・ノード番号』

1520 ページの『ss_number サブセクション番号 : モニター・エレメント』

1521 ページの『ss_sys_cpu_time サブセクションに使用されたシステム CPU 時間』

1521 ページの『ss_usr_cpu_time サブセクションに使用されたユーザー CPU 時間』

1698 ページの『tq_max_send_spills 表キュー・バッファー・オーバーフローの最大数』

1699 ページの『tq_rows_read 表キューから読み取られた行数』

1700 ページの『tq_rows_written 表キューに書き込まれた行数』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1509 ページの『sql_req_id SQL ステートメントの要求 ID』

event_table 論理データ・グループ

959 ページの『data_object_pages データ・オブジェクト・ページ数』

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

1012 ページの『event_time イベント時刻』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1098 ページの『index_object_pages 索引オブジェクト・ページ数』

1124 ページの『lob_object_pages LOB オブジェクト・ページ数』

1174 ページの『long_object_pages 長いオブジェクト・ページ数』

1246 ページの『overflow_accesses - オーバーフロー・レコードへのアクセス : モニター・エレメント』

1251 ページの『page_reorgs ページ再編成 : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1559 ページの『table_type - 表タイプ : モニター・エレメント』

1563 ページの『tablespace_id - 表スペース ID : モニター・エレメント』

1742 ページの『xda_object_pages XDA オブジェクト・ページ数』

event_tablespace 論理データ・グループ

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1012 ページの『event_time イベント時刻』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1057 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1271 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1277 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

1278 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

1279 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

1280 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

1281 ページの『pool_async_write_time - バッファー・プール非同期書き込み時間 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1343 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1286 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

TABLESPACE_FS_CACHING

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

event_thresholdviolations 論理データ・グループ

817 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

819 ページの『activity_collected 収集されたアクティビティー : モニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』

894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』

896 ページの『client_hostname - クライアント・ホスト名 : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

900 ページの『client_port_number - クライアント・ポート番号 : モニター・エレメント』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』

903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』

924 ページの『connection_start_time - 接続開始時刻 : モニター・エレメント』

940 ページの『coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント』

982 ページの『destination_service_class_id - 宛先サービス・クラス ID : モニター・エレメント』

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

1507 ページの『source_service_class_id ソース・サービス・クラス ID : モニター・エレメント』

1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』

1595 ページの『threshold_action しきい値アクション : モニター・エレメント』

1596 ページの『threshold_maxvalue しきい値最大値 : モニター・エレメント』

1597 ページの『threshold_predicate しきい値述部 : モニター・エレメント』

1598 ページの『threshold_queuesize しきい値キュー・サイズ : モニター・エレメント』

1598 ページの『thresholdid しきい値 ID : モニター・エレメント』

1599 ページの『time_of_violation 違反時刻 : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

event_wcstats 論理データ・グループ

811 ページの『act_cpu_time_top - アクティビティーの CPU 時間の最上位 : モニター・エレメント』

814 ページの『act_rows_read_top - アクティビティーの読み取り行数の最上位 : モニター・エレメント』

817 ページの『act_total アクティビティーの合計 : モニター・エレメント』

931 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

932 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

933 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

934 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

935 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

936 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

942 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

1123 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

1470 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1592 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

1736 ページの『work_action_set_id 作業アクション・セット ID : モニター・エレメント』

1736 ページの『work_action_set_name 作業アクション・セット名 : モニター・エレメント』

1737 ページの『work_class_id 作業クラス ID : モニター・エレメント』

1737 ページの『work_class_name 作業クラス名 : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_wlmetrics 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

1260 ページの『partition_number パーティション番号』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1739 ページの『workload_name ワークロード名 : モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』

831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』

1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』

897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』

1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』

1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』

1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』

1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』

1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』

1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』

860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』

862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』

864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』

986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』

1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』

1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』

841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』

1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』

1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』

1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』

1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』

813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』

809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』

1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』

1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』

1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』

1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』

1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』

1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』

1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』

1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』

1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』

1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』

1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』

1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』

1642 ページの『total_loads - ロード合計 : モニター・エレメント』

1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』

1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』

1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』

1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』

1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』

1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』

1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』

1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』

1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』

1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』

1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』

1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』

1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』

1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』

1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』

1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』

1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』

1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』

1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』

1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』

1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』

1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』

1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』

1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』

1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』

1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』

1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』

1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』

1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』

1357 ページの『pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』

1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』

1366 ページの『pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求のモニター・エレメント』

1353 ページの『pool_queued_async_other_reqs - プリフェッチャーが処理したその他の要求モニター・エレメント』

1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』

1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』

1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』

1355 ページの『pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求のモニター・エレメント』

1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』

1364 ページの『pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求のモニター・エレメント』

1308 ページの『pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求のモニター・エレメント』

1310 ページの『pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求のモニター・エレメント』

1322 ページの『pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求のモニター・エレメント』

1314 ページの『pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』

1317 ページの『pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント』

1319 ページの『pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求のモニター・エレメント』

1313 ページの『pool_failed_async_other_reqs - 失敗した非プリフェッチの要求のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

1647 ページの『total_peds - partial early distinct の合計回数のモニター・エレメント』

1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』

1416 ページの『post_threshold_peds - partial early distinct しきい値のモニター・エレメント』

1645 ページの『total_peas - partial early aggregation の合計回数のモニター・エレメント』

1413 ページの『post_threshold_peas - partial early aggregation しきい値のモニター・エレメント』

1703 ページの『tq_sort_heap_requests - 表キュー・ソート・ヒープ要求のモニター・エレメント』

1700 ページの『tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否のモニター・エレメント』

1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』

1622 ページの『total_connect_requests - 接続要求またはユーザー切り替え要求のモニター・エレメント』

1623 ページの『total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント』

1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』

1619 ページの『total_connect_authentications - 実行された接続またはユーザー切り替え認証のモニター・エレメント』

1620 ページの『total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1422 ページの『prefetch_waits - プリフェッチャーの待機カウントのモニター・エレメント』

1288 ページの『pool_data_gbp_indep_pages_found
_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』

1326 ページの『pool_index_gbp_indep_pages
_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』

1393 ページの『pool_xda_gbp_indep_pages
_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』

905 ページの『comm_exit_wait_time - 通信バッファー出口待機時間のモニター・エレメント』

906 ページの『comm_exit_waits - 通信バッファー出口待機回数のモニター・エレメント』

FCM_TQ_RECV_WAITS_TOTAL

FCM_MESSAGE_RECV_WAITS_TOTAL

FCM_TQ_SEND_WAITS_TOTAL

FCM_MESSAGE_SEND_WAITS_TOTAL

FCM_SEND_WAITS_TOTAL

FCM_RECV_WAITS_TOTAL

1092 ページの『ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント』

1093 ページの『ida_sends_total - データ送信回数 : モニター・エレメント』

1090 ページの『ida_send_volume - 送信された合計データ量 : モニター・エレメント』

1087 ページの『ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント』

1088 ページの『ida_recvs_total - データ受信回数 : モニター・エレメント』

1085 ページの『ida_recv_volume - 受信した合計データ量 : モニター・エレメント』

event_wlstats 論理データ・グループ

811 ページの『act_cpu_time_top - アクティビティーの CPU 時間の最上位 : モニター・エレメント』

814 ページの『act_rows_read_top - アクティビティーの読み取り行数の最上位 : モニター・エレメント』

816 ページの『act_throughput - アクティビティー・スループット : モニター・エレメント』

913 ページの『concurrent_wlo_act_top 並行 WLO アクティビティーの最上位 : モニター・エレメント』

913 ページの『concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント』

930 ページの『coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント』

930 ページの『coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント』

931 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

932 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

933 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

934 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

935 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

936 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

937 ページの『coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント』

942 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

951 ページの『cpu_utilization - CPU 使用率 : モニター・エレメント』

982 ページの『details_xml - 詳細 XML : モニター・エレメント』

1123 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

1159 ページの『lock_wait_time_top - ロック待機時間の最上位 : モニター・エレメント』

1203 ページの『metrics - メトリック : モニター・エレメント』

1470 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1592 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1710 ページの『uow_completed_total - 完了済みの合計作業単位 : モニター・エレメント』

1712 ページの『uow_lifetime_avg - 作業単位の平均存続期間 : モニター・エレメント』

1716 ページの『uow_throughput - 作業単位スループット : モニター・エレメント』

1716 ページの『uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント』

1735 ページの『wlo_completed_total 完了したワークロード・オカレンスの合計 : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1739 ページの『workload_name ワークロード名 : モニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

1159 ページの『lock_wait_time_global_top - 最長グローバル・ロック待機時間 : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_xact 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1164 ページの『locks_held_top - ロック保持最大数 : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1424 ページの『prev_uow_stop_time 直前の作業単位完了タイム・スタンプ』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1553 ページの『system_cpu_time システム CPU 時間 : モニター・エレメント』

1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

1713 ページの『uow_log_space_used - 使用されている作業単位ログ・スペース: モニター・エレメント』

1714 ページの『uow_start_time - 作業単位開始タイム・スタンプ : モニター・エレメント』

1715 ページの『uow_status 作業単位の状況』

1547 ページの『stop_time イベント停止時刻』

1721 ページの『user_cpu_time ユーザー CPU 時間 : モニター・エレメント』

1742 ページの『x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

evmonstart 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

965 ページの『db2start_time - データベース・マネージャー開始タイム・スタンプ』

965 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』

lock 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

1003 ページの『dl_conns - デッドロックに関係している接続 : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1459 ページの『rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント』

978 ページの『deadlock_type - デッドロック・タイプのモニター・エレメント』

lock_activity_values 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1543 ページの『stmt_value_index 値索引』

1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1543 ページの『stmt_value_data 値データ』

1010 ページの『event_id - イベント ID モニター・エレメント』

lock_participant_activities 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

821 ページの『activity_type アクティビティー・タイプ : モニター・エレメント』

926 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

1006 ページの『effective_isolation - 有効な分離 : モニター・エレメント』

1007 ページの『effective_query_degree - 有効な照会の度合い : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1097 ページの『incremental_bind - 追加バインドのモニター・エレメント』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_schema - パッケージ・スキーマ : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』

1447 ページの『reopt - REOPT バインド・オプションのモニター・エレメント』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1528 ページの『stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ : モニター・エレメント』

1530 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』

1531 ページの『stmt_last_use_time - ステートメント最終使用時タイム・スタンプ : モニター・エレメント』

1531 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』

1532 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1535 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』

1537 ページの『stmt_source_id ステートメント・ソース ID』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1542 ページの『stmt_unicode - ステートメントのユニコード・フラグのモニター・エレメント』

1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

lock_participants 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

827 ページの『agent_status DCS アプリケーション・エージェント』

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

853 ページの『application_handle - アプリケーション・ハンドル : モニター・エレメント』

866 ページの『auth_id 許可 ID』

893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』

894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』

902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』

903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』

936 ページの『coord_agent_tid - コーディネーター・エージェントのエンジン・ディスパッチ可能単位 ID のモニター・エレメント』

958 ページの『current_request - 現在の操作要求のモニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1129 ページの『lock_count ロック・カウント : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1142 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

LOCK_OBJECT_TYPE_ID

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

LOCK_RRIID

1148 ページの『lock_status - ロック状況 : モニター・エレメント』

1149 ページの『lock_timeout_val ロック・タイムアウト値 : モニター・エレメント』

1153 ページの『lock_wait_end_time - ロック待機終了タイム・スタンプ : モニター・エレメント』

1153 ページの『lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント』

1159 ページの『lock_wait_val - ロック待機値のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1232 ページの『object_requested - 要求されたオブジェクトのモニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1258 ページの『participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者』

1259 ページの『participant_type - 参加者タイプのモニター・エレメント』

1261 ページの『past_activities_wrapped - 過去のアクティビティー・リストの折り返しモニター・エレメント』

1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1554 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1598 ページの『thresholdid しきい値 ID : モニター・エレメント』

1596 ページの『threshold_name しきい値名 : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1739 ページの『workload_name ワークロード名 : モニター・エレメント』

828 ページの『agent_tid - エージェント・スレッド ID モニター・エレメント』

840 ページの『appl_action - アプリケーション・アクションのモニター・エレメント』

977 ページの『deadlock_member - デッドロック・メンバー : モニター・エレメント』

1435 ページの『queue_start_time - キュー開始タイム・スタンプのモニター・エレメント』

1437 ページの『queued_agents - キューに入れられているしきい値エージェントのモニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

1744 ページの『xid トランザクション ID』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

pkgcache 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

908 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

1006 ページの『effective_isolation - 有効な分離 : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1020 ページの『executable_id 実行可能 ID : モニター・エレメント』

1102 ページの『insert_timestamp - 挿入タイムスタンプ : モニター・エレメント』

1120 ページの『last_metrics_update - メトリック最終更新タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1210 ページの『num_coord_exec - コーディネーター・エージェントによる実行数 : モニター・エレメント』

1210 ページの『num_coord_exec_with_metrics - メトリック付きの、コーディネーター・エージェントによる実行数 : モニター・エレメント』

1212 ページの『num_exec_with_metrics メトリックが収集された実行数 : モニター・エレメント』

1211 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』

1220 ページの『num_routines - ルーチン数のモニター・エレメント』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_schema - パッケージ・スキーマ : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1423 ページの『prep_time 準備時間 : モニター・エレメント』

1433 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

1434 ページの『query_data_tag_list - 見積もりの照会データ・タグ・リストのモニター・エレメント』

1460 ページの『routine_id - ルーチン ID : モニター・エレメント』

1477 ページの『section_env セクション環境 : モニター・エレメント』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1479 ページの『section_type - セクション・タイプ標識 : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1541 ページの『stmt_type_id - ステートメント・タイプ ID : モニター・エレメント』

1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』

1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』

1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1546 ページの『stmtno - ステートメント番号のモニター・エレメント』

1176 ページの『max_coord_stmt_exec_time - コーディネーターの最大ステートメント実行時間のモニター・エレメント』

1179 ページの『max_coord_stmt_exec_timestamp - コーディネーターの最大ステートメント実行のタイム・スタンプのモニター・エレメント』

pkgcache_metrics 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』

860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』

862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』

864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』

986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』

941 ページの『coord_stmt_exec_time - コーディネーター・エージェントのステートメントの実行時間 : モニター・エレメント』

1657 ページの『total_routine_non_sect_proc_time - 非セクション処理時間 : モニター・エレメント』

1658 ページの『total_routine_non_sect_time - 非セクション・ルーチンの実行時間 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』

1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』

1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』

1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』

1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』

1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』

1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』

1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』

1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』

1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』

1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』

1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』

1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』

1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』

1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』

1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』

1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』

1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』

1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』

1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』

1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』

1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』

pkgcache_stmt_args 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1543 ページの『stmt_value_index 値索引』

1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

1543 ページの『stmt_value_data 値データ』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

regvar 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1443 ページの『regvar_name - レジストリー変数名』

1444 ページの『regvar_value - レジストリー変数の値』

1444 ページの『regvar_old_value - レジストリー変数の古い値』

1443 ページの『regvar_level - レジストリー変数のレベル』

1443 ページの『regvar_collection_type - レジストリー変数収集タイプ』

sqlca 論理データ・グループ

sqlcabc

sqlcaid

sqlcode

sqlerrd

sqlerrmc

sqlerrml

sqlerrp

sqlstate

sqlwarn

txncompletion 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1061 ページの『global_transaction_id - グローバル・トランザクション ID のモニター・エレメント』

1126 ページの『local_transaction_id - ローカル・トランザクション ID のモニター・エレメント』

1474 ページの『savepoint_id - セーブポイント ID』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

975 ページの『ddl_classification - DDL 種別』

1707 ページの『txn_completion_status - トランザクション完了状況』

uow 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

853 ページの『application_handle - アプリケーション・ハンドル : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』

894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』

896 ページの『client_hostname - クライアント・ホスト名 : モニター・エレメント』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

900 ページの『client_port_number - クライアント・ポート番号 : モニター・エレメント』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』

903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』

908 ページの『completion_status 完了状況 : モニター・エレメント』

924 ページの『conn_time データベース接続時刻 : モニター・エレメント』

939 ページの『coord_member - コーディネーター・メンバー : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1021 ページの『executable_list_size - 実行可能リスト・サイズのモニター・エレメント』

1061 ページの『global_transaction_id - グローバル・トランザクション ID のモニター・エレメント』

1110 ページの『intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント』

1126 ページの『local_transaction_id - ローカル・トランザクション ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

965 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1248 ページの『package_list_exceeded - パッケージ・リストの超過 : モニター・エレメント』

1248 ページの『package_list_size - パッケージ・リスト・サイズのモニター・エレメント』

service_class_id サービス・クラス ID

service_subclass_name サービス・サブクラス名

service_superclass_name サービス・スーパークラス名

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

start_time イベント開始時刻

stop_time イベント停止時刻

1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』

UOW_CLIENT_PLATFORM

UOW_CLIENT_PROTOCOL

uow_id 作業単位 ID

1713 ページの『uow_log_space_used - 使用されている作業単位ログ・スペース: モニター・エレメント』

workload_id ワークロード ID

workload_name - ワークロード名

workload_occurrence_id ワークロード・オカレンス ID

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

1022 ページの『executable_list_truncated - 実行可能リスト切り捨てのモニター・エレメント』

924 ページの『connection_start_time - 接続開始時刻 : モニター・エレメント』

uow_executable_list 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1020 ページの『executable_id 実行可能 ID : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1211 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

uow_id 作業単位 ID

uow_metrics 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』

831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』

1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』

897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』

1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』

1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』

1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』

1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』

1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』

1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』

860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』

862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』

864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』

986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』

1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』

1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』

841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』

1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』

1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』

1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』

1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』

813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』

809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』

1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』

1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』

1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』

1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』

1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』

1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』

1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』

1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』

1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』

1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』

1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』

1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』

1642 ページの『total_loads - ロード合計 : モニター・エレメント』

1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』

1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』

1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』

1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』

1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』

1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』

1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』

1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』

1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』

1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』

1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』

1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』

1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』

1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』

1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』

1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』

1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』

1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』

1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』

1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』

1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』

1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』

1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』

1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』

1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』

1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』

1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』

1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』

1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』

1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』

1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』

1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

uow_package_list 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1110 ページの『invocation_id - 呼び出し ID : モニター・エレメント』

1205 ページの『nesting_level - ネスト・レベル : モニター・エレメント』

1247 ページの『package_elapsed_time - パッケージ経過時間 : モニター・エレメント』

1247 ページの『package_id - パッケージ ID : モニター・エレメント』

1460 ページの『routine_id - ルーチン ID : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

utillocation 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

1729 ページの『utility_type ユーティリティー・タイプ』

983 ページの『device_type - 装置タイプ』

1127 ページの『location_type - ロケーション・タイプ・』

1127 ページの『location - ロケーション』

utilphase 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

1729 ページの『utility_type ユーティリティー・タイプ』

1726 ページの『utility_phase_type - ユーティリティー・フェーズ・タイプ』

1262 ページの『phase_start_event_id - フェーズ開始イベント ID』

1262 ページの『phase_start_event_timestamp - フェーズ開始イベントのタイム・スタンプ』

1237 ページの『objtype - オブジェクト・タイプ : モニター・エレメント』

1232 ページの『object_schema - オブジェクト・スキーマのモニター・エレメント』

1231 ページの『object_name - オブジェクト名モニター・エレメント』

1726 ページの『utility_phase_detail - ユーティリティー・フェーズ詳細』

utilstart 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

1729 ページの『utility_type ユーティリティー・タイプ』

1724 ページの『utility_operation_type - ユーティリティー操作のタイプ』

1724 ページの『utility_invoker_type - ユーティリティー呼び出し側タイプ』

1726 ページの『utility_priority ユーティリティー優先度』

1727 ページの『utility_start_type - ユーティリティー開始タイプ』

1237 ページの『objtype - オブジェクト・タイプ : モニター・エレメント』

1232 ページの『object_schema - オブジェクト・スキーマのモニター・エレメント』

1231 ページの『object_name - オブジェクト名モニター・エレメント』

1221 ページの『num_tbsps - 表スペース数のモニター・エレメント』

1584 ページの『tbsp_names - 表スペース名』

1723 ページの『utility_detail - ユーティリティー詳細』

utilstop 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

1729 ページの『utility_type ユーティリティー・タイプ』

1728 ページの『utility_stop_type - ユーティリティー停止タイプ』

1522 ページの『start_event_id - 開始イベント ID』

1522 ページの『start_event_timestamp - 開始イベント・タイム・スタンプ』

1510 ページの『sqlca SQL 連絡域 (SQLCA)』

ターゲット表、コントロール表、およびイベント・モニター表の管理:

SQL 表にイベント・レコードを保管するように、イベント・モニターを定義することができます。これを行うには、CREATE EVENT MONITOR ステートメントを WRITE TO TABLE 節を付けて使用します。

表書き込みイベント・モニターを作成すると、イベント・モニターは、データを戻す論理データ・グループのそれぞれについて、レコードを保管するためのターゲット表を作成します。それぞれの表において、列名はそれが表すモニター・エレメント名と一致しています。デフォルトでは、イベント・モニターはイベント・モニターの作成者のスキーマで表を作成し、その表に、対応する論理データ・グループ名とイベント・モニター名を連結した名前を付けます。

例えば、以下のステートメントについて考えてみましょう。これは、STATEMENTS イベントをキャプチャーするイベント・モニターを作成するものです。

```
CREATE EVENT MONITOR test FOR STATEMENTS WRITE TO TABLE
```

STATEMENTS イベント・タイプを使用するイベント・モニターは、event_connheader、event_stmt、および event_subsection 論理データ・グループからデータを収集します。個々のイベント・タイプに固有な論理データ・グループを表す各表に加えて、表書き込みイベント・モニターごとに 1 つのコントロール表が作成されます。ユーザー `riihi` によって作成されたイベント・モニター `test` の場合、データベース・マネージャーは以下の表を作成します。

- riihi.connheader_test
- riihi.stmt_test
- riihi.subsection_test
- riihi.control_test

最初の 3 つの表は、論理データ・グループ event_connheader、event_stmt、および event_subsection のそれぞれに対応しています。最後の表 riihi.control_test は、コントロール表です。コントロール表には、イベント・モニター・メタデータ、具体的には event_start、event_dbheader (conn_time モニター・エレメントのみ)、および event_overflow 論理データ・グループからのイベント・モニター・メタデータが含まれます。

モニター・エレメントがオーバーフローのグループに書き込まれるのは、*非ブロック化* イベント・モニターの場合のみです。非ブロック化イベント・モニターでは、イベントを生成するエージェントは、イベント・バッファーが満杯になった場合にバッファーが表に書き込まれるのを待機しません。代わりに、イベント・モニターは、自身がデータを書き込める速度よりも速くエージェントから来たモニター・データを廃棄します。この場合、イベント・モニターは、オーバーフローが発生したことを示す情報をコントロール表に記録します。この情報には、モニター・エレメント **`message`** が含まれます。オーバーフローが発生した場合には、このモニター・エレメントにテキスト `OVERFLOW:n` があります。*n* は、イベント・バッファーが満杯だったために廃棄されたイベント・レコードの数を表します。

表書き込みイベント・モニターはアクティブになると、各ターゲット表に対する IN 表ロックまたは IX 表ロックを獲得します。これによって、イベント・モニターがアクティブである間にターゲット表が変更されることを防ぎます。この表ロックは、イベント・モニターがアクティブである間は、すべての表において維持されます。ターゲット表に対する排他的アクセスが必要な場合 (例えば、ユーティリティ

ーを実行する場合) には、そのようなアクセスを試行する前に、まずイベント・モニターを非アクティブ化して表ロックを解放してください。

ターゲット表の各列名は、イベント・モニターのエレメント ID と一致しています。対応するターゲット表の列がないイベント・モニター・エレメントは無視されます。

表書き込みイベント・モニターのターゲット表 (未フォーマット・イベント (UE) 表を含む) は、手動で整理する必要があります。 非常にアクティブなシステムでは、イベント・モニターが大量のデータを記録するために、ディスク・スペースをすぐに使い尽くしてしまうことがあります。ファイルまたは名前付きパイプに書き込むイベント・モニターを定義する場合と異なり、表書き込みイベント・モニターは、特定の論理データ・グループまたはモニター・エレメントの情報だけを記録するように定義することができます。このフィーチャーを使用して、目的にかなうデータだけを収集すれば、イベント・モニターによって生成されるデータ量を削減することができます。例えば、次のステートメントは、 event_conn 論理データ・グループだけから接続イベントをキャプチャーし、**lock_waits** モニター・エレメントのみを含むようイベント・モニターを定義しています。

```
CREATE EVENT MONITOR conn_monitor FOR CONNECTIONS WRITE TO TABLE
                    CONN(INCLUDES(lock_waits))
```

イベント・モニターのターゲット表を、デフォルトの表スペース内にデフォルトの表名を使用してデフォルトのスキーマで作成することが望ましくない場合があります。大量のモニター・データが予想される場合は、ターゲット表を専用の表スペースに配置する必要があるでしょう。スキーマ、表、および表スペースの名前は CREATE EVENT MONITOR ステートメントで指定できます。そのスキーマ名と表名から、各表の派生名が形成されます。表スペース名は、表名の後に、オプションの IN 節を付けて追加することができます。 DB2 データベース・マネージャーが自動的に作成するターゲット表とは異なり、表スペースをイベント・モニター定義内に指定する場合、その表スペースは既に存在していなければなりません。表スペースを指定しなかった場合は、ユーザーが USE 特権を持つ表スペースが割り当てられます。

ターゲット表を使用できるのは、単一イベント・モニターに限られます。別のイベント・モニターにターゲット表を定義している場合、または他の理由でターゲット表を作成できない場合、CREATE EVENT MONITOR ステートメントは失敗します。

表スペース名は表名の後に、オプションの IN 節を付けて追加することができます。 DB2 データベース・マネージャーが自動的に作成するターゲット表とは異なり、表スペースがイベント・モニター定義に組み込まれている場合には、それがすでに存在していなければなりません。表スペースが指定されていない場合には、定義プログラムが USE 特権を有する表スペースが割り当てられます。

パーティション・データベース環境では、表書き込みイベント・モニターは、イベント・モニター表を含む表スペースが存在するデータベース・パーティション上でのみアクティブになります。特定のデータベース・パーティション上にアクティブ・イベント・モニターのターゲット表スペースが存在しない場合、そのデータベース・パーティションのイベント・モニターは非活動状態にされ、**db2diag** コマンド・ログ・ファイルにエラーが書き込まれます。

イベント・モニター・データの検索時のパフォーマンスを向上させるために、イベント表に索引を作成することができます。トリガー、関係保全、制約などの表属性を追加した場合、イベント・モニターはそれらを無視します。

例えば、次のステートメントは、event_connheader、event_stmt、および event_subsection 論理データ・グループを使用して、STATEMENTS イベントをキャプチャーするイベント・モニターを定義します。3 つのターゲット表のそれぞれには、異なるスキーマ、表、および表スペースの組み合わせがあります。

```
CREATE EVENT MONITOR test FOR STATEMENTS
WRITE TO TABLE CONNHEADER,
STMT (TABLE mydept.statements),
SUBSECTION (TABLE subsections, IN mytablespace)
```

上記のステートメントをユーザー riihi が発行したとすると、各ターゲット表の派生名および表スペースは以下のようになります。

- CONNHEADER: riihi.connheader_test (デフォルトの表スペース)
- STMT: mydept.statements (デフォルトの表スペース)
- SUBSECTION: riihi.subsections (表スペース mytablespace)

イベント・モニターを活動化しているときにターゲット表が存在しない場合でも、活動化は続行しますが、ターゲット表に挿入されるはずだったデータは無視されます。また、ターゲット表にモニター・エレメント専用の列がない場合にも、そのモニター・エレメントは無視されます。

表書き込みイベント・モニターがアクティブな場合、イベント・レコードを保管する表スペースがその限界に達する可能性があります。DMS 表スペースの場合、このリスクを制御するために、イベント・モニターを非アクティブ化する表スペース容量のパーセンテージを定義することができます。この値は、CREATE EVENT MONITOR ステートメントの PCTDEACTIVATE 節に指定できます。 SMS 表スペースの場合、この値は 100 に設定されます。ターゲット表スペースの自動サイズ変更フィーチャーを有効化した場合は、 PCTDEACTIVATE 値を 100 に設定する必要があります。

非パーティション・データベース環境では、最後のアプリケーションが終了すると (それまでにデータベースが明示的にアクティブ化されていないと)、表書き込みイベント・モニターはすべて非アクティブ化されます。パーティション・データベース環境では、カタログ・パーティションが非アクティブ化されると表イベント・モニターへの書き込みが非アクティブ化されます。

論理データ・グループおよびイベント・モニター出力表:

一緒に使用されることが多いモニター・エレメントは、*論理データ・グループ* にグループ化されています。表書き込みイベント・モニターは、一般的に、キャプチャーするモニター・エレメントの論理データ・グループごとに 1 つの出力表を生成します。

次の表は、イベント・タイプごとに、デフォルトのターゲット表名を示しています。

表 7. 表書き込みイベント・モニターの論理データ・グループ

イベント・タイプ	論理データ・グループ	論理グループの情報	論理グループに属するエレメントが書き込まれる表の名前
DEADLOCKS[1]	event_connheader	接続メタデータ。	CONNHEADER_*evmon-name*
	event_deadlock	デッドロック・データ。	DEADLOCK_*evmon-name*
	event_dlconn	デッドロックに関係しているアプリケーションとロック	DLCONN_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
DEADLOCKS WITH DETAILS[1]	event_connheader	接続メタデータ。	CONNHEADER_*evmon-name*
	event_deadlock	デッドロック・データ。	DEADLOCK_*evmon-name*
	event_detailed_dlconn	デッドロックに関係しているアプリケーション	DLCONN_*evmon-name*
	dllock	デッドロックに関係しているロック	DLLOCK_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
DEADLOCKS WITH DETAILS HISTORY[1]	event_connheader	接続メタデータ。	CONNHEADER_*evmon-name*
	event_deadlock	デッドロック・データ。	DEADLOCK_*evmon-name*
	event_detailed_dlconn	デッドロックに関係しているアプリケーション	DLCONN_*evmon-name*
	dllock	デッドロックに関係しているロック	DLLOCK_*evmon-name*
	event_stmt	作業単位内の以前のステートメントのリスト。	STMTHIST_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*

表 7. 表書き込みイベント・モニターの論理データ・グループ (続き)

イベント・タイプ	論理データ・グループ	論理グループの情報	論理グループに属するエレメントが書き込まれる表の名前
DEADLOCKS WITH DETAILS HISTORY VALUES[1]	event_connheader	接続メタデータ。	CONNHEADER_*evmon-name*
	event_deadlock	デッドロック・データ。	DEADLOCK_*evmon-name*
	event_detailed_dlconn	デッドロックに関係しているアプリケーション	DLCONN_*evmon-name*
	dllock	デッドロックに関係しているロック	DLLOCK_*evmon-name*
	event_stmt_history	作業単位内の以前のステートメントのリスト。	STMTHIST_*evmon-name*
	STMTVALS	STMTHIST 表内のステートメントの入力データ値。	STMTVALS_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
STATEMENT	event_connheader	接続メタデータ。	CONNHEADER_*evmon-name*
	event_stmt	ステートメント・データ。	STMT_*evmon-name*
	event_subsection	サブセクションに固有のステートメント・データ。	SUBSECTION_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
TRANSACTIONS[3]	event_connheader	接続メタデータ。	CONNHEADER_*evmon-name*
	event_xact	トランザクション・データ。	XACT_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
CONNECTIONS	event_connheader	接続メタデータ。	CONNHEADER_*evmon-name*
	event_conn	接続データ。	CONN_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
	event_connmemuse	メモリー・プール・メタデータ。	CONNMEMUSE_*evmon-name*
DATABASE	event_db	データベース・マネージャー・データ。	DB_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
	event_dbmemuse	メモリー・プール・メタデータ。	DBMEMUSE_*evmon-name*

表 7. 表書き込みイベント・モニターの論理データ・グループ (続き)

イベント・タイプ	論理データ・グループ	論理グループの情報	論理グループに属するエレメントが書き込まれる表の名前
BUFFERPOOLS	event_bufferpool	バッファー・プール・データ。	BUFFERPOOL_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
TABLESPACES	event_tablespace	表スペース・データ。	TABLESPACE_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
TABLES	event_table	表データ。	TABLE_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
ACTIVITIES	event_activity	実行が完了した、または進行中にキャプチャーされたアクティビティー。	ACTIVITY_*evmon-name*
	event_activitystmt	ステートメントであるアクティビティーに関するステートメント情報。	ACTIVITYSTMT_*evmon-name*
	event_activityvals	アクティビティーの入力データ値。CLOB、REF、BOOLEAN、STRUCT、DATALINK、LONG VARGRAPHIC、LONG、XMLLOB、および DBCLOB のデータ・タイプは報告されません。	ACTIVITYVALS_*evmon-name*
	activity_metrics	アクティビティー・メトリック。	ACTIVITYMETRICS_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
STATISTICS	event_scstats	システム内のそれぞれのサービス・クラス、処理クラス、またはワークロードで実行されたアクティビティーから算出される統計。	SCSTATS_*evmon-name*
	event_wcstats		WCSTATS_*evmon-name*
	event_wlstats		WLSTATS_*evmon-name*
	event_histogrambin		HISTOGRAMBIN_*evmon-name*
	event_qstats		QSTATS_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*

表 7. 表書き込みイベント・モニターの論理データ・グループ (続き)

イベント・タイプ	論理データ・グループ	論理グループの情報	論理グループに属するエレメントが書き込まれる表の名前
THRESHOLD VIOLATIONS	event_ thresholdviolations	違反したしきい値とその時間についてのリスト。	THRESHOLDVIOLATIONS_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
LOCKING	lock	ロック待機、ロック・タイムアウト、またはデッドロック・イベント情報のサマリー。	LOCK_EVENT*evmon-name*
	lock_participants	ロックの参加者に関する情報。	LOCK_PARTICIPANTS_*evmon-name*
	lock_participant _activities	ロックの各参加者のアクティビティー・データ。	LOCK_PARTICIPANT_ACTIVITIES_*evmon-name*
	lock_activity_values	特定のアクティビティーで処理されていた特定のデータに関する詳細	LOCK_ACTIVITY_VALUES_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*
PACKAGE CACHE	pkgcache	パッケージ・キャッシュ・イベント情報のサマリー。この情報に含まれている詳細なメトリックは、METRICS 列に XML 形式で入っています。	PKGCACHE_EVENT*evmon-name*
	pkgcache_metrics	PKGCACHE 表の METRICS 列に含まれるメトリックと同じものが入っている表。	PKGCACHE_METRICS_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*

表 7. 表書き込みイベント・モニターの論理データ・グループ (続き)

イベント・タイプ	論理データ・グループ	論理グループの情報	論理グループに属するエレメントが書き込まれる表の名前
UNIT OF WORK	uow	作業単位イベント情報のサマリー。この情報に含まれている詳細なメトリックは、METRICS 列に XML 形式で入っています。	UOW_EVENT*evmon-name*
	uow_metrics	PKGCACHE 表の METRICS 列に含まれるメトリックと同じものが入っている表。	UOW_METRICS_*evmon-name*
	uow_package_list	パッケージ・リストの詳細情報。[4]	UOW_PACKAGE_LIST_*evmon-name*
	uow_executable_list	実行可能リスト情報。[4]	UOW_EXECUTABLE_LIST_*evmon-name*
	CONTROL[2]	イベント・モニター・メタデータ。	CONTROL_*evmon-name*

1 このオプションは推奨されておらず、今後のリリースで除去される可能性があります。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

2 CONTROL 論理グループは、 event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

3 このオプションは推奨されておらず、今後のリリースで除去される可能性があります。トランザクション・イベントをモニターするには、 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用してください。

4 作業単位イベント・モニターに、作成する出力表を明示的に指定した場合を除き、この表はデフォルトで含まれます。関連情報を収集するための構成パラメーター (**mon_uow_pkglist** または **mon_uow_execlist**) を ON に設定していない場合であっても表は作成されます。ただし、データは入っていません。

次の論理データ・グループは、表書き込みイベント・モニターでは収集されません。

- log_stream_header
- log_header
- dbheader (**conn_time** モニター・エレメントだけが収集される)

イベント・モニター表の各列のデータ・タイプは、列によって表されるモニター・エレメントのデータ・タイプに対応します。以下の表に、モニター・エレメントの元のシステム・モニター・データ・タイプ (sqlmon.h ファイルにある) を、表列の SQL データ・タイプに対応させた、データ・タイプのマッピング・セットを示します。

表 8. システム・モニター・データ・タイプのマッピング

システム・モニターのデータ・タイプ	SQL データ・タイプ
SQLM_TYPE_STRING	CHAR[n]、VARCHAR[n]、CLOB[n]

表 8. システム・モニター・データ・タイプのマッピング (続き)

システム・モニターのデータ・タイプ	SQL データ・タイプ
SQLM_TYPE_U8BIT および SQLM_TYPE_8BIT	SMALLINT、INTEGER、または BIGINT
SQLM_TYPE_U16BIT および SQLM_TYPE_16BIT	SMALLINT、INTEGER、または BIGINT
SQLM_TYPE_U32BIT および SQLM_TYPE_32BIT	INTEGER または BIGINT
SQLM_TYPE_U64BIT および SQLM_TYPE_64BIT	BIGINT
SQLM_TIMESTAMP	TIMESTAMP
SQLM_TIME	BIGINT
SQLCA: SQLERRMC	VARCHAR[72]
SQLCA: SQLSTATE	CHAR[5]
SQLCA: SQLWARN	CHAR[11]
SQLCA: 他のフィールド	INTEGER または BIGINT
SQLM_TYPE_HANDLE	BLOB[n]

注:

1. すべて列は NOT NULL です。
2. CLOB 列がある表のパフォーマンスは VARCHAR 列がある表より劣るため、stmt evmGroup (またはデッドロックの詳細を使用するときに evmGroup) を指定するときは、 TRUNC キーワードを使用することを検討してください。
3. SQLM_TYPE_HANDLE は、コンパイル環境ハンドル・オブジェクトを表すために使用されます。

未フォーマット・イベント (UE) 表に書き込むイベント・モニターの作成

イベント・モニター・データ収集のパフォーマンスが特に重要である場合は、イベント・モニターがその出力を未フォーマット・イベント (UE) 表に書き込むようにすることができます。 UE 表へのデータ書き込みの多くは、インライン・バイナリー・データとして書き込まれます。このようにすると、データ収集時の入出力をより高速にできます。

UE 表がイベント・モニターの通常表よりも優っているもう 1 つの点は、通常、イベント・モニターの作成時点でさまざまなオプション (使用するバッファーのサイズ、イベント・モニターをブロック化するか非ブロック化するか、どのタイプのデータ (論理グループ) を収集する必要があるか) を考慮する必要がないということです。ただし、収集されるデータの大部分がバイナリー・フォーマットであるため、イベント・データを調査できるようにするには、UE 表を後処理する必要があります。

注: IBM DB2 10.1 バージョン 10.1 以降、UE 表に関連して以下のフィーチャーが使用できるようになりました。

- プロシージャー EVMON_UPGRADE_TABLES を使用して、以前のリリースのイベント・モニターにより生成された UE 表をアップグレードできます。この機能により、DB2 製品のアップグレード時に、イベント・モニター・データをより容易に保持することができます。

- 出力を UE 表に書き込めるすべてのイベント・モニターは、通常表にも書き込むことができます。
- プロシージャー EVMON_FORMAT_UE_TO_TABLES のオプション PRUNE_UE_TABLE を使用して、不要なデータを UE 表から除去することができます。

始める前に

未フォーマット・イベント表に書き込むイベント・モニターを作成するときは、以下の考慮事項に留意してください。

- UE 表に書き込むイベント・モニターを作成するには、SQLADM 権限または DBADM 権限が必要です。
- 未フォーマット・イベント表には、パフォーマンスが最適になるような表スペースを使用します。表スペースを作成するときは、以下のガイドラインに留意してください。
  - ページ・サイズ (PAGESIZE) を可能な限り大きく指定します (最大で 32KB)。ページ・サイズが大きいと、表の行を使用して、イベント・データを含む BLOB をインラインで書き込むことができます。ページ・サイズが小さすぎて BLOB をインライン化できない場合、イベント・モニターのパフォーマンスが低下することがあります。データベース・マネージャーは、未フォーマット・イベント表内の event_data BLOB 列をインライン化しようとしますが、これは必ずしも可能ではありません。未フォーマット・イベント表内の行がインライン化されているかどうかを確認するには、ADMIN_IS_INLINED 関数を使用します。行がインライン化されていない場合は、行が必要とするスペースを判別するために、ADMIN_EST_INLINE_LENGTH 関数を使用します。
  - NO FILE CACHING SYSTEM オプションを指定します。
- パーティション・データベース環境では、表スペースが存在するパーティションを考慮します。ターゲットの未フォーマット・イベント表の表スペースが、あるデータベース・パーティション上に存在していない場合、そのターゲットの未フォーマット・イベント表のデータは無視されます。この動作を利用すると、ユーザーは、特定のデータベース・パーティションにのみ存在する表スペースを作成することにより、選択するモニター用にデータベース・パーティションのサブセットを選択できることになります。

このタスクについて

以下のイベント・モニターのタイプは、UE 表の使用をサポートしています。

- 作業単位
- パッケージ・キャッシュ
- ロッキング

注: 未フォーマット・イベント表はその名前に反して、リレーショナル表です。例えばロッキング・イベント・モニターにより生成される UE 表と、ロッキング・イベント・モニターにより生成される通常表との主な違いは、UE 表の大部分のデータが EVENT_DATA 列にバイナリー・フォーマットで書き込まれるということです。 UE 表の構造について詳しくは、146 ページの『未フォーマット・イベント表列の定義』を参照してください。

手順

UE 表に書き込むイベント・モニターを作成するには、以下のようにします。

- WRITE TO UNFORMATTED EVENT TABLE 節を使用して、CREATE EVENT MONITOR ステートメントを作成します。 例えば、uowmon という作業単位イベント・モニターを作成するには、次のようなステートメントを使用できます。

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
```

  デフォルトでは、イベント・モニターが作成する UE 表の名前は、イベント・モニターの名前と同じです。

- デフォルトの表名とは別の名前を指定するには、TABLE 節を使用します。 例えば、UE 表の名前を myunitsofwork にしようとする場合、次のようなステートメントを構成します。

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
                    TABLE myunitsofwork
```

  IN *tablespace-name* 節を使用して、UE 表を格納するための表スペースを指定することもできます。

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
                    TABLE myunitsofwork
                    IN mytablespace
```

  または

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
                    IN mytablespace
```

  最初の例では、UE 表 myunitsofwork を表スペース mytablespace に配置します。 2 番目の例では、(表の名前が指定されていないためデフォルトで) uowmon という UE 表を表スペース mytablespace に配置します。

- デフォルトでは、UE 表に書き込むイベント・モニターが作成されて、データベースを活動化するとこれも自動的に活動化されます。 MANUALSTART 節を使用して、この動作をオーバーライドすることができます。

```
CREATE EVENT MONITOR uowmon FOR UNIT OF WORK
                    WRITE TO UNFORMATTED EVENT TABLE
                    MANUALSTART
```

  前述の例の場合、イベント・モニター uowmon は、SET EVENT MONITOR STATE ステートメントを使用して必ず手動で活動化する必要があります。

次のタスク

デフォルトでは、バージョン 9.7 以降に導入されたイベント・モニターは、AUTOSTART イベント・モニターとして作成されます。データベースが次に活動化されるとき、およびその後データベースが活動化されるときに、これらのイベント・モニターが自動的に活動化されます。データベースが次に活動化される前にイベント・モニターをただちに活動化しようとするときは、SET EVENT MONITOR STATE ステートメントを使用してイベント・モニターを手動で開始します。ロッキング、作業単位、およびパッケージ・キャッシュの各イベント・モニターだけでな

く、データ収集も有効化する必要があります。

未フォーマット・イベント表列の定義:

WRITE TO UNFORMATTED EVENT TABLE 節を含む CREATE EVENT MONITOR ステートメントを発行すると、未フォーマット・イベント表が作成されます。列定義は、分析するデータを抽出する場合や、表で不要なデータを整理する場合に役立ちます。

フォーマットされていないイベント表の列定義は、以下のいずれかのルーチンを使用して、フォーマットされていないイベント表からデータを抽出する場合に役立ちます。

- EVMON_FORMAT_UE_TO_XML - 未フォーマット・イベント表から XML 文書にデータを抽出します。
- EVMON_FORMAT_UE_TO_TABLES - 未フォーマット・イベント表から一連のリレーショナル表にデータを抽出します。

これらのルーチンの呼び出しは、抽出する行を指定する SELECT ステートメントを受け入れます。フォーマットされていないイベント表列定義を使用して、SELECT ステートメントの構成を支援します。

フォーマットされていないイベント表に書き込まれたイベント・データの自動パージは行われません。データは表から手動でパージする必要があります。フォーマットされていないイベント表の列定義は、ターゲットとなるレコードのセットをパージする場合に役立ちます。別のオプションは、TRUNCATE TABLE ステートメントを使用して表のすべての行を削除することです。

CREATE EVENT MONITOR ステートメントの一部として、関連した未フォーマット・イベント表に付ける名前を指定できます。指定されない場合、イベント・モニターと同じ名前がデフォルトで使用されます。SYSCAT.EVENTTABLES カタログ・ビューは、イベント・モニター、関連した未フォーマット表、および他の詳細をリストします。

次の表は、フォーマットされていないイベント表の列について記述したものです。キー列は event_data 列です。他の列は、関心があるイベントを探索するために使用できる ID を表します。表列の他の属性については、DESCRIBE ステートメントを発行してください。

表 9. 未フォーマット・イベント表列の定義

列名	列データ・タイプ	列の記述
appl_id	VARCHAR	appl_id - アプリケーション ID モニター・エレメント
appl_name	VARCHAR	appl_name - アプリケーション名モニター・エレメント

表 9. 未フォーマット・イベント表列の定義 (続き)

列名	列データ・タイプ	列の記述
event_correlation_id	BIT DATA	オプションのイベント相関 ID。NULL 値は、イベント相関 ID が使用不可だったことを示します。 この値はイベント・モニター・タイプに基づいています。 • LOCKING - 将来の利用のために予約済み • UOW- 将来の利用のために予約済み
event_data	BLOB	イベント・モニターによってキャプチャーされるイベント用のイベント・レコード・データ全体 (元のバイナリー形式で保管される)。
event_id	INTEGER	event_id - イベント ID モニター・エレメント
event_timestamp	TIMESTAMP	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
event_type	VARCHAR	event_type - イベント・タイプ・モニター・エレメント
member	SMALLINT	member - データベース・メンバー・モニター・エレメント
partitioning_key	INTEGER	表のパーティション・キー。これは、イベント・モニターが実行されるデータベース・パーティションで、挿入操作がローカルに実行されるためのものです。
record_seq_num	INTEGER	event_data 列内に保管されるレコードのシーケンス番号。
record_type	INTEGER	event_data 列内に保管されるレコードのタイプ。
service_subclass_name	VARCHAR	service_subclass_name - サービス・サブクラス名モニター・エレメント
service_superclass_name	VARCHAR	service_superclass_name - サービス・スーパークラス名モニター・エレメント
workload_name	VARCHAR	workload_name - ワークロード名モニター・エレメント

表 9. 未フォーマット・イベント表列の定義 (続き)

列名	列データ・タイプ	列の記述
mon_interval_id	BIGINT	mon_interval_id - モニター間隔 ID のモニター・エレメント

通常の表と UE 表の出力の相違点:

一般的に、通常の表と未フォーマット・イベント (UE) 表の両方に書き込めるイベント・モニターは、同じデータをキャプチャーします。ただし、注意すべきわずかな相違点もいくつかあります。

列の順序

最初の相違点は、表の列の順序に関するものです。イベント・モニターが通常の表を生成する場合、 UE 表に対して EVMON_FORMAT_UE_TO_TABLES を実行して生成される出力と比べて、通常、列はアルファベット順になります。ただし、次の 2 つの例外があります。

- PARTITION_KEY 列が出力に含まれている場合は、これが最初の列です。
- メトリックをレポートする表の場合は、関連する列がグループ化されます。例えば、システムで費やされた時間をレポートする各列は、グループ化されます。

戻される列

もう 1 つは、列のデータ・タイプに関するものです。ほとんどの場合、表書き込みイベント・モニターの列は、 UE 表に対して EVMON_FORMAT_UE_TO_TABLES を実行して生成される列と同じです。ただし、いくつかの相違点があります。これらの相違点について、表 10 に要約します。

表 10. 戻される列の相違点のサマリー

論理データ・グループ	通常の表に戻される列	EVMON_FORMAT_UE_TO_TABLES から戻される列
すべてのグループ	PARTITION_KEY 列を含む	
uow	TYPE 列を含む	TYPE 列を含まない
uow_package_list	ROUTINE_ID データ・タイプが BIGINT	ROUTINE_ID データ・タイプが INTEGER
pkgcache	XMLID 列を含まない	XMLID 列を含む
lock	DL_CONNS データ・タイプが BIGINT	DL_CONNS データ・タイプが INTEGER
	ROLLED_BACK_PARTICIPANT_NO データ・タイプが SMALLINT	ROLLED_BACK_PARTICIPANT_NO データ・タイプが INTEGER
	XMLID 列を含まない	XMLID 列を含む

表 10. 戻される列の相違点のサマリー (続き)

論理データ・グループ	通常の表に戻される列	EVMON_FORMAT_UE_TO_TABLES から戻される列
lock_participants	AGENT_STATUS データ・タイプが BIGINT	AGENT_STATUS データ・タイプが INTEGER
	APPL_ID データ・タイプが VARCHAR(64)	APPL_ID データ・タイプが VARCHAR(128)
	APPL_NAME データ・タイプが VARCHAR(255)	APPL_NAME データ・タイプが VARCHAR(128)
	CLIENT_ACCTING データ・タイプが VARCHAR(200)	CLIENT_ACCTNG データ・タイプが VARCHAR(255)
	TABLESPACE_NAME データ・タイプが VARCHAR(18)	TABLESPACE_NAME データ・タイプが VARCHAR(128)
	XMLID 列を含まない	XMLID 列を含む
	INTERNAL_DATA 列を含む	INTERNAL_DATA 内のデータが含まれない。
lock_participant_activities	ACTIVITY_ID データ・タイプが BIGINT	ACTIVITY_ID データ・タイプが INTEGER
	XMLID[1] ではなく EVENT_ID、EVENT_TYPE、および EVENT_TIMESTAMP を含む。	XMLID 列を含む
	CONSISTENCY_TOKEN が CHAR(8)	CONSISTENCY_TOKEN が VARCHAR(8)
lock_activity_values	ACTIVITY_ID データ・タイプが BIGINT	ACTIVITY_ID データ・タイプが INTEGER
	PARTICIPANT_NO データ・タイプが SMALLINT	PARTICIPANT_NO データ・タイプが INTEGER
	XMLID.[1] ではなく EVENT_ID、EVENT_TYPE、および EVENT_TIMESTAMP を含む。	XMLID 列を含む

1. XMLID 列は、 event_header、event_id、event_type、event_timestamp、およびパーティション・モニター・エレメントを連結して構成される複合モニター・エレメントを表します。

ファイル・イベント・モニターの作成

イベント・モニターの作成時に、それが収集した情報の保管場所を決定する必要があります。ファイル・イベント・モニターは、イベント・レコードをファイルに保管します。ファイル・イベント・モニターとそのオプションは、CREATE EVENT MONITOR ステートメントによって定義されます。

始める前に

ファイル・イベント・モニターを作成するには、SQLADM または DBADM 権限が必要です。

このタスクについて

ファイル・イベント・モニターは、 8 桁の番号の付いた、拡張子 "evt" のファイル (例えば 00000000.evt、 00000001.evt、00000002.evt) にイベント・レコードを流します。データを小さく分割した場合でも、全体を 1 つの論理ファイルと見なす必要があります (つまり、データ・ストリームの開始はファイル 00000000.evt の最初のバイトであり、データ・ストリームの終了はファイル nnnnnnnn.evt 内の最後のバイトです)。イベント・モニターでは、 1 つのイベント・レコードが 2 つのファイルにわたって入ることはありません。

手順

1. イベント・モニター・データがファイル (またはファイルのセット) に収集されることを指定し、イベント・ファイルが保管されるディレクトリーの場所を指定します。

   ```
   CREATE EVENT MONITOR dlmon FOR eventtype
                       WRITE TO FILE '/tmp/dlevents'
   ```

   dlmon はイベント・モニターの名前です。

   /tmp/dlevents は、イベント・モニターがイベント・ファイルを書き込むディレクトリー・パス (UNIX システム上) の名前です。

2. モニター対象のイベントのタイプを指定します。単一イベント・モニターで、1 つ以上のイベント・タイプをモニターすることができます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents'
   ```

   このイベント・モニターは、CONNECTIONS および DEADLOCKS WITH DETAILS イベント・タイプをモニターします。

3. BUFFERSIZE 値を調整することによって、ファイル・イベント・モニター・バッファーのサイズを指定します (4K ページ単位)。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
   ```

   8 は、2 つのイベント・ファイル・バッファーの容量です (4K ページ単位)。

   個々のバッファーのデフォルト・サイズは 4 ページです (16K のバッファーが 2 つ割り振られます)。最小サイズは 1 ページです。バッファーはモニター・ヒープから割り振られるので、バッファーの最大サイズはこのヒープのサイズによって制限されます。パフォーマンス上の理由で、アクティブ率の高いイベント・モニターには、アクティブ率が比較的低いものよりも大きなバッファーが必要です。

4. イベント・モニターをブロック化または非ブロック化する必要があるかどうかを指定します。ブロック化イベント・モニターの場合、イベントを生成する各エージェントは、イベント・バッファーが満杯であれば、それがファイルに書き込まれるまで待機します。これは、データベースのパフォーマンスを低下させることがあります。なぜなら、バッファーがクリアされるまで、延期されたエージェントおよび従属エージェントが実行できなくなるからです。イベント・データが脱落しないようにするには、BLOCKED 節を使用します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                    BLOCKED
```

イベント・モニターはデフォルトではブロック化されます。すべての単一イベント・レコードを収集することよりも、データベース・パフォーマンスのほうが重要な場合には、非ブロック化イベント・モニターを使用してください。その場合、イベントを生成する各エージェントは、イベント・バッファーが満杯のときに、それがファイルに書き込まれるのを待機しません。結果として、非ブロック化イベント・モニターは、アクティブ率の高いシステムではデータの脱落という影響を受けます。イベント・モニターによって生じる処理時間の増加を最小限に抑えるには、NONBLOCKED 節を使用します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                    NONBLOCKED
```

5. イベント・モニターについて収集できるイベント・ファイルの最大数を指定します。この限度に達すると、イベント・モニターは自分自身を非アクティブ化します。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       NONBLOCKED MAXFILES 5
   ```

   5 は、作成されるイベント・ファイルの最大数です。

   イベント・モニターが作成できるイベント・ファイルの数に制限がないことを指定することもできます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       NONBLOCKED MAXFILES NONE
   ```

6. イベント・モニターによって作成されるそれぞれのイベント・ファイルごとに、最大サイズを指定します (4K ページ単位)。この限度に達すると、新規ファイルが作成されます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       NONBLOCKED MAXFILES 5 MAXFILESIZE 32
   ```

   32 は、1 つのイベント・ファイルに入れることができる 4K ページ単位の最大数です。

   この値は、BUFFERSIZE パラメーターによって指定された値よりも大きくなければなりません。また、イベント・ファイルのサイズに制限がないことを指定することもできます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       NONBLOCKED MAXFILES NONE MAXFILESIZE NONE
   ```

7. データベースを開始するたびに、イベント・モニターが自動的に活動化されるかどうかを指定します。デフォルトでは、データベースの開始時にイベント・モニター (WLM イベント・モニターは例外) は自動的には活動化されません。
   - データベースの開始時に自動的に開始するイベント・モニターを作成するには、次のステートメントを発行する。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                    NONBLOCKED AUTOSTART
```

- データベースの開始時に自動的に開始しないイベント・モニターを作成するには、次のステートメントを発行する。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                    NONBLOCKED MANUALSTART
```

8. イベント・モニターを活動化または非アクティブ化するには、SET EVENT MONITOR STATE ステートメントを使用します。

タスクの結果

ファイル・イベント・モニターが作成されて活動化されると、指定されたイベントが発生するたびに、モニター・データを記録します。

イベント・モニターのファイル管理:

一部のイベント・モニターでは、イベント・データをテキスト・ファイルに書き込むことができます。CREATE または ALTER EVENT MONITOR ステートメントのオプションを使用すると、作成するファイル数およびファイル・サイズの上限を構成できます。

ファイル・イベント・モニターにより、イベント・モニターは、イベント・レコードをファイルに保管することができます。イベント・モニターのすべての出力は、CREATE EVENT MONITOR ステートメントの FILE パラメーターで指定されたディレクトリーに入れられます。したがって、モニターが活動化される前に、そのディレクトリーが存在していなければなりません。存在しないと、SET EVENT MONITOR コマンドがエラーを戻します。ディレクトリーが存在しない場合、データベース・マネージャーはそれを作成しません。

重要: ファイル・イベント・モニターが最初に活動化されるとき、このディレクトリーに db2event.ctl という名前の制御ファイルが作成されます。このファイルは除去したり修正したりしないでください。

デフォルトでは、イベント・モニターはそのトレースを 00000000.evt と呼ばれる単一のファイルに書き込みます。このファイルは、ファイル・システム上にスペースがあるかぎり大きくなります。 CREATE EVENT MONITOR ステートメントの **MAXFILESIZE** パラメーターを使ってファイル・サイズ限界を指定した場合には、1 つのファイルが満杯になると、新しいファイルに出力が送られます。新しいファイルが作成されるたびに、ファイル名を構成する番号が 1 つずつ増えます。したがって、アクティブ・ファイルは、番号の一番大きいファイルとなります。

また、CREATE EVENT MONITOR ステートメントの **MAXFILES** パラメーターを使って、イベント・モニター・トレース全体の最大サイズを制限できます。ファイルの数が MAXFILES で定義された最大値を超えると、イベント・モニターは自らを非アクティブ化し、以下のメッセージが管理通知ログに書き込まれます。

```
DIA1601I Event Monitor monitor-name was deactivated when it reached
its preset MAXFILES and MAXFILESIZE limit.
```

このメッセージを受け取った場合、どのイベント・モニター・ファイルも削除しないでください。削除した場合、**db2evmon** コマンドを使ってイベント・モニター情報を表示することが全くできなくなります (残りのファイルに入っている情報を含む)。そうする代わりに、以下のいずれかの操作を行ってください。

- **MAXFILES** および **MAXFILESIZE** 制限を使わずにイベント・モニターを再作成します。
- **MAXFILES** および **MAXFILESIZE** パラメーターによって実施される制限をそのまま残します。ディレクトリー内の最新の *.evt ファイルを除くすべてのファイルを、別のディレクトリーまたはファイル・システムに移動します。その後、新しいディレクトリー内のファイルからイベント・モニター情報を表示できます。必要に応じて、これを自動的に行うスクリプトを作成できます。

いずれの場合も、DIA1601I メッセージ受信後の情報収集を再開するために、ステートメント **SET EVENT MONITOR** *event-monitor-name* **STATE 1** を使ってイベント・モニターを再び活動化する必要があります。

ファイル・イベント・モニターを再始動する場合、既存のデータを消去するか、既存のデータの後に新規データを追加することができます。このオプションは、CREATE EVENT MONITOR ステートメントで指定されます。そこでは、APPEND モニターまたは REPLACE モニターのどちらかを作成することができます。APPEND がデフォルトのオプションです。 APPEND イベント・モニターは、最後に使用していたファイルの終わりから書き込みを開始します。そのファイルを除去すると、次の順番のファイル番号が使用されます。追加のイベント・モニターが再始動されると、start_event だけが生成されます。イベント・ログ・ヘッダーとデータベース・ヘッダーは、最初の活動化のときに生成されます。 REPLACE イベント・モニターは常に既存のイベント・ファイルを削除し、00000000.evt から書き込みを開始します。

注: イベント・モニターに関して REPLACE オプションを使用しなかった場合、以下の手順を実行して、イベント・モニターによる新しいデータ・セットの収集を強制的に開始することができます。

1. **SET EVENT MONITOR** *event-monitor-name* **STATE 0** コマンドを使ってイベント・モニターを非アクティブ化します。
2. CREATE EVENT MONITOR ステートメントの FILE オプションで指定されたディレクトリーの中にあるファイルをすべて削除します。
3. **SET EVENT MONITOR** *event-monitor-name* **STATE 1** コマンドを使ってイベント・モニターを再び活動化します。

ファイル・イベント・モニターがディスク・スペースを使い尽くした場合、システム・エラー・レベル・メッセージを管理通知ログに記録した後、自動的にシャットダウンします。

イベント・モニターがアクティブになっている時点でモニター・データを処理するとよいかもしれません。そのことは可能です。さらに、ファイルの処理を終えたとき、そのファイルを削除し、次のモニター・データのためにスペースを解放することができます。イベント・モニターを停止して再始動しないかぎり、次のファイルに強制的に切り替えることはできません。また、APPEND モードでなければなりません。アクティブ・ファイル内でどのイベントの処理が終わったかを追跡するため

に、処理された最後のレコードのファイル番号と場所を追跡する単純なアプリケーションを作成できます。次回にトレースを処理するときには、アプリケーションはそのファイル場所を検索できます。

パイプ・イベント・モニターの作成

イベント・モニターの作成時に、それが収集した情報の保管場所を決定する必要があります。パイプ・イベント・モニターは、イベント・モニターから Named PIPE に直接イベント・レコードを流します。

始める前に

- パイプ・イベント・モニターを作成するには、SQLADM 権限または DBADM 権限が必要です。
- このタスクは、名前付きパイプが既に作成されていることを前提としています。UNIX または Linux システムで名前付きパイプを作成するには、これらのシステムで提供されている **mkfifo** コマンドを使用します。

このタスクについて

イベント・モニターがイベント・データを書き込んですぐにデータをパイプから読み取るのは、モニター・アプリケーションの責任です。イベント・モニターがデータをパイプに書き込めない場合 (例えばパイプが満杯になっている場合) は、モニター・データは失われます。

パイプ・イベント・モニターは、CREATE EVENT MONITOR ステートメントで定義されます。

手順

1. イベント・モニター・データが Named PIPE に送られることを指定します。

   ```
   CREATE EVENT MONITOR myevmon FOR eventtype
                       WRITE TO PIPE '/home/dbadmin/dlevents'
   ```

   myevmon はイベント・モニターの名前です。

   /home/dbadmin/dlevents は、イベント・モニターがイベント・レコードを送る宛先の Named PIPE (UNIX 上) の名前です。CREATE EVENT MONITOR ステートメントは、UNIX および Windows パイプ名前構文をサポートします。

   CREATE EVENT MONITOR ステートメントで指定される Named PIPE は、イベント・モニターを活動化するときに存在し、開いている必要があります。イベント・モニターが自動的に開始するように指定する場合には、イベント・モニターの作成前に Named PIPE が存在している必要があります。
2. モニター対象のイベントのタイプを指定します。単一イベント・モニターで、1 つ以上のイベント・タイプをモニターすることができます。

   ```
   CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                       WRITE TO PIPE '/home/dbadmin/myevents'
   ```

   このイベント・モニターは、BUFFERPOOLS および TABLESPACES イベント・タイプをモニターします。

3. データベースを開始するたびに、イベント・モニターが自動的に活動化されるかどうかを指定します。デフォルトでは、データベースの開始時にイベント・モニターは自動的には活動化されません。
   - データベースの開始時に自動的に開始するイベント・モニターを作成するには、次のステートメントを発行する。

     ```
     CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                          WRITE TO PIPE '/home/dbadmin/myevents'
                          AUTOSTART
     ```

   - データベースの開始時に自動的に開始しないイベント・モニターを作成するには、次のステートメントを発行する。

     ```
     CREATE EVENT MONITOR myevmon FOR BUFFERPOOLS, TABLESPACES
                          WRITE TO PIPE '/home/dbadmin/myevents
                          MANUALSTART
     ```

4. 名前付きパイプから読み取りを行うクライアント・アプリケーションを始動します。例えば、db2evmon ツールを開始すると、データがパイプに渡されるときにこれを処理することができます。
5. イベント・モニターを活動化または非アクティブ化するには、SET EVENT MONITOR STATE ステートメントを使用します。

タスクの結果

パイプ・イベント・モニターが作成されて活動化されると、指定されたイベントが発生するたびに、モニター・データを記録します。

イベント・モニターの Named PIPE 管理:

一部のイベント・モニターでは、イベント・データを名前付きパイプに書き込むことができます。以下に、名前付きパイプのイベント・モニターをより効率的に使用するためのガイドラインを示します。

パイプ・イベント・モニターでは、Named PIPE によって、イベント・モニターのデータ・ストリームを処理することができます。パイプ・イベント・モニターの使用は、リアルタイムでイベント・レコードを処理する必要がある場合に有用です。別の利点として、アプリケーションがパイプから読み取った時に不要なデータを無視できるので、ストレージ要件を相当減らせる可能性があります。

AIX® では、mkfifo コマンドを使用して Named PIPE を作成することができます。Linux および他の UNIX タイプ (Solaris オペレーティング・システムなど) では、pipe() ルーチンを使用します。 Windows では、CreateNamedPipe() ルーチンを使用して Named PIPE を作成することができます。

データをパイプに送ると、入出力は必ずブロック化され、バッファリングだけがパイプによって実行されます。イベント・モニターがイベント・データを書き込んですぐにデータをパイプから読み取るのは、モニター・アプリケーションの責任です。イベント・モニターがデータをパイプに書き込めない場合 (例えばパイプが満杯になっている場合) は、モニター・データは失われます。

さらに、Named PIPE には、着信するイベント・レコードを処理するための十分なスペースがなければなりません。アプリケーションが十分な速さで Named PIPE か

らデータを読み取らないと、パイプは満杯になり、オーバーフローが発生します。パイプ・バッファーが小さいほど、オーバーフローの発生する可能性が大きくなります。

パイプのオーバーフローが発生すると、モニターはオーバーフローが発生していることを示すオーバーフロー・イベント・レコードを作成します。イベント・モニターは OFF になりませんが、モニター・データは失われます。モニターが非アクティブ化されるときに未解決のオーバーフロー・イベント・レコードがある場合は、診断メッセージが記録されます。それ以外の場合、パイプが書き込み可能になれば、オーバーフロー・イベント・レコードは、パイプに書き込まれます。

パイプに一度に書き込めるデータ量は、基礎となるオペレーティング・システムによって決まります。オペレーティング・システムでパイプ・バッファー・サイズを定義できる場合は、少なくとも 32K のパイプ・バッファーを使用してください。大ボリュームのイベント・モニターの場合、モニター・アプリケーションの処理優先順位を、エージェント処理優先順位と同等かそれ以上の値に設定する必要があります。

アクティビティー・イベント・モニターまたは統計イベント・モニターの単一の書き込み操作から生じたデータ・ストリームに、Named PIPE に書き込めるデータ量より多くのデータが含まれている場合があります。そのような場合は、バッファーに収まるようにデータ・ストリームがブロックに分割されます。各ブロックはヘッダーによって識別され、一番目のブロックは、エレメント ID SQLM_ELM_EVENT_STARTPIPEBLOCK を含む論理ヘッダーによって識別されます。最後のブロックは、エレメント ID SQLM_ELM_EVENT_ENDPIPEBLOCK を含む論理ヘッダーによって識別されます。間にあるブロックはすべて、エレメント ID SQLM_ELM_EVENT_MIDPIPEBLOCK を含む論理ヘッダーによって識別されます。パイプを読み取るモニター・アプリケーションは、これらのヘッダーを認識し、ブロックを再組み立てして完全なデータ・ストリームに戻す必要があります。必要な場合はブロック・ヘッダーを除去してからブロックを再組み立てして、完全かつ有効なデータ・ストリームを形成します。db2evmon ツールにはこの機能が備わっており、 Named PIPE に書き込むイベント・モニターによって生成されるすべてのイベントに、必要に応じてブロックを再組み立てすることで定様式の出力を提供します。選択したイベントまたはモニター・エレメントのみを処理するには、そのような処理を行う独自のアプリケーションを作成します。

表書き込みイベント・モニターおよびファイル・イベント・モニターのバッファー方式

一部の表書き込みイベント・モニターおよびファイル・イベント・モニターは、出力をファイルまたは表に書き出す前にバッファーに格納します。

表 11 に、どのイベント・モニターがこのような出力バッファーを使用するかを示します。

表 11. イベント・モニターおよび出力バッファー

イベント・モニター・タイプ	出力をディスクへ書き込む前にバッファーに書き込むか?
アクティビティー	いいえ
バッファー・プール	はい

表 11. イベント・モニターおよび出力バッファー (続き)

イベント・モニター・タイプ	出力をディスクへ書き込む前にバッファーに書き込むか?
変更履歴	いいえ
接続	はい
データベース	はい
デッドロック (すべてのバージョン)	はい
ロッキング	いいえ
パッケージ・キャッシュ	いいえ
ステートメント	はい
統計	はい
表スペース	はい
表	はい
トランザクション	はい
作業単位	いいえ

バッファーを使用しないイベント・モニターは、ディスクへの出力の書き込みに、より新しくより高速なメカニズムを使用しているため、バッファーを使用する必要がありません。

バッファーを使用するこれらのイベント・モニターの場合、バッファーが満杯になったときに自動的にレコードはディスクに書き込まれます。そのため、より大きなバッファーを指定してディスク・アクセスの回数を減らせば、大量のスループットのあるイベント・モニターのモニター・パフォーマンスを改善することができます。イベント・モニターにそのバッファーをフラッシュさせるには、それを非アクティブ化するか、または FLUSH EVENT MONITOR ステートメントを使用してバッファーを空にする必要があります。

バッファーを使用するイベント・モニターの場合は、イベント・モニターの出力を *ブロック化* するかまたは *非ブロック化* するかを指定できます。ブロック化されたイベント・モニターは、両方のバッファーがいっぱいのときに、モニター・データを送信しているデータベース処理を一時停止します。これは、ブロック化されたイベント・モニターがアクティブのときに、イベント・レコードが廃棄されないようにするためです。一時停止されたデータベース処理とその結果として付随するデータベース処理は、バッファーが書き込まれるまでは実行できません。これは、ワークロードの種類や入出力装置の速度によっては、パフォーマンスを大きく消費することがあります。イベント・モニターはデフォルトではブロック化されます。

ブロック化されていないイベント・モニターは、イベント・モニターがデータを書き込むことのできる速度よりも速くデータが到着するとき、エージェントから来るモニター・データを廃棄します。これにより、イベント・モニターが、他のデータベース・アクティビティーのパフォーマンスに影響しないようにします。

イベント・レコードを廃棄したイベント・モニターは、オーバーフロー・イベントを生成します。これは、モニターがイベントを廃棄した開始時刻と停止時刻、およびその期間に廃棄されたイベントの数を記述します。イベント・モニターは、ペン

ディングのオーバーフローがあることを報告して、終了または非アクティブ化することがあります。その場合、次のメッセージが管理ログに書き込まれます。

```
DIA2503I Event Monitor monitor-name had a pending overflow record
when it was deactivated.
```

イベント・モニター・データの消失は、個々のイベント・レコードについても生じる可能性があります。イベント・レコードの長さがイベント・バッファリング・サイズを超えると、バッファー内に収まらないデータは切り捨てられます。例えば、stmt_text モニター・エレメントをキャプチャーしている場合に、モニター中のデータベースにアタッチしたアプリケーションが長い SQL ステートメントを発行すると、この状態が生じる可能性があります。イベント・レコード情報のすべてをキャプチャーする必要がある場合には、より大きなバッファーを指定してください。より大きなバッファーを指定すれば、ファイルまたは表への書き込み頻度が減ることに留意してください。

イベント・モニター自己記述型データ・ストリーム

パイプまたはファイルに書き込むイベント・モニターは、論理データ・グループのバイナリー・ストリームを出力します。その論理データ・グループは、パイプ・イベント・モニターでもファイル・イベント・モニターでもまったく同じです。データ・ストリームのフォーマットは、**db2evmon** コマンドを使用するか、またはクライアント・アプリケーションを開発することによって行えます。

このデータ・ストリームは、自己記述型フォーマットで表示されます。

159 ページの図 4 では、データ・ストリームの構造を示し、 159 ページの表 12 では、戻される論理データ・グループおよびモニター・エレメントのいくつかの例を示します。

注: これらの例や表では、ID として記述名を使用しています。これらの名前は、実際のデータ・ストリームでは **SQLM_ELM_** という接頭部が付きます。例えば **db_event** は、イベント・モニター出力では SQLM_ELM_DB_EVENT と表示されます。タイプは、実際のデータ・ストリームでは **SQLM_TYPE_** という接頭部が付きます。例えば、ヘッダーはデータ・ストリームで SQLM_TYPE_HEADER と表示されます。

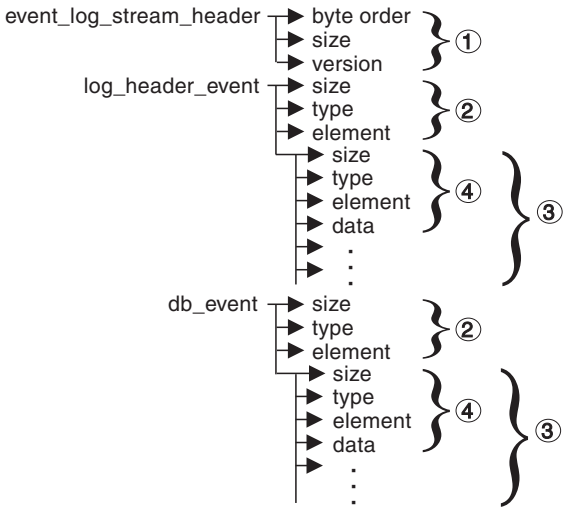


図 4. パイプ・イベント・モニターまたはファイル・イベント・モニターのデータ・ストリーム

1. sqlm_event_log_data_stream_header の構造は、データ・ストリーム内のほかのヘッダーとは異なります。バージョン・フィールドにより、出力を自己記述型データ・ストリームとして処理できるかどうかが決まります。

   このヘッダーのサイズとタイプは、バージョン 6 以前のイベント・モニター・ストリームと同じです。これにより、アプリケーションは、イベント・モニター出力が自己記述型か、バージョン 6 より前の静的形式かを判別できます。

   注: このモニター・エレメントは、データ・ストリームから sizeof(sqlm_event_log_data_stream) のバイトを読み取ることにより抽出されます。
2. 各論理データ・グループは、サイズとエレメント名を示すヘッダーで始まります。これは、 event_log_stream_header にはあてはまりません。そのサイズ・エレメントには、逆方向互換性を保持するためのダミー値が含まれるからです。
3. ヘッダーのサイズ・エレメントは、論理データ・グループのデータ全体のサイズを示します。
4. モニター・エレメント情報が、論理データ・グループ・ヘッダーに続きます。これも自己記述型です。

表 12. イベント・データ・ストリームの例

論理データ・グループ	データ・ストリーム	説明
event_log_stream_header	┬►sqlm_little_endian ├►200 └►sqlm_dbmon_version9	使用されない (以前のリリースとの互換性のため)。 使用されない (以前のリリースとの互換性のため)。 データを戻したデータベース・マネージャーのバージョン。イベント・モニターは自己記述型フォーマットでデータを書き込む。

表 12. イベント・データ・ストリームの例 (続き)

論理データ・グループ	データ・ストリーム	説明
log_header_event	┬►100	論理データ・グループのサイズ。
	├►header	論理データ・グループが始まることを示す。
	├►log_header	論理データ・グループの名前。
	├ →►4	このモニター・エレメントに保管されるデータのサイズ
	│ ├►u32bit	モニター・エレメント・タイプ - 32 ビット数値。
	│ ├►byte_order	収集されたモニター・エレメントの名前。
	│ └►little_endian	このエレメントに対して収集された値。
	└ ┬►2	このモニター・エレメントに保管されるデータのサイズ
	├►u16bit	モニター・エレメント・タイプ - 符号なし 16 ビット数値。
	├►codepage_id	収集されたモニター・エレメントの名前。
	└►850	このエレメントに対して収集された値。
db_event	┬►100	論理データ・グループのサイズ。
	├►header	論理データ・グループが始まることを示す。
	├►db_event	論理データ・グループの名前。
	└ →►4	このモニター・エレメントに保管されるデータのサイズ
	├►u32bit	モニター・エレメント・タイプ - 符号なし 32 ビット数値。
	├►lock_waits	収集されたモニター・エレメントの名前。
	└►2	このエレメントに対して収集された値。

event_log_stream_header は、データを戻したデータベース・マネージャーのバージョンを示します。イベント・モニターは自己記述型フォーマットでデータを書き込みます。スナップショット・モニターと違って、イベント・モニターにはトレースの合計サイズを戻す **size** エレメントがありません。 event_log_stream_header に示された数値は、逆方向互換性のために示されたダミー値です。イベント・トレースの合計サイズは、 event_log_stream_header が書き込まれるときには不明です。通常は、ファイルまたはパイプの終わりに達するまで、イベント・モニター・トレースを読み取ることになります。

ログ・ヘッダーではトレースの特性を記述します。これには、トレースが収集されたサーバーのメモリー・モデル (例えばリトル・エンディアン)、およびデータベースのコード・ページなどの情報が含まれています。トレースを読み取るシステムのメモリー・モデルがサーバーとは異なる場合 (例えば、Windows 2000 システム上の UNIX サーバーからトレースを読み取る場合)、数値に関してバイトのスワッピングを行う必要があります。データベースが、トレースを読み取るマシンとは異なる言語で構成されている場合、コード・ページ変換を行う必要が生じることもあります。トレースを読み取っているとき、 **size** エレメントを使用して、トレース内の論理データ・グループを読み飛ばせます。

イベント・タイプの論理データ・グループへのマッピング

ファイルおよびパイプのイベント・モニターに関しては、イベント・モニターの出力は、順序付けされた一連の論理データ・グループから成っています。どのイベント・モニター・タイプの場合にも、出力レコードには必ず同じ開始論理データ・グループが含まれています。

論理データ・グループが示すフレームは、イベント・モニターによって記録されるイベント・タイプによって異なります。

ファイルおよびパイプ・イベント・モニターの場合、イベント・レコードはどの接続についても生成されることがあるため、ストリーム中にいろいろな順序で現れる場合があります。すなわち、接続 1 のトランザクション・イベントの直後に接続 2 の接続イベントを取得することがあるということです。しかし、単一の接続または単一のイベントに属するレコードは、論理順序で現れます。例えば、ステートメント・レコード (ステートメントの終わり) は、トランザクション・レコード (UOW の終わり) があれば、必ずその前に来ます。同様に、デッドロック・イベント・レコードは、必ずデッドロックに関係する各接続のデッドロック接続イベント・レコードの前に来ます。**アプリケーション ID** または**アプリケーション・ハンドル (agent_id)** を使って、レコードと接続を一致させることができます。

接続ヘッダー・イベントは通常、データベースへのそれぞれの接続ごとに書き込まれます。詳細付きデッドロック・イベント・モニターの場合は、デッドロックが生じたときにだけ書き込まれます。この場合、接続ヘッダー・イベントは、デッドロックに関係したものについてのみ書き込まれます。データベースへのすべての接続について書き込まれるわけではありません。

論理データ・グループは、4 つの異なるレベルに従って配列されます。すなわちモニター、プロローグ、内容、およびエピローグです。以下は、各レベルの詳細な記述です。対応するイベント・タイプおよび論理データ・グループも示しています。

モニター

モニター・レベルの情報は、すべてのイベント・モニターに対して生成されます。これは、イベント・モニターのメタデータから成っています。

表 13. イベント・モニター・データ・ストリーム : モニター・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
モニター・レベル	event_log_stream_header	イベント・モニターのバージョン・レベルおよびバイトの並び順を識別する。アプリケーションはこのヘッダーを使用して、 evmon 出力ストリームを処理できるかどうかを判別できます。

プロローグ

プロローグ情報は、イベント・モニターが活動化されると生成されます。

表 14. イベント・モニター・データ・ストリーム : プロローグ・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
ログ・ヘッダー	event_log_header	トレースの特性、例えば、サーバーのタイプおよびメモリーのレイアウト。
データベース・ヘッダー	event_db_header	データベース名、パスおよび活動化時間。
イベント・モニター開始	event_start	モニターが開始されるか再始動された時間。

表 14. イベント・モニター・データ・ストリーム : プロローグ・セクション (続き)

イベント・タイプ	論理データ・グループ	入手できる情報
接続ヘッダー	event_connheader	現行接続のヘッダーごとに 1 つ。接続時間とアプリケーション名を含みます。イベント接続ヘッダーが生成されるのは、接続、ステートメント、トランザクション、およびデッドロックの各イベント・モニターに対してのみです。詳細付きデッドロック・イベント・モニターは、デッドロックが生じたときだけ接続ヘッダーを生成します。

内容

イベント・モニターの指定されたイベント・タイプに固有の情報は、内容セクションに表示されます。

表 15. イベント・モニター・データ・ストリーム : 内容セクション

イベント・タイプ	論理データ・グループ	入手できる情報
ステートメント・イベント	event_stmt	ステートメント・レベル・データ。動的ステートメントのテキストを含む。ステートメント・イベント・モニターは、フェッチのログを取りません。
サブセクション・イベント	event_subsection	サブセクション・レベル・データ。
トランザクション・イベント[1]	event_xact	トランザクション・レベル・データ。
接続イベント	event_conn	接続レベル・データ。
デッドロック・イベント	event_deadlock	デッドロック・レベル・データ。
デッドロック接続イベント	event_dlconn	デッドロックに関係している接続ごとに 1 つ。関係するアプリケーションと競合しているロックを含みます。
詳細付きデッドロック接続イベント	event_detailed_dlconn、lock	デッドロックに関係している接続ごとに 1 つ。関係するアプリケーション、競合しているロック、現在のステートメント情報、およびアプリケーション競合によって保持された他のロックを含みます。
オーバーフロー	event_overflow	脱落したレコードの数。書き込み装置が (ブロック化されていない) イベント・モニターに追い付かないときに生成されます。

表 15. イベント・モニター・データ・ストリーム : 内容セクション (続き)

イベント・タイプ	論理データ・グループ	入手できる情報
詳細付きデッドロック履歴[2]	event_stmt_history	デッドロックに関係する作業単位で実行されたステートメントのリスト。
詳細付きデッドロック履歴の値[2]	event_data_value	event_stmt_history リスト内のステートメント用のパラメーター・マーカー。
アクティビティー	event_activity	システムで実行が完了した、または完了前にキャプチャーされたアクティビティーのリスト。
	event_activitystmt	アクティビティー・タイプがステートメントであった際に、そのアクティビティーが実行したステートメントに関する情報。
	event_activityvals	SQL ステートメントである各アクティビティーの入力変数として使用されるデータ値。こうしたデータ値には、LOB データ、LONG データ、または構造化タイプ・データは含まれません。
統計	event_scstats	システム内のそれぞれのサービス・クラス、処理クラス、またはワークロードで実行されたアクティビティーから算出される統計、さらにしきい値キューから算出される統計。
	event_wcstats	
	event_wlstats	
	event_qstats	
	event_histogrambin	
しきい値違反	event_thresholdviolations	違反したしきい値とその時間を示す情報。

1 このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用して、トランザクション・イベントをモニターしてください。

2 このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

エピローグ

エピローグ情報は、データベースが非アクティブ化状態にあるとき (最後のアプリケーションが切断を終了したとき) に生成されます。

表 16. イベント・モニター・データ・ストリーム : エピローグ・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
データベース・イベント	event_db	データベース・マネージャー・レベル・データ。
バッファー・プール・イベント	event_bufferpool	バッファー・プール・レベル・データ。
表スペース・イベント	event_tablespace	表スペース・レベル・データ。
表イベント	event_table	表レベル・データ。

データベースに作成されているイベント・モニターのリストの表示

カタログ・ビュー SYSCAT.EVENTMONITORS を使用すると、データベースに既に定義されているイベント・モニターを確認できます。

手順

システム上に定義されているイベント・モニターのリストを表示するには、カタログ・ビュー SYSCAT.EVENTMONITORS を照会します。 例えば、イベント・モニター名、ターゲット出力タイプ (つまり、通常表、ファイル、名前付きパイプ、または未フォーマット・イベント表)、および所有者を含むイベント・モニターのリストを表示するには、以下のような照会を使用できます。

```
SELECT SUBSTR(EVMONNAME,1,20) AS EVMON_NAME, TARGET_TYPE, OWNER
FROM SYSCAT.EVENTMONITORS
```

上記の照会によって、以下のような結果が戻されます。

```
EVMON_NAME           TARGET_TYPE OWNER
-------------------- ----------- ----------------------------------
DB2DETAILDEADLOCK    F           DBADMIN1
CACHEEVMON           T           DBADMIN1
INVTLOCK             T           DBADMIN1
INVTUOW              T           DBADMIN1
INVTACT              T           DBADMIN1
INVTSTATS            T           DBADMIN1
INVTTHRESHOLD        T           DBADMIN1
TABLE_INVTTABLE      T           DBADMIN1
BUFFER_INVT          T           DBADMIN1
TABLESPACES_INVT     T           DBADMIN1
CONNECTIONS_INVT     T           DBADMIN1
TRANSAC_INVT         T           DBADMIN1
DEADLOCK_INVT        T           DBADMIN1
QUINNJN_LOC_UNF      U           DBADMIN1
UNFORM               U           DBADMIN1
RM                   U           DBADMIN1
UOWINVT              U           DBADMIN1
LOCK_UP_STAFF        U           DBADMIN1
INVTLOCK2            T           DBADMIN1
STAFF_UOW            T           DBADMIN1
STAFFSTATS           T           DBADMIN1

  21 record(s) selected.
```

例

また、カタログ・ビューを使用して、特定のイベント・タイプのモニター用の既存のイベント・モニターを表示することもできます。 SYSCAT.EVENTS ビューは、イベント・モニターと、イベント・モニターがデータを記録するイベント・タイプのリストを戻します。

```
SELECT SUBSTR(TYPE,1,20) AS EVENT_TYPE,
   SUBSTR(EVMONNAME,1,20) AS EVENT_MONITOR_NAME
   FROM SYSCAT.EVENTS
   ORDER BY TYPE

EVENT_TYPE           EVENT_MONITOR_NAME
-------------------- --------------------
ACTIVITIES           INVTACT
BUFFERPOOLS          BUFFER_INVT
CONNECTIONS          CONNECTIONS_INVT
DEADLOCKS            DEADLOCK_INVT
DETAILDEADLOCKS      DB2DETAILDEADLOCK
LOCKING              INVTLOCK
LOCKING              QUINNJN_LOC_UNF
LOCKING              UNFORM
LOCKING              RM
LOCKING              LOCK_UP_STAFF
LOCKING              INVTLOCK2
PKGCACHEBASE         CACHEEVMON
STATISTICS           INVTSTATS
STATISTICS           STAFFSTATS
TABLES               TABLE_INVTTABLE
TABLESPACES          TABLESPACES_INVT
THRESHOLDVIOLATIONS  INVTTHRESHOLD
TRANSACTIONS         TRANSAC_INVT
UOW                  INVTUOW
UOW                  UOWINVT
UOW                  STAFF_UOW

  21 record(s) selected.
```

パーティション・データベースおよび DB2 pureScale 環境のデータベースにおけるイベント・モニター

一般的に、パーティション・データベース・システムまたは DB2 pureScale® 環境のイベント・モニターは、非パーティション単一メンバー・データベースで実行されるイベント・モニターと同様に動作します。ただし、注意すべき相違点もいくつかあります。

パーティション・データベース環境

通常表および未フォーマット・イベント (UE) 表に書き込むイベント・モニター

特定のパーティション上の通常表および UE 表に書き込みを行うイベント・モニターを作成することはできません。そうではなく、パーティション・データベース環境では、イベント・モニター処理は各パーティション上で実行されます。より正確に言えば、イベント・モニター処理は、ターゲット表が存在するデータベース・パーティション・グループに属する各パーティションのメンバー上で実行されます。

イベント・モニター処理を実行するそれぞれのパーティションに、特定のイベント・モニター用の同じセットのターゲット表があります。それらの表の中のデータは、パーティションごとに異なります。ある特定のパーティショ

ンにおけるデータは、そのパーティション上で発生したイベントのみを反映するからです。表イベント・モニターの場合は、各パーティションからイベント・モニター表のデータを収集する SQL ステートメントを実行することによって、全パーティションの集約値を取得できます。 UE 表イベント・モニターの場合は、 EVMON_FORMAT_UE_TO_TABLE ストアード・プロシージャーに指定する SQL ステートメントを使用するか、または EVMON_FORMAT_UE_TO_XML 表関数を使用することによって、パーティション間でデータを集約できます。

各イベント・モニター表の最初の列は PARTITION_KEY という名前で、これは表のパーティション・キーとして使用されます。各イベント・モニター処理は、この列の値を選択し、処理を実行中のデータベース・パーティションにデータを挿入します。つまり、挿入操作は、そのイベント・モニター処理が実行されているデータベース・パーティション上でローカルに実行されます。任意のデータベース・パーティションで、PARTITION_KEY フィールドは同じ値を含みます。このため、データ・パーティションをドロップしてデータの再配分を行う場合、ドロップされたデータベース・パーティション上のすべてのデータは均等に配分されるのではなく、他の 1 つのデータベース・パーティションに移動することになります。したがって、データベース・パーティションをドロップする前に、そのデータベース・パーティション上のすべての表の行を削除することを検討してください。

また、パーティション・データベース環境では、PARTITION_NUMBER または MEMBER という名前の列が各表に定義されます。この列には、データが挿入されたパーティションまたはメンバーの番号が含まれます。

イベントは、ターゲット表用の表スペースが存在するパーティション上のイベント・モニター・ターゲット表に書き込まれます。イベント・モニターのターゲット表用の表スペースが、イベント・モニターを実行するパーティション上に存在しない場合、そのようなパーティションではデータは収集されませんが、エラーは戻されません。さらには、表スペースが存在しない場合、イベントに関するログ・レコードも書き込まれません。この動作を利用すると、ユーザーは、特定のパーティションにのみ存在する表スペースを作成することにより、モニター用にパーティションのサブセットを選択できることになります。

表書き込みイベント・モニターの活動化の際、 FIRST_CONNECT および EVMON_START に対するコントロール表の各行は、ターゲット表用の表スペースが存在するすべてのデータベース・パーティション上で挿入されます。

イベント・モニターの活動化の際にパーティションがまだアクティブでない場合は、そのパーティションが次回活動化された時にイベント・モニターは活動化されます。

ファイルおよび名前付きパイプに書き込むイベント・モニター

ファイル・イベント・モニターおよびパイプ・イベント・モニターは、1 つの例外を除き、それらのイベント・モニターが実行されているデータベース・パーティション (モニター・パーティション) 上で発生したイベントのみをキャプチャーします。このようなイベント・モニターは、*ローカル・イベント・モニター* と呼ばれます。前述の例外とは、 DEADLOCK イベント・モニターの場合です。これは、ローカル・イベント・モニターとして

も、グローバル・イベント・モニターとしても作成できます。グローバル・イベント・モニターとして作成すると、すべてのデータベース・パーティションでデッドロック情報が収集され、イベント・モニター処理を実行している特定のデータベース・パーティションに報告されます。[3]

パーティション・データベース環境でファイル・イベント・モニターまたはパイプ・イベント・モニターを作成する場合、イベント・モニターを実行するパーティションを CREATE EVENT MONITOR ステートメントの一部として指定できます。パーティション番号を省略すると、イベント・モニターの作成時に接続していたデータベース・パーティション上でイベント・モニターは実行されます。

イベント・モニターは、モニター・パーティションがアクティブである場合にのみ活動化できます。イベント・モニターを活動化するために SET EVENT MONITOR ステートメントを使用したものの、モニター・パーティションがまだアクティブではなかった場合、イベント・モニターの活動化は、モニター・パーティションが次回開始される時に行われます。また、このイベント・モニターの活動化は、明示的にイベント・モニターまたはインスタンスを非アクティブ化しない限り、自動的に行われます。例えば、以下の一連のステートメントを考慮してみましょう。

```
DB2 CONNECT TO PAYROLL
DB2 CREATE EVENT MONITOR ABC ... ON DBPARTITIONNUM 2
DB2 SET EVENT MONITOR ABC STATE 1
```

上記のステートメントを実行した後、イベント・モニター ABC は、データベース PAYROLL がデータベース・パーティション 2 でアクティブになると、必ず自動的にアクティブになります。この自動活動化は、ステートメント DB2 SET EVENT MONITOR ABC STATE 0 を実行するか、あるいはパーティション 2 が停止するまで行われます。

データベース・パーティションを追加した場合に、既存のグローバルな表イベント・モニターまたは UE 表イベント・モニターが、その新しく作成されたパーティションのデータ収集を自動的に開始することはありません。新しいパーティションに関するデータを収集して記録するには、以下のいずれかの手順を実行する必要があります。

- グローバル・イベント・モニター (つまり DEADLOCKS イベント・モニター) の場合は、イベント・モニターの再始動。
- 表または UE 表イベント・モニターの場合は、イベント・モニターのドロップ、再作成、および再始動。

DB2 pureScale 環境

DB2 pureScale 環境では、実質的に 1 つのデータ・パーティションと、データを処理する 2 つ以上のメンバーが存在します。したがって、イベント・モニターを作成すると、書き込み先がファイル、パイプ、表、または UE 表のいずれであろうと、イベント・モニター処理はすべてのメンバー上で実行されます。

3. このイベント・モニターは推奨されなくなりました。ロックおよびデッドロックのイベント情報をキャプチャーするための推奨イベント・モニターは、LOCKING イベント・モニターです。

イベント・データの報告は、メンバー単位で行われます。この結果、**total_cpu_time** モニター・エレメントのような、あるメンバーに関連するモニター・エレメントまたはメトリックは、そのメンバー固有のデータを報告します。ただし、 **tablespace_total_pages** モニター・エレメントのような、データそのものに関連する他のモニター・エレメントは、どのメンバーが報告するかにかかわらず同じ値を示します。

例

例 1: ファイル書き込みイベント・モニターをパーティション・データベース環境で作成する

以下の例は、パーティション 3 で実行され、バッファー・プール関連イベントに関するデータを収集し、その出力をファイルに書き込むイベント・モニターを作成する方法を示しています。

```
CREATE EVENT MONITOR bpmon FOR BUFFERPOOLS
                    WRITE TO FILE '/tmp/dlevents'
                    ON DBPARTITION 3
```

例 2: 表イベント・モニターをパーティション・データベース環境で作成する

以下の例は、アクティビティー関連イベントに関するデータを収集し、その出力を表に書き込む表モニターを作成する方法を示しています。

```
CREATE EVENT MONITOR myacts FOR ACTIVITIES
                    WRITE TO TABLE
```

この例では、イベント・モニターに対して論理データ・グループを指定していないため、このタイプのイベント・モニターに関連付けられているすべての論理データ・グループに関して表が作成されます。それらの表はどれも、各パーティションのデフォルトの表スペースに作成されます (各パーティションにデフォルトの表スペースが存在する場合)。各データベース・パーティション上の表に収集されるデータは、そのパーティションで発生したイベントに関連するデータです。

特定のパーティションのイベント・モニター・データを表示するには、そのパーティションを照会する SELECT ステートメントを実行します。

```
SELECT TOTAL_CPU_TIME FROM myacts WHERE PARTITION_NUMBER = 3
```

イベント・モニターのデータ収集の有効化

使用しているイベント・モニターのタイプによっては、イベント・モニターの作成後に収集を構成しなければならない場合があります。

デフォルトで、一部のイベント・モニターは、活動化するとただちに一定のデータを収集します。その他のイベント・モニターでは、イベント・モニターの作成とは関係なく、データの収集を明示的に構成する必要があります。これらのタイプのイベント・モニターは、パッシブ・イベント・モニターと呼ばれることがあります。

始める前に

すべてのイベント・モニターは、そのターゲット出力表 (通常表または UE 表)、ファイルまたはパイプにデータが書き込まれる前に活動化されている必要があります。一部のイベント・モニターは、デフォルトで AUTOSTART イベント・モニターとして構成されます。このことは、データベースが活動化されると、これらのイベント・モニターが自動的に活動化されることを意味しています。その他のイベン

ト・モニターはデフォルトで、手動による活動化を必要とするように構成されています。いずれにせよ、デフォルトの開始オプションは、オーバーライドが可能です。ただし、イベント・モニターの作成後、次にデータベースを活動化する前に自動イベント・モニターを始動するには、SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを手動で活動化する必要があります。

このタスクについて

一部のイベント・モニターは、CREATE ステートメントまたは ALTER EVENT MONITOR ステートメントに WHERE 節を使用して、イベント情報を選択的にキャプチャーする操作をサポートしています。ただし、以下のイベント・モニターは、イベント・モニターの定義には関係なく、どのイベント・データを収集するかを制御する機能を提供しています。

- アクティビティー
- 変更履歴
- ロッキング
- 統計
- 作業単位

ここに挙げたイベント・モニターの一部は、活動化後に、デフォルトで一定のタイプのデータを収集します。その他のイベント・モニターは、データの収集を明示的に有効化する必要があります。いずれの場合も、データの収集を、次に示す 2 つの方法のうち、対象となるアクティビティーの範囲に応じたいずれかの方法で有効化することができます。

データベース内のすべてのアクティビティー

データベース内のすべてのアクティビティーにわたってモニター・データが収集されるようにするには、調べようとするデータのタイプに該当する構成パラメーターを変更します。例えば、データベース内で稼働しているすべての作業単位について作業単位データを収集するには、**mon_uow_data** を BASE に設定します。場合によっては、構成パラメーターのデフォルトの設定で、日付を受け取るのに適したアクティブなイベント・モニターがある場合、一部のタイプのデータが必ず収集されるようになっています。例えば、**mon_req_metrics** のデフォルトの設定は BASE です。この設定をオーバーライドしない限り、アクティブな統計または作業単位イベント・モニターは、要求モニター・エレメントの BASE セットの値を記録します。

要確認: WHERE 述部の使用をサポートしているイベント・モニターは、関係する構成パラメーターの設定に関係なく、この述部で指定された条件を満たすデータのみを収集します。

選択したアクティビティー

一部のイベント・モニター - 特に、ワークロード管理イベント・モニター (しきい値違反、統計およびアクティビティー) - は、特定のワークロード管理オブジェクトに関するデータ収集を制御する機能を提供しています。例えば、特定のサービス・スーパークラスで稼働しているアクティビティーに関するアクティビティー情報を収集するように選択することができます。このレベルでの収集の構成では、通常、CREATE ステートメントまたは ALTER WORKLOAD (あるいは SERVICE CLASS または WORK ACTION) ステー

トメントに COLLECT 節を追加して、その WLM オブジェクトの保護下で稼働しているアクティビティーに関してどのタイプの情報を収集するかを指定することが必要です。例えば、サービス・クラス urgent に関する拡張統計情報の収集を有効にするには、次のステートメントを使用するとよいでしょう。

```
ALTER SERVICE CLASS urgent
      COLLECT AGGREGATE ACTIVITY DATA EXTENDED
```

注: WLM CREATE ステートメントまたは ALTER ステートメントで COLLECT 節が指定されている場合、その WLM オブジェクトに関しては、この節で指定された設定のほうが、構成パラメーターを使用して構成されたデータベース全体の設定よりも優先になります。例えば、**mon_req_metrics** が EXTENDED に設定されており、ワークロード payroll が BASErequest メトリックを収集するように構成されていた場合 (例えば CREATE WORKLOAD payroll COLLECT REQUEST METRICS BASE など)、データベース内の payroll ワークロード*以外の* すべてのアクティビティーについて、拡張要求メトリックが収集されます。

手順

このセクションの冒頭で示したイベント・モニターのタイプの 1 つについてデータの収集を有効にするには、以下のステップを実行します。

1. デフォルトでデータが既に収集されている場合、それが何のデータであるかを判別します。関心のあるデータを収集するのに、何らかの設定を変更しなくてもよい場合があります。
2. データを収集する対象のアクティビティーの範囲を決定します。データを収集するのはデータベース全体か。それとも特定のワークロード、サービス・クラス、または作業アクションについてのみか。
3. どのタイプのモニター・エレメントを収集するかを決定します。一部のイベント・モニターでは、要求モニター・エレメント、アクティビティー・データなど、さまざまなタイプのモニター・データの収集をサポートしています。
4. 収集するモニター・データのセット別に、各セット内で収集するデータの範囲を決定します。通常、データを収集しない (NONE)、基本データを収集する (BASE)、または拡張データを収集する (EXTENDED) という選択肢があります。設定ごとに、どのデータを収集するかを調べて決定します。
5. 前述のステップでの決定に基づいて、構成パラメーターか COLLECT 節のいずれかを使用して、データの収集を構成します。
   a. データベース全体にわたる収集を構成するには、それに適した構成パラメーターを設定します。例えば、データベース SALES でロッキング・イベント・モニターによりロック待機情報と履歴の収集を有効にするには、次のコマンドを実行します。

      ```
      UPDATE DATABASE CONFIGURATION for SALES USING mon_lockwait HISTORY
      ```

   b. 特定のワークロードについての収集を構成するには、そのワークロードに適した COLLECT 節を含めてワークロードを作成するか、そのように変更します。 例えば、MANAGERS ワークロードで 5 秒を超えて待機しているロックについてのロック待機データをステートメント履歴とともに収集するよう構成するには、次に示すようなステートメントを実行します。

```
ALTER WORKLOAD MANAGERS
  COLLECT LOCK WAIT DATA FOR LOCKS WAITING MORE THAN 5 SECONDS
    WITH HISTORY
```

次のタスク

これでイベント・モニターが作成されてアクティブになり、データの収集が有効になりました。アプリケーションまたはワークロードを実行してください。

イベント・モニター情報へのアクセス方法

使用するイベント・モニターのタイプ、および生成する出力のタイプに応じて、イベント・モニターのデータにアクセスしてデータを表示する方法には、さまざまな選択肢があります。

例えば、

- 表イベント・モニターによって生成されたデータは、SQL を使用して直接照会できます。
- パイプに書き込むイベント・モニターのデータは、生成されるとすぐに参照できます。
- ファイル・イベント・モニターのデータは、イベント・モニターを非アクティブ化した後に出力ファイルを開いて参照できます。
- また、ファイルおよびパイプのイベント・モニターのデータは、両方とも **db2evmon** コマンドを使用してデータをレポートにフォーマットすることができます。
- UE 表に書き込まれたデータを参照するには、後処理を実行する必要があります。 UE イベント・モニター・データは表または XML に変換できます。これによって、SQL または XML 照会手法を使用してデータを照会できるようになります。また、変換プロセスを経ることなく、 UE 表のデータを定様式レポートにフォーマットすることもできます。

以降のセクションでは、イベント・モニターによって生成された情報にアクセスするさまざまな方法について説明します。

通常表のイベント・モニター・データへのアクセス

通常のリレーショナル表に書き込まれたイベント・モニター・データには、SQL を使用して直接アクセスすることができます。

始める前に

データにアクセスする前に、以下の作業を行う必要があります。

- イベント・モニターの作成および活動化
- データ収集の有効化 (使用しているイベント・モニターのタイプおよび収集しようとするデータのタイプに必要な場合)
- モニタリング・データ収集の対象とするワークロードまたはアプリケーションの実行

オプションで、イベント・モニター・データをどのように使用しているかに応じて、イベント・データの調査を開始する前にデータの収集を非アクティブ化します。イベント・モニターがアクティブになったままだと、出力表にデータが書き込

まれ続けます。このため、ある照会の結果が、同じ照会を後で実行したときに取得される結果と異なる場合があります。

このタスクについて

リレーショナル表からイベント・モニター・データにアクセスするには、SQL を使用して照会を作成し、イベント・モニターによって生成された表からデータを取得する必要があります。

手順

表への書き込みを行うイベント・モニターによって生成された表から情報を取得するには、以下のようにします。

1. SELECT ステートメントを作成して、見ようとするモニター・エレメント・データを表示します。 例えば、payroll ワークロードに関するロック・データを mylocks という名前のロック・イベント・モニターから取得するよう要求するには、次のような照会を使用します。

   ```
   SELECT DISTINCT CAST(STMT_TEXT AS VARCHAR(25)) STMT, LP.PARTICIPANT_NO,
      VARCHAR(LP.APPL_NAME,10) APPL_NAME, LP.LOCK_MODE_REQUESTED,
        LP.PARTICIPANT_TYPE
   FROM LOCK_PARTICIPANT_ACTIVITIES_LOCK_MYLOCKS AS LPA
   JOIN LOCK_PARTICIPANTS_LOCK_MYLOCKS AS LP
      ON LPA.EVENT_ID = LP.EVENT_ID
      WHERE LP.WORKLOAD_NAME = 'PAYROLL'
   ```

   この例では、イベント・モニター mylocks の LOCK_PARTICIPANTS 表のデータと LOCK_PARTICIPANTS_ACTIVITIES 表の情報を結合して、以下の結果が戻されます。

2. この SQL ステートメントを実行します。

タスクの結果

```
STMT                      PARTICIPANT_NO APPL_NAME  LOCK_WAIT_VAL
------------------------- -------------- ---------- --------------------
select * from staff                    2 db2bp                         0
select * from staff                    1 db2bp                      1000

LOCK_MODE_REQUESTED  PARTICIPANT_TYPE
-------------------- ----------------
                   0 OWNER
                   1 REQUESTER


  2 record(s) selected.
```

未フォーマット・イベント表の情報へのアクセス方法

未フォーマット・イベント (UE) 表の情報にアクセスするには、さまざまな方法があります。読み取り専有のテキスト・レポートを生成できます。また、リレーショナル表または XML 内にデータを抽出することもできます。この方法の場合、SQL または pureXML® を使用してデータを照会できます。

UE 表に書き込むイベント・モニターは、イベント・データをバイナリー・フォーマットで書き込みます。このデータには、db2evmonfmt コマンドまたはこの目的で提供されているルーチンを使用してアクセスできます。

db2evmonfmt コマンドを使用すると、以下を実行できます。

- イベント ID、イベント・タイプ、時間枠、アプリケーション、ワークロード、またはサービス・クラスの各属性に基づいて、関心対象のイベントを選択します。
- テキスト・レポートまたはフォーマット済み XML 文書のどちらの形式で出力を受け取るかを選択します。
- db2evmonfmt によって提供されているものを使用する代わりに、独自の XSLT スタイル・シートを作成して、出力形式を完全に制御します。

さらに、以下のルーチンを使用して、未フォーマット・イベント表からデータを抽出することもできます。

- EVMON_FORMAT_UE_TO_XML - 未フォーマット・イベント表から XML 文書にデータを抽出します。
- EVMON_FORMAT_UE_TO_TABLES - 未フォーマット・イベント表から一連のリレーショナル表にデータを抽出します。

これらの 2 つのルーチンにより、SELECT ステートメントを使用して、未フォーマット・イベント表の抽出対象行を正確に指定できます。

イベント・モニター・データ読み取り用の db2evmonfmt ツール:

Java™ ベースの、汎用 XML パーサー・ツールである **db2evmonfmt** は、未フォーマット・イベント表を使用するイベント・モニターによって生成されたデータから、判読可能なフラット・テキスト出力 (テキスト・バージョン) またはフォーマット済み XML 出力を生成します。

指定したパラメーターに基づいて、**db2evmonfmt** ツールは、イベント・モニター・データの解析方法および作成する出力のタイプを決定します。

db2evmonfmt ツールは、Java ソース・コードとして提供されます。使用する前に、以下のステップを実行して、このツールをセットアップおよびコンパイルする必要があります。

1. sqllib/samples/java/jdbc ディレクトリーでソース・コードを見つける。
2. Java ソース・ファイルに組み込まれている説明に従って、ツールをセットアップおよびコンパイルする。

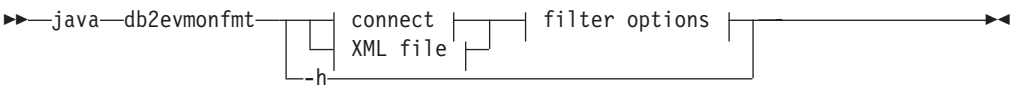
ソース・コードは、ユーザーの好みに合わせて出力を変えるために変更できます。

このツールは、XSLT スタイル・シートを使用して、イベント・データをフォーマット済みテキストにトランスフォームします。これらのスタイル・シートについては理解する必要はありません。ツールは、イベント・モニター・タイプに基づいて正しいスタイル・シートを自動的にロードし、イベント・データをトランスフォームします。各イベント・モニターは、sqllib/samples/xml/data ディレクトリー内でデフォルトのスタイル・シートを提供します。さらにツールは、以下のフィルター・オプションを提供します。

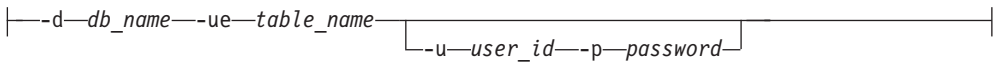
- イベント ID
- イベント・タイム・スタンプ
- イベント・タイプ
- ワークロード名
- サービス・クラス名

- アプリケーション名

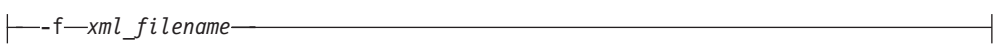
ツールの構文



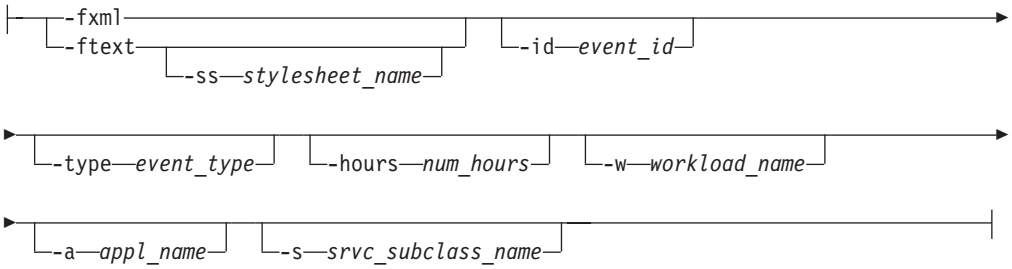
connect:



XML file:



filter options:



ツールのパラメーター

java
db2evmonfmt Java ベース・ツールを正常に実行するには、**java** キーワードをツール名の先頭に付ける必要があります。このツールを正常に実行するための正しい Java バージョンは、DB2 製品のインストール中に sqllib/java/jdk64 ディレクトリーからインストールされます。

-d *db_name*
接続先のデータベース名を指定します。

-ue *table_name*
未フォーマット・イベント表の名前を指定します。

-u *user_id*
ユーザー ID を指定します。

-p *password*
パスワードを指定します。

-f *xml_filename*
フォーマットする入力 XML ファイルの名前を指定します。

-fxml
フォーマット済み XML 文書を生成します (STDOUT にパイプ)。

-ftext
XML 文書をフォーマットしてテキスト文書にします (STDOUT にパイプ)。

-ss *stylesheet_name*
XML 文書のトランスフォームに使用する XSLT スタイル・シートを指定します。

-id *event_id*
指定したイベント ID に一致するすべてのイベントを表示します。

-type *event_type*
指定したイベント・タイプに一致するすべてのイベントを表示します。

-hours *num_hours*
指定した直近の数時間内に発生したすべてのイベントを表示します。

-w *workload_name*
指定したワークロードの一部であるすべてのイベントを表示します。

-a *appl_name*
指定したアプリケーションの一部であるすべてのイベントを表示します。

-s *srvc_subclass_name*
指定したサービス・サブクラスの一部であるすべてのイベントを表示します。

XSLT スタイル・シート

DB2 データベース・マネージャーは、デフォルトの XSLT スタイル・シートを提供しています (表 1 を参照)。これは `sqllib/samples/java/jdbc` ディレクトリー内にあります。これらのスタイル・シートは、必要とする出力を生成するために変更できます。

表 17. イベント・モニター用のデフォルトの XSLT スタイル・シート

イベント・モニター	デフォルトの XSLT スタイル・シート
ロッキング	DB2EvmonLocking.xsl
作業単位	DB2EvmonUOW.xsl
パッケージ・キャッシュ	DB2EvmonPkgCache.xsl

XML 文書をトランスフォームするための独自の XSLT スタイル・シートを作成できます。これらのスタイル・シートは、`-ss stylesheet_name` オプションを使用して Java ベース・ツールに渡すことができます。

例

例 1 直近の 32 時間内に発生したすべてのイベントについてのフォーマット済みテキスト出力を、データベース SAMPLE にあるパッケージ・キャッシュの未フォーマット・イベント表 PKGから得るには、以下のコマンドを発行します。

```
java db2evmonfmt -d sample -ue pkg -ftext -hours 32
```

例 2 直近の 24 時間内に発生した LOCKTIMEOUT タイプのすべてのイベントについてのフォーマット済みテキスト出力を、データベース SAMPLE にある未フォーマット・イベント表 LOCK から得るには、以下のコマンドを発行します。

```
java db2evmonfmt -d sample -ue LOCK -ftext -hours 24 -type locktimeout
```

例 3 直近の 5 時間内で、イベント・タイプ LOCKWAIT と一致するすべてのイベントを抽出するために、XML ソース・ファイル LOCK.XML からフォーマット済みテキスト出力を得るには、以下のコマンドを発行します。

```
java db2evmonfmt -f lock.xml -ftext -type lockwait -hours 5
```

例 4 データベース SAMPLE 内の未フォーマット・イベント表 UOW にあるすべてのイベントについて、作成済みの XSLT スタイル・シート SUMMARY.XSL を使用してフォーマット済みテキスト出力を得るには、以下のコマンドを発行します。

```
java db2evmonfmt -d sample -ue uow -ftext -ss summary.xsl
```

サンプルのフォーマット済みフラット・テキスト出力

ロック・イベント・モニターの XSLT スタイル・シートから生成されたフォーマット済みフラット・テキスト出力のサンプルを以下に示します。

```
------------------------------------------------
Event Entry     : 0
Event ID        : 1
Event Type      : Locktimeout
Event Timestamp : 2008-05-23-12.00.14.132329000
------------------------------------------------

Lock Details
------------
Lock Name       : 02000401000000000000000054
Lock Type       : Table
Lock Attributes : 00000000
Lock Count      : 1
Lock Hold Count : 0
Lock rrIID      : 0
Lock Status     : Waiting
Cursor Bitmap   : 00000000
Tablespace Name : USERSPACE1
Table Name      : NEWTON  .SARAH

Attributes           Requestor                      Holder
----------           ---------                      ------
Application Handle   [0-35]                         [0-16]
Application ID       *LOCAL.horton.080523160016     *LOCAL.horton.080523155938
Application Name     xaplus0001                      db2bp
Authentication ID    NEWTON                         HORTON
Requesting Agent     65                             21
Coordinating Agent   65                             21
Application Status   SQLM_CONNECTPEND               SQLM_CONNECTPEND
Lock Timeout         5000                           0
Workload Name        XAPLUS0010_WL02                SYSDEFAULTUSERWORKLOAD
Service Subclass     XAPLUS0010_SC02                SYSDEFAULTSUBCLASS
Current Request      Execute                        Execute Immediate
Lock Mode            Intent Exclusive               Exclusive
tpmon Userid
tpmon Wkstn
tpmon App
tpmon Accstring

Lock Requestor Current Activities
---------------------------------
Activity ID     : 2
Uow ID          : 1
Package ID      : 65426E4D4B584659
Package SectNo  : 3
```

```
Package Name    : NEWTON
Package Schema  : AKINTERF
Package Version :
Reopt           : always
Eff Isolation   : Cursor Stability
Eff Locktimeout : 5
Eff Degree      : 0
Nesting Level   : 0
Stmt Unicode    : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Insert/Update/Delete
Stmt Text       : INSERT INTO SARAH VALUES(:H00008, :H00013, :H00014)


Lock Requestor Past Activities
------------------------------
Activity ID     : 1
Uow ID          : 1
Package ID      : 65426E4D4B584659
Package SectNo  : 2
Package Name    : NEWTON
Package Schema  : AKINTERF
Package Version :
Reopt           : always
Eff Isolation   : Cursor Stability
Eff Locktimeout : 5
Eff Degree      : 0
Nesting Level   : 0
Stmt Unicode    : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Insert/Update/Delete
Stmt Text       : INSERT INTO NADIA VALUES(:H00007)


Lock Holder Current Activities
------------------------------

Lock Holder Past Activities
------------------------------
Activity ID     : 1
Uow ID          : 2
Package ID      : 41414141414E4758
Package SectNo  : 201
Package Name    : NULLID
Package Schema  : SQLC2G13
Package Version :
Reopt           : none
Eff Isolation   : Cursor Stability
Eff Locktimeout : 5
Eff Degree      : 0
Nesting Level   : 0
Stmt Unicode    : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Select (blockable)
Stmt Text       : select * from newton.sarah

Activity ID     : 2
Uow ID          : 2
Package ID      : 41414141414E4758
Package SectNo  : 203
Package Name    : NULLID
Package Schema  : SQLC2G13
Package Version :
Reopt           : none
Eff Isolation   : Cursor Stability
Eff Locktimeout : 5
Eff Degree      : 0
```

```
Nesting Level   : 0
Stmt Unicode    : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Lock Table
Stmt Text       : lock table newton.sarah in exclusive mode


-----------------------------------------------
Event Entry     : 1
Event ID        : 2
Event Type      : Locktimeout
Event Timestamp : 2008-05-23-12.04.42.144896000
-----------------------------------------------

...
...
...
```

使用上の注意

db2evmonfmt ユーティリティーは、Java ベース・ツールであり、正常に実行するには **java** キーワードを先頭に付ける必要があります。必要な Java バージョンは、DB2 製品と共に sqllib/java/jdk64 ディレクトリーからインストールされたものです。

注: さらに、EVMON_FORMAT_UE_TO_XML 表関数を使用して、未フォーマット・イベント表の BLOB 列に含まれているバイナリー・イベントを XML 文書にフォーマットすることもできます。

未フォーマット・イベント表からデータを抽出するルーチン:

未フォーマット・イベント (UE) 表に書き込むイベント・モニターによって収集されたデータに対して照会を実行するためには、まずデータを UE 表から抽出する必要があります。抽出には、この目的のために用意されている 2 つのルーチンのどちらかを使用します。

EVMON_FORMAT_UE_TO_TABLES プロシージャーは、 UE 表からデータを抽出してリレーショナル表を作成します。 EVMON_FORMAT_UE_TO_XML 表関数は、XML 文書を作成します。

EVMON_FORMAT_UE_TO_TABLES

EVMON_FORMAT_UE_TO_TABLES プロシージャーは、イベント・モニターが生成した UE 表を参照し、表の中のデータを、照会可能なリレーショナル表に抽出します。作成される表の数は、イベント・モニターのタイプ、およびイベント・モニターがデータを収集する論理データ・グループによって異なります。一般的には、論理データ・グループごとに別の表にデータは書き込まれます。例えば、パッケージ・キャッシュ・イベント・モニターは、 3 つの論理データ・グループ (pkgcache と pkgcache_metrics、および pkgcache_stmt_args) からイベント・データを収集します。したがって、EVMON_FORMAT_UE_TO_TABLES は 3 つの表を作成します。

注: EVMON_FORMAT_UE_TO_TABLES は、 CONTROL 論理データ・グループについては表を作成しません。

UE 表からリレーショナル表を作成するだけでなく、バージョン 10.1 以降、 EVMON_FORMAT_UE_TO_TABLES プロシージャーは、UE 表からデ

ータを除去する機能も備えています。 PRUNE_UE_TABLES オプションを EVMON_FORMAT_UE_TO_TABLES に使用すると、リレーショナル表に正常に挿入されたデータは、未フォーマット・イベント (UE) 表から削除されます。

EVMON_FORMAT_UE_TO_XML

EVMON_FORMAT_UE_TO_XML 表関数は、イベント・モニターが生成した UE 表を参照し、表の中のデータを XML 文書に抽出します。この場合、 pureXML を使用して、この文書を必要に応じて何度でも照会することができます。

注:

- この表関数の機能は、**db2evmonfmt** ユーティリティーに -fxml オプションを指定して使用する場合と似ています。
  EVMON_FORMAT_UE_TO_XML を **db2evmonfmt** の代わりに使用する場合、次のような相違点があります。
  - EVMON_FORMAT_UE_TO_XML は表関数です。そのため、 SQL ステートメントの一部として呼び出されます。**db2evmonfmt** は、個別のユーティリティーとして実行されます。
  - EVMON_FORMAT_UE_TO_XML の場合、SELECT ステートメントに WHERE 節を指定して、UE 表のイベントをフィルタリングできます。**db2evmonfmt** の場合、イベント・データのフィルタリングについては限られた機能しかありません。
- EVMON_FORMAT_UE_TO_XML の出力 XML 文書を、 **db2evmonfmt** によってフォーマットしてフラット・テキスト・ファイルを作成することができます。

どちらのルーチンも、ルーチンの呼び出しには SELECT ステートメントを含めて、どのデータを抽出するか条件を指定する必要があります。

UE 表からのデータの整理:

EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用して UE 表からデータを抽出する場合は、PRUNE_UE_TABLE オプションを使用して、必要なくなったデータを削除することができます。

始める前に

UE 表からデータを抽出できるようにするには、その前に、UE 表に書き込むイベント・モニターを作成し、活動化して、データの収集を有効にしておく必要があります。

このタスクについて

UE 表はパフォーマンス上の利点があるだけでなく、イベント・モニターの出力として UE 表を使用すると、 EVMON_FORMAT_UE_TO_TABLES プロシージャーの自動整理フィーチャーを活用できます。 . このプロシージャーを使用すると、UE 表から抽出されて通常表に書き込まれたデータを、UE 表から自動的に削除することができます。このプロシージャーにより、UE 表の管理が簡単になります。例えば、作業単位イベント・モニターを使用して、会計目的 (アプリケーションまたは

照会で使用された CPU 時間に対する部門への課金など) で日次レポートを生成するための情報をキャプチャーする必要があるとします。この場合、レポートを生成した後にデータを整理することが必要な場合があります。

手順

UE 表からデータを抽出して整理するには、以下のようにします。

PRUNE_UE_TABLE オプションを指定して EVMON_FORMAT_UE_TO_TABLES プロシージャーを呼び出す SQL ステートメントを実行して、データを通常表に抽出します。 例えば、TRACKWORK という作業単位イベント・モニターがある場合、次のようなステートメントを作成できます。

```
CALL EVMON_FORMAT_UE_TO_TABLES
   ('UOW', NULL, NULL, NULL, NULL, NULL, 'PRUNE_UE_TABLE', -1,
      'SELECT * FROM TRACKWORK')
```

すべてのイベント・データが UE 表から UOW_EVENT_TRACKWORK 表および UOW_METRICS_TRACKWORK 表にコピーされます。さらに、コピーされたレコードは、すべて UE 表から削除されます。

コマンド行からのファイルまたはパイプ・イベント・モニター出力のフォーマット

ファイルまたはパイプ・イベント・モニターの出力はバイナリー・ストリームの論理データ・グループです。 **db2evmon** コマンドを使用することによって、このデータ・ストリームをコマンド行からフォーマットすることができます。

この生産性向上ツールは、イベント・モニターのファイルまたはパイプからイベント・レコードを読み込んだ後、それらを画面に書き出します (標準出力)。

始める前に

データベースに接続しているのではない限り、権限は不要です。データベースに接続している場合には、以下のいずれかの権限が必要です。

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM

このタスクについて

イベント・ファイルのパスを提供するか、またはデータベースおよびイベント・モニターの名前を提供することによって、どのイベント・モニターの出力をフォーマットするのかを指定することができます。

手順

イベント・モニター出力をフォーマットするには、次のようにします。

- イベント・モニター・ファイルが格納されているディレクトリーを指定する。

  ```
  db2evmon -path '/tmp/dlevents'
  ```

  /tmp/dlevents は (UNIX) パスです。

- データベースおよびイベント・モニター名を指定する。

  ```
  db2evmon -db 'sample' -evm 'dlmon'
  ```

  sample は、イベント・モニターが属するデータベースを示します。

  dlmon はイベント・モニターを示します。

イベント・モニターの変更

イベント・モニターを変更することはできませんが、例外が 1 つだけあります。イベント・モニターで収集する論理データ・グループのセットに 1 つ以上の論理データ・グループを追加することができます。論理グループを追加するには、ALTER EVENT MONITOR ステートメントを使用します。

このタスクについて

表に書き込むイベント・モニターを作成すると、デフォルトでは、そのイベント・モニターに関連付けられているすべての論理データ・グループのモニター・エレメントがキャプチャーされます。ただし、CREATE EVENT MONITOR ステートメントに論理データ・グループの名前を含めた場合は、それらのグループのみがキャプチャーされます。例えば、以下の例に示すように、event_activity および event_activity_metrics 論理データ・グループのデータのみをキャプチャーするアクティビティー・イベント・モニターを作成することができます。

```
CREATE EVENT MONITOR myacts FOR ACTIVITIES
   WRITE TO TABLE
   event_activity, event_activity_metrics
```

上記の DDL ステートメントは、ACTIVITY_myacts および ACTIVITY_METRICS_myacts という 2 つの表に書き込みを行うイベント・モニターを作成します。

制約事項

ALTER EVENT MONITOR ステートメントは、イベント・モニターに論理データ・グループを追加する場合にのみ使用できます。論理データ・グループは除去できません。また、データ・グループに属するモニター・エレメントのデータをキャプチャーするために使用する表に関連付けられている、名前、ターゲット表スペース、および PCTDEACTIVATE 値を変更することはできません。

手順

他の論理データ・グループをイベント・モニターに追加するには、以下のようにします。

1. 追加する論理データ・グループを決定します。 前述の例の、2 つの論理データ・グループのみをキャプチャーするロック・イベント・モニターに、event_activitystmt および event_activityvals 論理データ・グループを追加するとします。
2. これらの新しい論理データ・グループを追加するための ALTER EVENT MONITOR ステートメントを用意します。

   ```
   ALTER EVENT MONITOR mylacts
      ADD LOGICAL GROUP event_activitystmt
      ADD LOGICAL GROUP event_activityvals
   ```

3. ステートメントを実行します。

タスクの結果

この ALTER EVENT MONITOR ステートメントの実行が完了すると、イベント・モニター myacts に以下の 2 つの表が追加で作成されます。

ACTIVITYSTMT_myacts

ACTIVITYVALS_myacts

次にイベント・モニターが活動化されたときに、これらの表には、対応する論理データ・グループのデータが追加されます。

要確認: 新しい論理データ・グループをイベント・モニターに追加した場合、新たに追加された論理グループの表には、もともと表に含まれていた論理データ・グループの既存データに対応する行はありません。論理グループを追加した後は、必要に応じて照会を調整するか、表の古いデータを整理することを検討してください。

例

データベース管理者は、次の SQL ステートメントを使用して、mylocks というロッキング・イベント・モニターを作成します。

```
CREATE EVENT MONITOR mylocks FOR LOCKING WRITE TO TABLE LOCK, LOCK_PARTICIPANTS
```

このステートメントは、lock 論理データ・グループおよび lock_participants 論理データ・グループのモニター・エレメントに関する情報を収集します。モニター・エレメント・データが書き込まれる先の表は、デフォルトの表名である LOCK_MYLOCKS および LOCK_PARTICIPANTS_MYLOCKS で作成されます。

その後、データベース管理者は LOCK_PARTICIPANT_ACTIVITIES 論理データ・グループ内の情報を収集することを決定します。データベース管理者は次のステートメントを使用して、イベント・モニターを変更します。

```
ALTER EVENT MONITOR mylocks ADD LOGICAL GROUP LOCK_PARTICIPANT_ACTIVITIES
```

このステートメントにより、lock_participant_activities のモニター・エレメントが、既に収集されているその他のエレメントに加えて収集されます。この新しいセットのモニター・エレメントは、表 LOCK_PARTICIPANT_ACTIVITIES_MYLOCKS に書き込まれます。

その後、データベース管理者が CONTROL 論理データ・グループのデータも必要だと判断したとします。しかし、このデータはデフォルト以外の名前の表、およびデフォルトの表スペース以外の表スペースに書き込むことが望ましいと考えます。そこで、次のステートメントを使用します。

```
ALTER EVENT MONITOR mylocks ADD LOGICAL GROUP CONTROL TABLE ctl_mylocks IN mytbsp3
```

このステートメントは、制御論理データ・グループをイベント・モニターの出力に追加します。このステートメントは、制御論理データ・グループをイベント・モニターの出力に追加します。データは表 CTL_MYLOCKS に書き込まれます。この表は、デフォルトの表スペースではなく表スペース mytbsp3 に書き込まれます。

さまざまなイベント・タイプのモニター

ロック・イベントおよびデッドロック・イベントのモニター

大規模な DB2 環境内でのロック競合状態の診断および修正は、複雑で時間がかかる作業となる場合があります。ロック・イベント・モニターは、ロック・データを収集してこの作業を単純化するように設計されています。

注: デッドロック・イベント・モニターは推奨されなくなり、このイベント・モニターが提供していた機能は、ロック・イベント・モニターに統合されました。また、DB2DETAILDEADLOCK イベント・モニターも推奨されなくなりました。186 ページの『推奨されないロック・モニター機能』を参照して、このイベント・モニターの使用に関する重要な情報を確認してください。

ロック・イベント・モニターは、ロック・イベントの発生時に、それらのロック・イベントに関する説明情報をキャプチャーするために使用します。キャプチャーされた情報により、ロック・イベントの原因となっているロック競合に関係した主要なアプリケーションを識別できます。ロック・リクエスター (デッドロックまたはロック・タイムアウト・エラーを受け取ったアプリケーション、または指定された期間を超えてロックを待機したアプリケーション)、および現在のロック所有者の両方についての情報がキャプチャーされます。

ロック・イベント・モニターにより収集された情報は、データベース内の未フォーマット・イベント表にバイナリー・フォーマットで書き込めます。この場合、キャプチャーされたデータは、キャプチャーの後工程で処理する必要があります。代わりに、通常の表のセットにロック・イベント情報を書き込むこともできます。最も適切な出力形式を選択する方法の詳細については、 44 ページの『イベント・モニターの出力オプション』を参照してください。

動的または静的 SQL のいずれかを使用して、ロック・イベント情報を収集するために、DB2 リレーショナル・モニター・インターフェース (表関数) に直接アクセスすることもできます。

デッドロックまたはロック・タイムアウトが発生しているかどうかの判別も、単純化されています。これらのイベントのいずれかが発生している場合、メッセージが管理通知ログに書き込まれます。これはアプリケーションに返される SQL0911N (sqlcode -911) エラーの補足となります。加えて、ロック・エスカレーションの通知も管理通知ログに書き込まれます。この情報は、ロック表のサイズおよびアプリケーションが使用できる表の量の調整に役立てることができます。ロック・タイムアウト (**lock_timeouts**)、ロック待機 (**lock_waits**)、およびデッドロック (**deadlocks**) のカウンターもあり、これらを調べることもできます。

ロック・データのキャプチャーの対象にできるアクティビティーのタイプには、以下があります。

- SQL ステートメント。例えば以下のようなものがあります。
  - DML
  - DDL
  - CALL
- **LOAD** コマンド

- **REORG** コマンド
- **BACKUP DATABASE** コマンド
- ユーティリティー要求

ロック・イベント・モニターは、非推奨のデッドロック・イベント・モニター (CREATE EVENT MONITOR FOR DEADLOCKS ステートメントおよび DB2DETAILDEADLOCK)、および非推奨のロック・タイムアウト・レポート機能 (DB2_CAPTURE_LOCKTIMEOUT レジストリー変数) を、ロック・イベント・データ収集用の単純化されて一貫性があるインターフェースで置き換え、ロック待機に関するデータをキャプチャーする機能を追加します。

機能の概要

ロック・イベント・モニターを使用してロック・イベント・データをキャプチャーできるようにするには、以下の 2 つのステップが必要です。

1. CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して LOCK EVENT モニターを作成する必要があります。モニターの名前を入力する他、出力形式として UE 表を使用する場合は、ロック・イベント・データが書き込まれる未フォーマット・イベント表の名前も入力します。

   注: イベント・モニターの出力に通常の表を使用するように選択した場合は、デフォルトの表名が割り当てられます。このデフォルトは、必要があれば、CREATE EVENT MONITOR ステートメントを使用してオーバーライドすることができます。
2. 以下のいずれかの方式を使用して、ロック・イベント・データをキャプチャーするレベルを指定する必要があります。
   - 既存のワークロードを変更するか、CREATE または ALTER WORKLOAD ステートメントを使用して新規ワークロードを作成することにより、特定のワークロードを指定することができます。ワークロード・レベルで、キャプチャーするロック・イベント・データのタイプ (デッドロック、ロック・タイムアウト、またはロック待機)、およびロックに関係したアプリケーションの SQL ステートメント履歴および入力値を必要とするかどうかを指定しなければなりません。ロック待機の場合、アプリケーションがロックを待機する期間 (その後ロック待機についてデータがキャプチャーされる) を指定することも必要です。
   - 以下の適切なデータベース構成パラメーターを設定することで、データをデータベース・レベルで収集し、すべての DB2 ワークロードに影響を与えることができます。

     mon_lockwait
     このパラメーターは、ロック待機イベントの生成を制御します。

     ベスト・プラクティスは、ロック待機データ収集をワークロード・レベルで有効にすることです。

     mon_locktimeout
     このパラメーターは、ロック・タイムアウト・イベントの生成を制御します。

ベスト・プラクティスは、ロック・タイムアウト・データ収集がアプリケーションで予期されていない場合には、それをデータベース・レベルで有効にすることです。予期されている場合には、ワークロード・レベルで有効にします。

mon_deadlock
このパラメーターは、デッドロック・イベントの生成を制御します。

ベスト・プラクティスは、デッドロック・データ収集をデータベース・レベルで有効にすることです。

mon_lw_thresh
このパラメーターは、**mon_lockwait** のイベントが生成されるまでのロック待機時間を制御します。

SQL ステートメント履歴および入力値のキャプチャーにより、プロセッサー時間、メモリー、およびストレージの使用量が増大しますが、ロック問題を適切にデバッグするには、多くの場合このレベルの詳細情報が必要です。

ロック・イベントが発生した後に、イベント・モニターによって生成された出力内で、そのイベントのデータを参照できます。 UE 表を使用した場合、その未フォーマット・イベント表内のバイナリー・データは、 **db2evmonfmt** と呼ばれる付属の Java ベース・アプリケーションを使用して XML またはテキスト文書に変換できます。さらに、未フォーマット・イベント表の BLOB 列内のバイナリー・イベント・データは、EVMON_FORMAT_UE_TO_XML 表関数を使用して XML レポート文書にフォーマットするか、または EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用してリレーショナル表にフォーマットするかのいずれかが可能です。

出力形式として通常の表を使用した場合は、 SQL を使用して直接データを照会できます。

ロック・イベントについてモニター対象とするワークロードを判別するための助けとして、管理通知ログを調べることができます。デッドロックまたはロック・タイムアウトが検出されるたびに、メッセージがそのログに書き込まれます。これらのメッセージでは、ロック・リクエスターまたはロック所有者 (複数の場合あり) が実行しているワークロード、およびロック・イベントのタイプが示されます。ロック・タイムアウト (**lock_timeouts**)、ロック待機 (**lock_waits**)、およびデッドロック (**deadlocks**) のワークロード・レベルのカウンターもあり、これらを調べることもできます。

ロック・イベントについて収集される情報

ロック・イベント・モニターにより収集されるロック・イベントの情報のいくつかには、以下があります。

- イベントの原因となったロック
- ロック・イベントの原因となったロックを保持しているアプリケーション
- ロック・イベントの原因となるロックを待機または要求していたアプリケーション
- ロック・イベント時のアプリケーションの実行内容。

推奨されないロック・モニター機能

デフォルトでは、各データベースで、推奨されない詳細なデッドロック・イベント・モニター DB2DETAILDEADLOCK が作成され、データベースがアクティブになるときに開始されます。ロック・イベント・モニターを使用してデッドロックを検出する場合は、 DB2DETAILDEADLOCK イベント・モニターを無効にすることを検討してください。DB2DETAILDEADLOCK イベント・モニターがアクティブなまま、ロック・イベント・モニターでもデッドロック情報を収集すると、両方のイベント・モニターがデータを収集することになり、パフォーマンスに重大な影響を与える可能性があります。

DB2DETAILDEADLOCK イベント・モニターを除去するには、以下の SQL ステートメントを発行します。

```
SET EVENT MONITOR DB2DETAILDEADLOCK state 0
DROP EVENT MONITOR DB2DETAILDEADLOCK
```

ロック・イベント・モニターによって生成されるデータ

ロック・イベント・モニターは、システム内のロックおよびデッドロックに関するデータを生成します。ロック・イベント・モニターの出力先を通常の表にするか、未フォーマット・イベント (UE) 表にするか選択できます。

UE 表にデータを書き込む場合、データを表示するにはデータに後処理を実行する必要があります。

選択した出力形式にかかわらず、ロック・イベント・データは、次の 4 つの論理グループのいずれかから取得されます。

- lock
- lock_participants
- lock_participant_activities
- lock_activity_values

ロック・イベント・データを通常の表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

注: デフォルトでは、ロック・イベント・モニターではデッドロック情報のみが生成されます。他のタイプのロック・データを生成するには、そのデータ収集を明示的に有効にする必要があります。

ロック・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、ロック・イベント・モニターによって書き込まれる情報。

ロック・イベント・モニターの出力タイプとして WRITE TO TABLE を選択した場合、デフォルトでは、5 つの表が生成されます。各表には、1 つ以上の論理データ・グループのモニター・エレメントが入っています。

表 18. 表書き込みロック・イベント・モニターによって生成される表

デフォルトの表名	レポートされる論理データ・グループ
LOCK_*evmon-name*	lock
LOCK_PARTICIPANTS_*evmon-name*	lock_participants
LOCK_PARTICIPANT_ACTIVITIES_*evmon-name*	lock_participant_activities
LOCK_ACTIVITY_VALUES_*evmon-name*	lock_activity_values
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

重要: 5 つの表はすべてデフォルトで生成されますが、収集対象のロック情報の種類について、データ収集が有効になっているか確認する必要があります。有効になっていない場合、一部の列には NULL 値が入ります。

イベント・モニターの出力を特定の表に制限する場合は、 CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで、表を作成する論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 19. ロック・イベント・モニターに戻される情報 : デフォルトの表名: LOCK_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
DEADLOCK_TYPE	VARCHAR(10)	978 ページの『deadlock_type - デッドロック・タイプのモニター・エレメント』
DL_CONNS	INTEGER	dl_conns デッドロックに関係している接続
EVENT_ID	BIGINT NOT NULL	event_id - イベント ID モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP NOT NULL	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
EVENT_TYPE	VARCHAR(128) NOT NULL	event_type - イベント・タイプ・モニター・エレメント
MEMBER	SMALLINT NOT NULL	member - データベース・メンバー
ROLLED_BACK_PARTICIPANT_NO	INTEGER	rolled_back_participant_no ロールバック参加アプリケーション

表 20. ロック・イベント・モニターに戻される情報 : デフォルトの表名: *LOCK_PARTICIPANTS_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
AGENT_STATUS	INTEGER	agent_status DCS アプリケーション・エージェント
AGENT_TID	BIGINT	828 ページの『agent_tid - エージェント・スレッド ID モニター・エレメント』
APPL_ACTION	VARCHAR(64)	840 ページの『appl_action - アプリケーション・アクションのモニター・エレメント』
APPL_ID	VARCHAR(128)	appl_id - アプリケーション ID
APPL_NAME	VARCHAR(128)	appl_name アプリケーション名
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル
AUTH_ID	VARCHAR(128)	auth_id 許可 ID
CLIENT_ACCTNG	VARCHAR(255)	client_acctng - クライアント・アカウンティング・ストリング
CLIENT_APPLNAME	VARCHAR(255)	client_applname - クライアント・アプリケーション名
CLIENT_USERID	VARCHAR(255)	client_userid - クライアントのユーザー ID
CLIENT_WRKSTNNAME	VARCHAR(255)	client_wrkstnname - クライアント・ワークステーション名
COORD_AGENT_TID	BIGINT	936 ページの『coord_agent_tid - コーディネーター・エージェントのエンジン・ディスパッチ可能単位 ID のモニター・エレメント』
CURRENT_REQUEST	VARCHAR(32)	958 ページの『current_request - 現在の操作要求のモニター・エレメント』
DEADLOCK_MEMBER	SMALLINT	977 ページの『deadlock_member - デッドロック・メンバー : モニター・エレメント』
EVENT_ID	BIGINT	event_id - イベント ID モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
EVENT_TYPE	VARCHAR(128)	event_type - イベント・タイプ・モニター・エレメント
INTERNAL_DATA	VARCHAR(255)	
LOCK_ATTRIBUTES	CHAR(8)	lock_attributes ロック属性
LOCK_COUNT	BIGINT	lock_count ロック・カウント
LOCK_CURRENT_MODE	BIGINT	lock_current_mode - 変換前の元のロック・モード
LOCK_ESCALATION	CHAR(3)	lock_escalation ロック・エスカレーション

表 20. ロック・イベント・モニターに戻される情報 : デフォルトの表名: *LOCK_PARTICIPANTS_evmon-name* (続き)

列名	データ・タイプ	説明
LOCK_HOLD_COUNT	BIGINT	lock_hold_count ロック保留カウント
LOCK_MODE	BIGINT	lock_mode ロック・モード
LOCK_MODE_REQUESTED	BIGINT	lock_mode_requested 要求されているロック・モード
LOCK_NAME	CHAR(32)	lock_name ロック名
LOCK_OBJECT_TYPE	BIGINT	lock_object_type 待機中のロック対象タイプ
LOCK_OBJECT_TYPE_ID	CHAR(1)	将来の利用のために予約済み
LOCK_RELEASE_FLAGS	CHAR(8)	lock_release_flags ロック保留解除フラグ
LOCK_RRIID	BIGINT	
LOCK_STATUS	BIGINT	lock_status ロック状況
LOCK_TIMEOUT_VAL	BIGINT	lock_timeout_val ロック・タイムアウト値
LOCK_WAIT_END_TIME	TIMESTAMP	1153 ページの『lock_wait_end_time - ロック待機終了タイム・スタンプ : モニター・エレメント』
LOCK_WAIT_START_TIME	TIMESTAMP	lock_wait_start_time - ロック待機開始タイム・スタンプ
LOCK_WAIT_VAL	BIGINT	1159 ページの『lock_wait_val - ロック待機値のモニター・エレメント』
MEMBER	SMALLINT	member - データベース・メンバー
OBJECT_REQUESTED	VARCHAR(10)	1232 ページの『object_requested - 要求されたオブジェクトのモニター・エレメント』
PARTICIPANT_NO	INTEGER	participant_no デッドロック内の参加者
PARTICIPANT_NO_HOLDING_LK	INTEGER	participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者
PARTICIPANT_TYPE	VARCHAR(10)	1259 ページの『participant_type - 参加者タイプのモニター・エレメント』
PAST_ACTIVITIES_WRAPPED	CHAR(3)	1261 ページの『past_activities_wrapped - 過去のアクティビティー・リストの折り返しモニター・エレメント』
QUEUE_START_TIME	TIMESTAMP	1435 ページの『queue_start_time - キュー開始タイム・スタンプのモニター・エレメント』
QUEUED_AGENTS	BIGINT	1437 ページの『queued_agents - キューに入れられているしきい値エージェントのモニター・エレメント』
SERVICE_CLASS_ID	INTEGER	service_class_id サービス・クラス ID
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name サービス・サブクラス名
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name サービス・スーパークラス名

表 20. ロック・イベント・モニターに戻される情報 : デフォルトの表名: *LOCK_PARTICIPANTS_evmon-name (続き)*

列名	データ・タイプ	説明
TABLE_FILE_ID	BIGINT	table_file_id 表ファイル ID
TABLE_NAME	VARCHAR(128)	table_name 表名
TABLE_SCHEMA	VARCHAR(128)	table_schema 表スキーマ名
TABLESPACE_NAME	VARCHAR(128)	tablespace_name 表スペース名
THRESHOLD_ID	INTEGER	1598 ページの『thresholdid しきい値 ID : モニター・エレメント』
THRESHOLD_NAME	VARCHAR(128)	threshold_name しきい値名
UTILITY_INVOCATION_ID	VARCHAR(32) FOR BIT DATA	
WORKLOAD_ID	INTEGER	workload_id ワークロード ID
WORKLOAD_NAME	VARCHAR(128)	workload_name ワークロード名
XID	VARCHAR(140)	xid トランザクション ID

表 21. ロック・イベント・モニターに戻される情報 : デフォルトの表名: *LOCK_PARTICIPANT_ACTIVITIES_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACTIVITY_ID	BIGINT	activity_id アクティビティー ID
ACTIVITY_TYPE	VARCHAR(10)	activity_type アクティビティー・タイプ
CONSISTENCY_TOKEN	CHARACTER(8)	consistency_token パッケージ整合性トークン
EFFECTIVE_ISOLATION	CHARACTER(2)	effective_isolation - 有効な分離
EFFECTIVE_QUERY_DEGREE	BIGINT	effective_query_degree - 有効な照会の度合い
EVENT_ID	BIGINT	event_id - イベント ID モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
EVENT_TYPE	VARCHAR(128)	event_type - イベント・タイプ・モニター・エレメント
INCREMENTAL_BIND	CHARACTER(3)	1097 ページの『incremental_bind - 追加バインドのモニター・エレメント』
MEMBER	SMALLINT	member - データベース・メンバー
PACKAGE_NAME	VARCHAR(128)	package_name パッケージ名
PACKAGE_SCHEMA	VARCHAR(128)	package_schema - パッケージ・スキーマ
PACKAGE_VERSION_ID	VARCHAR(64)	package_version_id パッケージ・バージョン
PARTICIPANT_NO	SMALLINT	participant_no デッドロック内の参加者
QUERY_ACTUAL_DEGREE	INTEGER	query_actual_degree - パーティション内並列処理の実際の実行時の多重度
REOPT	VARCHAR(10)	1447 ページの『reopt - REOPT バインド・オプションのモニター・エレメント』
SECTION_NUMBER	BIGINT	section_number セクション番号
STMT_FIRST_USE_TIME	TIMESTAMP	stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ

表 21. ロック・イベント・モニターに戻される情報 : デフォルトの表名: *LOCK_PARTICIPANT_ACTIVITIES_evmon-name* (続き)

列名	データ・タイプ	説明
STMT_INVOCATION_ID	BIGINT	stmt_invocation_id ステートメント呼び出し ID
STMT_LAST_USE_TIME	TIMESTAMP	stmt_last_use_time - ステートメント最終使用時タイム・スタンプ
STMT_LOCK_TIMEOUT	INTEGER	stmt_lock_timeout ステートメント・ロック・タイムアウト
STMT_NEST_LEVEL	BIGINT	stmt_nest_level ステートメント・ネスト・レベル
STMT_OPERATION	VARCHAR(128)	1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』
STMT_PKGCACHE_ID	BIGINT	stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID
STMT_QUERY_ID	BIGINT	stmt_query_id ステートメント照会 ID
STMT_SOURCE_ID	BIGINT	stmt_source_id ステートメント・ソース ID
STMT_TEXT	CLOB	stmt_text SQL ステートメント・テキスト
STMT_TYPE	BIGINT	stmt_type ステートメント・タイプ
STMT_UNICODE	CHARACTER(3)	1542 ページの『stmt_unicode - ステートメントのユニコード・フラグのモニター・エレメント』
UOW_ID	INTEGER	uow_id 作業単位 ID

表 22. ロック・イベント・モニターに戻される情報 : デフォルトの表名: *LOCK_ACTIVITY_VALUES_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACTIVITY_ID	BIGINT	activity_id アクティビティー ID
EVENT_ID	BIGINT	event_id - イベント ID モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
EVENT_TYPE	VARCHAR(128)	event_type - イベント・タイプ・モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー
PARTICIPANT_NO	SMALLINT	participant_no デッドロック内の参加者
STMT_VALUE_DATA	CLOB	stmt_value_data 値データ
STMT_VALUE_INDEX	INTEGER	stmt_value_index 値索引
STMT_VALUE_ISNULL	INTEGER	stmt_value_isnull NULL 値の値
STMT_VALUE_ISREOPT	INTEGER	stmt_value_isreopt ステートメント再最適化に使用される変数
STMT_VALUE_TYPE	CHARACTER(16)	stmt_value_type 値タイプ
UOW_ID	INTEGER	uow_id 作業単位 ID

表 23. ロック・イベント・モニターに戻される情報 : デフォルトの表名: *CONTROL_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号

ロック・イベント・モニターの場合に EVMON_FORMAT_UE_TO_TABLES によってリレーショナル表に書き込まれる情報:

EVMON_FORMAT_UE_TO_TABLES 表関数からロック・イベント・モニター用に書き込まれる情報。これは、`sqllib/misc/DB2EvmonLocking.xsd` ファイルにも記載されています。

表 24. ロック・イベント・モニターに戻される情報: 表名: *LOCK_EVENT*

列名	データ・タイプ	説明
XMLID	VARCHAR(256) NOT NULL	1744 ページの『xmlid - XML ID モニター・エレメント』
DEADLOCK_TYPE	VARCHAR(10)	978 ページの『deadlock_type - デッドロック・タイプのモニター・エレメント』
EVENT_ID	BIGINT NOT NULL	event_id - イベント ID モニター・エレメント
EVENT_TYPE	VARCHAR(128) NOT NULL	event_type - イベント・タイプ・モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP NOT NULL	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
MEMBER	SMALLINT NOT NULL	member - データベース・メンバー
DL_CONNS	INTEGER	dl_conns デッドロックに関係している接続
ROLLED_BACK_PARTICIPANT_NO	INTEGER	rolled_back_participant_no ロールバック参加アプリケーション

表 25. ロック・イベント・モニターに戻される情報: 表名: *LOCK_PARTICIPANTS*

列名	データ・タイプ	説明
XMLID	VARCHAR(256) NOT NULL	1744 ページの『xmlid - XML ID モニター・エレメント』
PARTICIPANT_NO	INTEGER	participant_no デッドロック内の参加者

表 25. ロック・イベント・モニターに戻される情報: 表名: LOCK_PARTICIPANTS (続き)

列名	データ・タイプ	説明
PARTICIPANT_TYPE	VARCHAR(10)	1259 ページの『participant_type - 参加者タイプのモニター・エレメント』
PARTICIPANT_NO_HOLDING_LK	INTEGER	participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル
APPL_ACTION	VARCHAR(64)	840 ページの『appl_action - アプリケーション・アクションのモニター・エレメント』
APPL_ID	VARCHAR(128)	appl_id - アプリケーション ID
APPL_NAME	VARCHAR(128)	appl_name アプリケーション名
AUTH_ID	VARCHAR(128)	auth_id 許可 ID
AGENT_TID	BIGINT	828 ページの『agent_tid - エージェント・スレッド ID モニター・エレメント』
COORD_AGENT_TID	BIGINT	936 ページの『coord_agent_tid - コーディネーター・エージェントのエンジン・ディスパッチ可能単位 ID のモニター・エレメント』
AGENT_STATUS	INTEGER	agent_status DCS アプリケーション・エージェント
DEADLOCK_MEMBER	SMALLINT	977 ページの『deadlock_member - デッドロック・メンバー : モニター・エレメント』
LOCK_TIMEOUT_VAL	BIGINT	lock_timeout_val ロック・タイムアウト値
LOCK_WAIT_VAL	BIGINT	1159 ページの『lock_wait_val - ロック待機値のモニター・エレメント』
WORKLOAD_ID	INTEGER	workload_id ワークロード ID
WORKLOAD_NAME	VARCHAR(128)	workload_name - ワークロード名
SERVICE_CLASS_ID	INTEGER	service_class_id サービス・クラス ID
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name サービス・スーパークラス名
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name サービス・サブクラス名
CURRENT_REQUEST	VARCHAR(32)	958 ページの『current_request - 現在の操作要求のモニター・エレメント』

表 25. ロック・イベント・モニターに戻される情報: 表名: *LOCK_PARTICIPANTS (続き)*

列名	データ・タイプ	説明
LOCK_ESCALATION	CHAR(3)	lock_escalation ロック・エスカレーション
PAST_ACTIVITIES_WRAPPED	CHAR(3)	1261 ページの『past_activities_wrapped - 過去のアクティビティー・リストの折り返しモニター・エレメント』
CLIENT_USERID	VARCHAR(255)	client_userid - クライアントのユーザー ID
CLIENT_WRKSTNNAME	VARCHAR(255)	client_wrkstnname - クライアント・ワークステーション名
CLIENT_APPLNAME	VARCHAR(255)	client_applname - クライアント・アプリケーション名
CLIENT_ACCTNG	VARCHAR(255)	client_acctng - クライアント・アカウンティング・ストリング
OBJECT_REQUESTED	VARCHAR(10)	1232 ページの『object_requested - 要求されたオブジェクトのモニター・エレメント』
LOCK_NAME	CHAR(32)	lock_name ロック名
LOCK_OBJECT_TYPE	VARCHAR(32)	lock_object_type 待機中のロック対象タイプ
LOCK_OBJECT_TYPE_ID	CHAR(1) FOR BIT DATA	将来の利用のために予約済み
LOCK_ATTRIBUTES	CHAR(8)	lock_attributes ロック属性
LOCK_CURRENT_MODE	BIGINT	lock_current_mode 移行前の元のロック・モード
LOCK_MODE_REQUESTED	BIGINT	lock_mode_requested 要求されているロック・モード
LOCK_MODE	BIGINT	lock_mode ロック・モード
LOCK_COUNT	BIGINT	lock_count ロック・カウント
LOCK_HOLD_COUNT	BIGINT	lock_hold_count ロック保留カウント
LOCK_RRIID	BIGINT	
LOCK_STATUS	BIGINT	lock_status ロック状況
LOCK_RELEASE_FLAGS	CHAR(8)	lock_release_flags ロック保留解除フラグ
LOCK_WAIT_START_TIME	TIMESTAMP	lock_wait_start_time ロック待機開始タイム・スタンプ
LOCK_WAIT_END_TIME	TIMESTAMP	1153 ページの『lock_wait_end_time - ロック待機終了タイム・スタンプ : モニター・エレメント』

表 25. ロック・イベント・モニターに戻される情報: 表名: LOCK_PARTICIPANTS (続き)

列名	データ・タイプ	説明
QUEUED_AGENTS	BIGINT	1437 ページの『queued_agents - キューに入れられているしきい値エージェントのモニター・エレメント』
QUEUE_START_TIME	TIMESTAMP	1435 ページの『queue_start_time - キュー開始タイム・スタンプのモニター・エレメント』
TABLE_FILE_ID	BIGINT	table_file_id 表ファイル ID
TABLE_NAME	VARCHAR(128)	table_name 表名
TABLE_SCHEMA	VARCHAR(128)	table_schema 表スキーマ名
TABLESPACE_NAME	VARCHAR(128)	tablespace_name 表スペース名
THRESHOLD_ID	INTEGER	1598 ページの『thresholdid しきい値 ID : モニター・エレメント』
THRESHOLD_NAME	VARCHAR(128)	threshold_name しきい値名
UTILITY_INVOCATION_ID	VARCHAR(32) FOR BIT DATA	
XID	VARCHAR(140) FOR BIT DATA)	xid トランザクション ID

表 26. ロック・イベント・モニターに戻される情報: 表名: LOCK_PARTICIPANT_ACTIVITIES

列名	データ・タイプ	説明
XMLID	VARCHAR(256) NOT NULL	1744 ページの『xmlid - XML ID モニター・エレメント』
PARTICIPANT_NO	INTEGER	participant_no デッドロック内の参加者
ACTIVITY_ID	INTEGER	activity_id アクティビティー ID
ACTIVITY_TYPE	VARCHAR(10)	activity_type アクティビティー・タイプ
UOW_ID	INTEGER	uow_id 作業単位 ID
PACKAGE_NAME	VARCHAR(128)	package_name パッケージ名
PACKAGE_SCHEMA	VARCHAR(128)	package_schema - パッケージ・スキーマ
PACKAGE_VERSION_ID	VARCHAR(64)	package_version_id パッケージ・バージョン
CONSISTENCY_TOKEN	VARCHAR(8)	consistency_token パッケージ整合性トークン
SECTION_NUMBER	BIGINT	section_number セクション番号
REOPT	VARCHAR(10)	1447 ページの『reopt - REOPT バインド・オプションのモニター・エレメント』
INCREMENTAL_BIND	CHAR(3)	1097 ページの『incremental_bind - 追加バインドのモニター・エレメント』

表 26. ロック・イベント・モニターに戻される情報: 表名: *LOCK_PARTICIPANT_ACTIVITIES (続き)*

列名	データ・タイプ	説明
EFFECTIVE_ISOLATION	CHAR(2)	effective_isolation - 有効な分離
EFFECTIVE_QUERY_DEGREE	BIGINT	effective_query_degree - 有効な照会の度合い
STMT_LOCK_TIMEOUT	INTEGER	stmt_lock_timeout ステートメント・ロック・タイムアウト
STMT_TYPE	BIGINT	stmt_type ステートメント・タイプ
STMT_QUERY_ID	BIGINT	stmt_query_id ステートメント照会 ID
STMT_NEST_LEVEL	BIGINT	stmt_nest_level ステートメント・ネスト・レベル
STMT_INVOCATION_ID	BIGINT	stmt_invocation_id ステートメント呼び出し ID
STMT_OPERATION	VARCHAR(128)	1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』
STMT_SOURCE_ID	BIGINT	stmt_source_id ステートメント・ソース ID
STMT_PKGCACHE_ID	BIGINT	stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID
STMT_FIRST_USE_TIME	TIMESTAMP	stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ
STMT_LAST_USE_TIME	TIMESTAMP	stmt_last_use_time - ステートメント最終使用時タイム・スタンプ
STMT_TEXT	CLOB(2097152)	stmt_text SQL ステートメント・テキスト
STMT_UNICODE	CHAR(3)	1542 ページの『stmt_unicode - ステートメントのユニコード・フラグのモニター・エレメント』
QUERY_ACTUAL_DEGREE	INTEGER	query_actual_degree - パーティション内並列処理の実際の実行時の多重度

表 27. ロック・イベント・モニターに戻される情報: 表名: *LOCK_ACTIVITY_VALUES*

列名	データ・タイプ	説明
XMLID	VARCHAR(256) NOT NULL	1744 ページの『xmlid - XML ID モニター・エレメント』
PARTICIPANT_NO	INTEGER	participant_no デッドロック内の参加者
ACTIVITY_ID	INTEGER	activity_id アクティビティー ID

表 27. ロック・イベント・モニターに戻される情報: 表名: LOCK_ACTIVITY_VALUES (続き)

列名	データ・タイプ	説明
UOW_ID	INTEGER	uow_id 作業単位 ID
STMT_VALUE_INDEX	INTEGER	stmt_value_index 値索引
STMT_VALUE_ISREOPT	INTEGER	stmt_value_isreopt ステートメント再最適化に使用される変数
STMT_VALUE_ISNULL	INTEGER	stmt_value_isnull NULL 値の値
STMT_VALUE_TYPE	CHAR(16)	stmt_value_type 値タイプ
STMT_VALUE_DATA	CLOB (32K)	stmt_value_data 値データ

ロック・イベント・モニターの場合に EVMON_FORMAT_UE_TO_XML によって XML に書き込まれる情報:

EVMON_FORMAT_UE_TO_XML 表関数からロック・イベント・モニター用に書き込まれる情報。これは、sqllib/misc/DB2EvmonLocking.xsd ファイルにも記載されています。

db2_lock_event

ロック・タイムアウト、ロック待機、またはデッドロック・イベントを詳述するメイン・スキーマ。

エレメント・コンテント: ((『db2_deadlock_graph』 {0 または 1 個 (?)}、 198 ページの『db2_participant』 {1 個以上 (+)}) | (198 ページの『db2_message』、 198 ページの『db2_event_file』))

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			必須	
type				必須	
timestamp	xs:dateTime			必須	
メンバー				必須	
release	xs:long			必須	
ANY ネーム・スペースの ANY 属性					

db2_deadlock_graph

スキーマ・エレメントは DB2 Deadlock Graph を表します。このグラフは、デッドロックに関係するすべての参加者を略述します。

これを含むエレメント: 『db2_lock_event』

エレメント・コンテント: (209 ページの『db2_participant』 {1 個以上 (+)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
dl_conns	xs:int			必須	
rolled_back _participant_no	xs:int			必須	
type			必須		
ANY ネーム・スペースの ANY 属性					

db2_participant

スキーマ・エレメントは、ロック・イベントに関係するすべての参加者のアプリケーション情報を表します。

これを含むエレメント: 197 ページの『db2_lock_event』、 197 ページの『db2_deadlock_graph』

エレメント・コンテント: (203 ページの『db2_object_requested 』 {0 または 1 個 (?)}、 203 ページの『db2_app_details』、 204 ページの『db2_activity』 {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
いいえ	xs:int			必須	
type			必須		
participant_no _holding_lk	xs:int			オプション	
deadlock_member	xs:int			オプション	
ANY ネーム・スペースの ANY 属性					

db2_message

エラー・メッセージ

これを含むエレメント: 197 ページの『db2_lock_event』

db2_event_file

イベントが書き込まれたファイルの絶対パス。

これを含むエレメント: 197 ページの『db2_lock_event』

application_handle

システム全体での、アプリケーションのユニーク ID。詳しくは、モニター・エレメント 824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

appl_id

アプリケーションがデータベース・マネージャーでデータベースに接続すると、この ID が生成されます。詳しくは、モニター・エレメント 842 ページの『appl_id - アプリケーション ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

appl_name

クライアントで実行中のアプリケーションの名前。データベースが識別できる名前です。詳しくは、モニター・エレメント 847 ページの『appl_name アプリケーション名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

auth_id

モニターされているアプリケーションを呼び出したユーザーの許可 ID。詳しくは、モニター・エレメント 866 ページの『auth_id 許可 ID』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

agent_tid

これを含むエレメント: 203 ページの『db2_app_details』

エレメント・コンテント :

タイプ	ファセット
xs:long	

coord_agent_tid

これを含むエレメント: 203 ページの『db2_app_details』

エレメント・コンテント :

タイプ	ファセット
xs:long	

agent_status

現在のロック待機の状況ではなく、ロック待機状況になる前に有効だったアプリケーションの状況。詳しくは、モニター・エレメント 850 ページの『appl_status アプリケーション状況 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			オプション	

appl_action

クライアント・アプリケーションが実行中のアクション/要求。

これを含むエレメント: 203 ページの『db2_app_details』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			オプション	

lock_timeout_val

データベース構成パラメーターのロック・タイムアウト。値は秒単位です。詳しくは、モニター・エレメント 1149 ページの『lock_timeout_val ロック・タイムアウト値 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_wait_val

ロック・イベント時に有効なロック待機パラメーター。これは、ワークロード・レベルで指定されたデータベース構成パラメーター MON_LKWAIT_THRSH または COLLECT LOCK WAIT DATA 設定のいずれかになります。値はミリ秒単位です。

これを含むエレメント: 203 ページの『db2_app_details』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tentry_state

TEntry 状態。内部使用専用です。

これを含むエレメント: 203 ページの『db2_app_details』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			オプション	

tentry_flag1

TEntry flags1。内部使用専用です。

これを含むエレメント: 203 ページの『db2_app_details』

tentry_flag2

TEntry flags2。内部使用専用です。

これを含むエレメント: 203 ページの『db2_app_details』

xid

XID - グローバル・トランザクション ID

これを含むエレメント: 203 ページの『db2_app_details』

workload_id

このアプリケーションが所属するワークロードの ID。詳しくは、モニター・エレメント 1738 ページの『workload_id ワークロード ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

workload_name

このアプリケーションが所属するワークロードの名前。詳しくは、モニター・エレメント 1739 ページの『workload_name ワークロード名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

service_class_id

このアプリケーションが所属するサービス・サブクラスの ID。詳しくは、モニター・エレメント 1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

service_subclass_name

このアプリケーションが所属するサービス・サブクラスの名前。詳しくは、モニター・エレメント 1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

current_request

現在処理中または最後に処理された操作。

これを含むエレメント: 203 ページの『db2_app_details』

lock_escalation

ロック要求がロック・エスカレーションの一部として行われたかどうかを示します。詳しくは、モニター・エレメント 1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』 を参照してください。可能な値は Yes または No です。

これを含むエレメント: 203 ページの『db2_app_details』

past_activities_wrapped

アクティビティー・リストが折り返したかどうかを示します。 1 つのアプリケーションが保持する過去のアクティビティーの数のデフォルトの限度は 250 です。このデフォルトは、レジストリー変数 DB2_MAX_INACT_STMTS を使用してオーバーライドできます。この限度に対して異なる値を選択することによって、非アクティブ・ステートメントの情報のために用いられるシステム・モニター・ヒープの量を増減できます。

これを含むエレメント: 203 ページの『db2_app_details』

client_userid

トランザクション・マネージャーが生成してサーバーに提供したクライアント・ユーザー ID。詳しくは、モニター・エレメント 902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

client_wrkstnname

この接続で sqleseti API が発行された場合に、クライアントのシステムまたはワークステーションを示します。詳しくは、モニター・エレメント 903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

client_applname

この接続で sqleseti API が発行された場合に、トランザクションを実行中のサーバー・トランザクション・プログラムを示します。詳しくは、モニター・エレメント 894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_app_details』

client_acctng

この接続で sqleseti API が発行された場合に、ロギングおよび診断の目的でターゲット・データベースに渡されたデータです。詳しくは、モニター・エレメント 893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』 を参照してください。

これを含むエレメント: 『db2_app_details』

utility_invocation_id

これを含むエレメント: 『db2_app_details』

service_superclass_name

このアプリケーションが所属するサービス・スーパークラスの名前。詳しくは、モニター・エレメント 1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 『db2_app_details』

db2_object_requested

スキーマ・エレメントは、要求者が取得しようとしている DB2 ロック (所有者によって保持されている) を表します。

これを含むエレメント: 198 ページの『db2_participant』

エレメント・コンテント: ((204 ページの『lock_name』、 204 ページの『lock_object_type』、 205 ページの『lock_specifics』、 205 ページの『lock_attributes』、 205 ページの『lock_current_mode』、 205 ページの『lock_mode_requested』、 205 ページの『lock_mode』、 206 ページの『lock_count』、 206 ページの『lock_hold_count』、 206 ページの『lock_rriid』、 206 ページの『lock_status』、 207 ページの『lock_release_flags』 、 207 ページの『tablespace_name』、 207 ページの『table_name』、 207 ページの『table_schema』、 208 ページの『lock_object_type_id』、 208 ページの『lock_wait_start_time』、 208 ページの『lock_wait_end_time』、 ANY コンテント (検証しない (skip)) {0 個以上 (*)}) | (208 ページの『threshold_name』、 208 ページの『threshold_id』、 208 ページの『queued_agents』、 209 ページの『queue_start_time』、 ANY コンテント (検証しない (skip)) {0 個以上 (*)}))

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
type			必須		

db2_app_details

スキーマ・エレメントは、この参加者に関する詳細を表します。

これを含むエレメント: 198 ページの『db2_participant』

エレメント・コンテント: (198 ページの『application_handle』 、 199 ページの『appl_id』 、 199 ページの『appl_name』 、 199 ページの『auth_id』 、 199 ページの『agent_tid』 、 199 ページの『coord_agent_tid』 、 199 ページの『agent_status』 、 200 ページの『appl_action』 、 200 ページの『lock_timeout_val』 、 200 ページの『lock_wait_val』 、 200 ページの『tentry_state』 、 201 ページの『tentry_flag1』 、 201 ページの『tentry_flag2』 、 201 ページの『xid』 、 201 ページの『workload_id』 、 201 ページの『workload_name』 、 201 ページの『service_class_id』 、 201 ページの『service_subclass_name』 、 202 ページの『current_request』 、 202 ページの『lock_escalation』 、 202 ページの『past_activities_wrapped 』 、 202 ページの『client_userid』 、 202 ページの『client_wrkstnname』 、 202 ページの『client_applname』 、 203 ページの『client_acctng』 、 203 ページの『utility_invocation_id』 、 203 ページの『service_superclass_name』、ANY コンテント (検証しない (skip)) {0 個以上 (*)})

db2_activity

アプリケーションが現在実行しているか、あるいはすでに実行したすべての DB2 アクティビティーのリスト。

これを含むエレメント: 198 ページの『db2_participant』

エレメント・コンテント: (215 ページの『db2_activity_details 』、 215 ページの『db2_input_variable』 {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
type			必須		
ANY ネーム・スペースの ANY 属性					

lock_name

内部バイナリー・ロック名。このエレメントはロックのユニーク ID を示します。詳しくは、モニター・エレメント 1143 ページの『lock_name ロック名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

lock_object_type

アプリケーションがロックを取得するために待機しているオブジェクトのタイプ。詳しくは、モニター・エレメント 1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	

lock_specifics

ロックに関する内部特性。通知メッセージです。

これを含むエレメント: 203 ページの『db2_object_requested 』

lock_attributes

ロックの属性。詳しくは、モニター・エレメント 1128 ページの『lock_attributes ロック属性 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

lock_current_mode

変換する前の元のロック。詳しくは、モニター・エレメント 1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	
モード				オプション	

lock_mode_requested

この参加者が要求しているロック・モード。詳しくは、モニター・エレメント 1142 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	
モード				オプション	

lock_mode

保持されているロックのタイプ。詳しくは、モニター・エレメント 1140 ページの『lock_mode - ロック・モード : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	
モード				オプション	

lock_count

保持されているロックに関するロックの数。詳しくは、モニター・エレメント 1129 ページの『lock_count ロック・カウント : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_hold_count

ロックに置かれている保留の数。詳しくは、モニター・エレメント 1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_rriid

行ロック用の IID。内部使用専用です。

これを含むエレメント: 203 ページの『db2_object_requested 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_status

ロックの内部状況を示します。詳しくは、モニター・エレメント 1148 ページの『lock_status - ロック状況 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			オプション	

lock_release_flags

ロック保留解除フラグ。詳しくは、モニター・エレメント 1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

tablespace_name

ロックが保持されている表スペースの名前。詳しくは、モニター・エレメント 1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	

table_name

ロックが保持されている表の名前。詳しくは、モニター・エレメント 1555 ページの『table_name - 表名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	
data_member_id				オプション	情報が戻されるデータ・メンバーの ID。

table_schema

表のスキーマ。詳しくは、モニター・エレメント 1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

lock_object_type_id

アプリケーションがロックを取得するために待機しているオブジェクトのタイプ。詳しくは、モニター・エレメント 1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

lock_wait_start_time

ロック所有者によって現在ロックされているオブジェクトのロックを取得するための待機をアプリケーションが開始した日時。詳しくは、モニター・エレメント 1153 ページの『lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 203 ページの『db2_object_requested 』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

lock_wait_end_time

ロック所有者によって現在ロックされているオブジェクトのロックを取得するための待機をアプリケーションが停止した日時。

これを含むエレメント: 203 ページの『db2_object_requested 』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

threshold_name

しきい値キューの名前。

これを含むエレメント: 203 ページの『db2_object_requested 』

threshold_id

しきい値キューの ID。

これを含むエレメント: 203 ページの『db2_object_requested 』

queued_agents

しきい値で現在キューに入れられているエージェント数の合計。

これを含むエレメント: 203 ページの『db2_object_requested 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

queue_start_time

しきい値チケットを取得するためにキューでの待機をアプリケーションが開始した日時。

これを含むエレメント: 203 ページの『db2_object_requested 』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

db2_participant

スキーマ・エレメントは、デッドロック・グラフ内の単一スタック項目を表します。

これを含むエレメント: 197 ページの『db2_lock_event』、 197 ページの『db2_deadlock_graph』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
いいえ	xs:int			必須	
deadlock_member				必須	
participant_no _holding_lk	xs:int			必須	
application_handle				必須	
ANY ネーム・スペースの ANY 属性					

activity_id

特定の作業単位内のアプリケーションのアクティビティーを一意的に識別するカウンター。詳しくは、モニター・エレメント 819 ページの『activity_id アクティビティー ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

uow_id

このアクティビティー・レコードが適用される作業単位 ID。詳しくは、モニター・エレメント 1711 ページの『uow_id 作業単位 ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

package_name

現在実行中の SQL ステートメントが含まれているパッケージの名前。詳しくは、モニター・エレメント 1248 ページの『package_name - パッケージ名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

package_schema

SQL ステートメントに関連したパッケージのスキーマ名。詳しくは、モニター・エレメント 1249 ページの『package_schema - パッケージ・スキーマ : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

package_version_id

パッケージ・バージョンは、現在実行中の SQL ステートメントを含むパッケージのバージョン ID を示します。詳しくは、モニター・エレメント 1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

consistency_token

パッケージ整合性トークンを使用すると、現在実行中の SQL ステートメントを含むパッケージのバージョンを識別できます。詳しくは、モニター・エレメント 926 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

section_number

現在処理中または最後に処理された SQL ステートメントのパッケージにある内部セクション番号。詳しくは、モニター・エレメント 1477 ページの『section_number - セクション番号 : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

reopt

このパッケージをプリコンパイルするために使用される REOPT バインド・オプション。可能な値は NONE、ONCE、および ALWAYS です。詳しくは、REOPT バインド・オプションを参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

incremental_bind

パッケージが実行時に増分でバインドされました。可能な値は Yes または No です。

これを含むエレメント: 215 ページの『db2_activity_details 』

effective_isolation

SQL ステートメントが実行されていたときに有効であった分離の値。詳しくは、モニター・エレメント 1006 ページの『effective_isolation - 有効な分離 : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	

effective_query_degree

SQL ステートメントが実行されていたときに有効であった度合いの値。詳しくは、モニター・エレメント 1007 ページの『effective_query_degree - 有効な照会の度合い : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_unicode

SQL ステートメントのユニコード・フラグ。可能な値は Yes または No です。

これを含むエレメント: 215 ページの『db2_activity_details 』

stmt_lock_timeout

SQL ステートメントが実行されていたときに有効であったロック・タイムアウトの値。詳しくは、モニター・エレメント 1531 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:int	

stmt_type

処理される SQL ステートメントのタイプ。可能な値は Dynamic または Static です。詳しくは、モニター・エレメント 1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			必須	

stmt_operation

これを含むエレメント: 215 ページの『db2_activity_details 』

stmt_query_id

SQL ステートメントに与えられる内部照会 ID。詳しくは、モニター・エレメント 1535 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_nest_level

このエレメントは、ステートメントが実行されたときにそのステートメントに対して有効だった、ネストまたは再帰のレベルを示します。詳しくは、モニター・エレメント 1532 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_invocation_id

このエレメントは、SQL ステートメントが実行されたルーチン呼び出しの ID を示します。詳しくは、モニター・エレメント 1530 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_source_id

このエレメントは、実行された SQL ステートメントのソースに付けられた内部 ID を示します。詳しくは、モニター・エレメント 1537 ページの『stmt_source_id ステートメント・ソース ID』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_pkgcache_id

このエレメントは、動的 SQL ステートメントの内部パッケージ・キャッシュ ID を示します。詳しくは、モニター・エレメント 1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_text

SQL ステートメントのテキスト。詳しくは、モニター・エレメント 1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

stmt_first_use_time

このエレメントは、ステートメント項目が最初に処理されたときを示します。カーソル操作の場合、 1528 ページの『stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ : モニター・エレメント』 はカーソルがオープンされたときを示します。アプリケーション調整ノードでは、この値はアプリケーション要求を反映します。非コーディネーター・ノードでは、この値は要求が起点ノードから受信されたときを反映します。詳しくは、モニター・エレメント stmt_first_use_time を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

stmt_last_use_time

このエレメントは、ステートメント項目が最後に処理されたときを示します。カーソル操作の場合、 1531 ページの『stmt_last_use_time - ステートメント最終使用時タイム・スタンプ : モニター・エレメント』 は、カーソルに対する最後のアクションの時刻を示します。そのときのアクションとして、オープン、フェッチ、またはクローズが考えられます。アプリケーション調整ノードでは、この値はアプリケーション要求を反映します。非コーディネーター・ノードでは、この値は要求が起点ノードから受信されたときを反映します。詳しくは、モニター・エレメント stmt_last_use_time を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

query_actual_degree

SQL ステートメント実行時の実際の多重度の値。詳しくは、モニター・エレメント 1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_activity_details 』

エレメント・コンテント :

タイプ	ファセット
xs:int	

db2_activity_details

スキーマは、このアクティビティーに関する詳細を表します。

これを含むエレメント: 204 ページの『db2_activity』

エレメント・コンテント: (209 ページの『activity_id』、 209 ページの『uow_id』、 210 ページの『package_name』、 210 ページの『package_schema』、 210 ページの『package_version_id』、 210 ページの『consistency_token』、 210 ページの『section_number』、 210 ページの『reopt』、 211 ページの『incremental_bind』、 211 ページの『effective_isolation』、 211 ページの『effective_query_degree 』、 211 ページの『stmt_unicode』、 211 ページの『stmt_lock_timeout』、 212 ページの『stmt_type』、 212 ページの『stmt_operation』、 212 ページの『stmt_query_id』、 212 ページの『stmt_nest_level』、 213 ページの『stmt_invocation_id』、 213 ページの『stmt_source_id』、 213 ページの『stmt_pkgcache_id』、 213 ページの『stmt_text』、 214 ページの『stmt_first_use_time』、 214 ページの『stmt_last_use_time』、 214 ページの『query_actual_degree』、 ANY コンテント (検証しない (skip)) {0 個以上 (*)})

db2_input_variable

スキーマ・エレメントは、SQL ステートメントに関連付けられた入力変数のリストを表します。

これを含むエレメント: 204 ページの『db2_activity』

エレメント・コンテント: (『stmt_value_index』、『stmt_value_isreopt』、 216 ページの『stmt_value_isnull』、 216 ページの『stmt_value_type』、 216 ページの『stmt_value_data』、 ANY コンテント (検証しない (skip)) {0 個以上 (*)})

stmt_value_index

このエレメントは、SQL ステートメントで使用される入力パラメーター・マーカーまたはホスト変数の位置を表します。詳しくは、モニター・エレメント 1543 ページの『stmt_value_index 値索引』 を参照してください。

これを含むエレメント: 『db2_input_variable』

stmt_value_isreopt

このエレメントは、変数がステートメント最適化のやり直し中に使用されたかどうかを示します。詳しくは、モニター・エレメント 1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 『db2_input_variable』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			必須	

stmt_value_isnull

このエレメントは、SQL ステートメントに関連したデータ値が NULL 値かどうかを示します。詳しくは、モニター・エレメント 1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』 を参照してください。

これを含むエレメント: 215 ページの『db2_input_variable』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			必須	

stmt_value_type

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

これを含むエレメント: 215 ページの『db2_input_variable』

stmt_value_data

このエレメントは、SQL ステートメントに関連したデータ値のストリング表記です。詳しくは、モニター・エレメント 1543 ページの『stmt_value_data 値データ』を参照してください。

これを含むエレメント: 215 ページの『db2_input_variable』

ロック・イベント・データの収集とレポートの生成

ロック・イベント・モニターを使用してロック・タイムアウト、ロック待機、およびデッドロック情報を収集することにより、ロッキングに関する問題の識別と解決に役立てることができます。ロック・イベント・データは読めない形式で未フォーマット・イベント表に収集されるので、このタスクでは、可読テキスト・レポートを後から取得する方法について説明します。

始める前に

ロッキング・イベント・モニターを作成してロック・イベント・モニター・データを収集するには、DBADM または SQLADM 権限がなければなりません。

このタスクについて

ロック・イベント・モニターは、ロッキングに関する問題の識別と解決に役立つ関連情報を収集します。ロック・イベント・モニターが収集するロック・イベントの情報には、例えば以下の情報が含まれます。

- ロック・イベントになったロック。
- ロック・イベントになったロックを要求または保持しているアプリケーション。
- ロック・イベント時のアプリケーションの実行内容。

このタスクでは、特定のワークロードに関するロック・イベント・データを収集するための手順を説明します。次のような状況下では、ロック・イベント・データを収集することをお勧めします。

- MON_GET_WORKLOAD 表関数を使用したところ、ロック待機の値がいつもより長い。
- アプリケーションが「ロック・タイムアウトのために、トランザクションがロールバックされました」という内容の -911 SQL 戻りコードを理由コード 68 と共に管理通知ログに戻す。詳しくは、メッセージ SQL0911N も参照してください。
- 管理通知ログにデッドロック・イベント・メッセージ (「デッドロックのために、トランザクションがロールバックされました」という内容の -911 SQL 戻りコードで、理由コードが 2) が含まれている。ログ・メッセージが、2 つのアプリケーション (例えば、ワークロード FINANCE の一部であるアプリケーション A とワークロード PAYROLL の一部であるアプリケーション B) の間でロック・イベントが発生したことを示している。詳しくは、メッセージ SQL0911N も参照してください。

制約事項

データ値を表示するには、EVMON_FORMAT_UE_* ルーチンに対する EXECUTE 特権が必要です。この特権は、SQLADM および DBADM 権限が暗黙的に保持しています。未フォーマット・イベント表に対する SELECT 特権も必要です。DATAACCESS 権限を持つユーザーと、イベント・モニターおよび関連する未フォーマット・イベント表の作成者は、デフォルトでこの特権を保持しています。

手順

今後発生する可能性のあるロック・イベントに関する詳細情報を収集するには、以下のステップを実行します。

1. 次の例に示すように、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、lockevmon というロック・イベント・モニターを作成します。

   ```
   CREATE EVENT MONITOR lockevmon FOR LOCKING
      WRITE TO UNFORMATTED EVENT TABLE
   ```

   注: 以下は、イベント・モニターを作成する際に覚えておかなければならない重要点のリストです。

   - イベント・モニターは前もって作成できます。また、ディスク・スペースが使い尽くされることを心配する必要はありません。データベース・レベルまたはワークロード・レベルのデータ収集を活動化するまでは何も書き込まれません。
   - パーティション・データベース環境では、すべてのノードにわたるパーティション表スペースにイベント・モニターを配置するようにします。こうしないと、パーティション表スペースが存在しないパーティションでは、ロック・イベントが見落とされることになります。
   - データ取得のための表へのアクセス時に進行中の作業によるハイパフォーマンス作業に対する妨害が最小になるように、表スペースとバッファー・プールをセットアップするようにします。
2. 次に示すステートメントを実行して、ロック・イベント・モニター lockevmon を活動化します。

```
SET EVENT MONITOR lockevmon STATE 1
```

3. ワークロード・レベルでロック・イベント・データ収集を使用可能にするには、次に示す COLLECT 節の 1 つを使用して ALTER WORKLOAD ステートメントを発行します。COLLECT LOCK TIMEOUT DATA、COLLECT DEADLOCK DATA、または COLLECT LOCK WAIT DATA のいずれかです。COLLECT 節で WITH HISTORY オプションを指定します。データベース構成パラメーターを設定すると、データベース・レベルのロック・イベント・データ収集に影響を与え、すべてのワークロードが影響を受けます。

   ロック待機イベントの場合

   FINANCE アプリケーションについては 5 秒経過後に獲得されたロックのロック待機データを収集し、PAYROLL アプリケーションについては 10 秒経過後に獲得されたロックのロック待機データを収集するには、以下のステートメントを発行します。

   ```
   ALTER WORKLOAD finance COLLECT LOCK WAIT DATA WITH HISTORY AND VALUES
      FOR LOCKS WAITING MORE THAN 5 SECONDS
   ALTER WORKLOAD payroll COLLECT LOCK WAIT DATA
      FOR LOCKS WAITING MORE THAN 10 SECONDS WITH HISTORY
   ```

   SAMPLE データベースの **mon_lockwait** データベース構成パラメーターを HIST_AND_VALUES 入力データ値を指定して設定し、**mon_lw_thresh** データベース構成パラメーターを 10 秒に設定するには、以下のコマンドを発行します。

   ```
   db2 update db cfg for sample using mon_lockwait hist_and_values
   db2 update db cfg for sample using mon_lw_thresh 10000000
   ```

   ロック・タイムアウト・イベントの場合

   FINANCE および PAYROLL アプリケーションのロック・タイムアウト・データを収集するには、以下のステートメントを発行します。

   ```
   ALTER WORKLOAD finance COLLECT LOCK TIMEOUT DATA WITH HISTORY
   ALTER WORKLOAD payroll COLLECT LOCK TIMEOUT DATA WITH HISTORY
   ```

   SAMPLE データベースの **mon_locktimeout** データベース構成パラメーターを HIST_AND_VALUES 入力データ値を指定して設定するには、次のコマンドを発行します。

   ```
   db2 update db cfg for sample using mon_locktimeout hist_and_values
   ```

   デッドロック・イベントの場合

   FINANCE および PAYROLL アプリケーションのデータを収集するには、以下のステートメントを発行します。

   ```
   ALTER WORKLOAD finance COLLECT DEADLOCK DATA WITH HISTORY
   ALTER WORKLOAD payroll COLLECT DEADLOCK DATA WITH HISTORY
   ```

   SAMPLE データベースの **mon_deadlock** データベース構成パラメーターを HIST_AND_VALUES 入力データ値を指定して設定するには、次のコマンドを発行します。

   ```
   db2 update db cfg for sample using mon_deadlock hist_and_values
   ```

4. 別のロック・イベント通知を受け取るために、ワークロードを再実行します。
5. データベースに接続します。
6. 以下のいずれかのアプローチで、ロッキング・イベント・レポートを取得します。

a. XML パーサー・ツール **db2evmonfmt** を使用して、未フォーマット・イベント表に収集されたイベント・データに基づいた、デフォルトのスタイルシートを使用するフラット・テキスト・レポートを生成します。例えば、次のようにします。

```
java db2evmonfmt -d db_name -ue table_name -ftext -u user_id -p password
```

b. EVMON_FORMAT_UE_TO_XML 表関数を使用して、XML 文書を取得します。

c. EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用して、データをリレーショナル表に出力します。

7. レポートを分析し、ロック・イベントに関する問題の理由を判別して、問題を解決します。

8. 以下のステートメントを実行するか、データベース構成パラメーターを再設定して、FINANCE と PAYROLL の両方のアプリケーションのロック・データ収集をオフにします。

ロック待機イベントの場合

```
ALTER WORKLOAD finance COLLECT LOCK WAIT DATA NONE
ALTER WORKLOAD payroll COLLECT LOCK WAIT DATA NONE
```

SAMPLE データベースの **mon_lockwait** データベース構成パラメーターをデフォルトの NONE 入力データ値を指定して再設定し、**mon_lw_thresh** データベース構成パラメーターを再設定してデフォルト値の 5 秒に戻すには、以下のコマンドを発行します。

```
db2 update db cfg for sample using mon_lockwait none
db2 update db cfg for sample using mon_lw_thresh 5000000
```

ロック・タイムアウト・イベントの場合

```
ALTER WORKLOAD finance COLLECT LOCK TIMEOUT DATA NONE
ALTER WORKLOAD payroll COLLECT LOCK TIMEOUT DATA NONE
```

SAMPLE データベースの **mon_locktimeout** データベース構成パラメーターをデフォルトの NONE 入力データ値を指定して再設定するには、次のコマンドを発行します。

```
db2 update db cfg for sample using mon_locktimeout none
```

デッドロック・イベントの場合

```
ALTER WORKLOAD finance COLLECT DEADLOCK DATA NONE
ALTER WORKLOAD payroll COLLECT DEADLOCK DATA NONE
```

SAMPLE データベースの **mon_deadlock** データベース構成パラメーターをデフォルトの WITHOUT_HIST 入力データ値を指定して再設定するには、次のコマンドを発行します。

```
db2 update db cfg for sample using mon_deadlock without_hist
```

次のタスク

該当アプリケーションを再実行して、ロッキングに関する問題が除去されたことを確認します。

作業単位イベント・モニター

作業単位イベント・モニターは、作業単位が完了した場合はいつでも、つまりコミットまたはロールバックがあるときはいつでも、イベントを記録します。

個々の作業単位に関するこの履歴情報は、チャージバックの目的 (CPU の使用に応じて課金される)、および応答時間サービス・レベル目標の順守のモニターに役立ちます。

作業単位イベント・モニターは、要求メトリックを使用してシステム・パースペクティブ・モニターを実行する 1 つの方法です。作業単位イベント・モニターに最も近い代替手段または補足手段は、統計イベント・モニター、
MON_GET_UNIT_OF_WORK 表関数、および
MON_GET_UNIT_OF_WORK_DETAILS 表関数です。

作業単位イベント・モニターを使用すると、作業単位内で使用されたパッケージ、および各パッケージが使用されたネスティング・レベルのリストを収集できます。この情報によって、ストアード・プロシージャーのトラブルシューティングが容易になります。DB2 バージョン 10.1 以降、作業単位内で実行された各ステートメントに関する、実行可能 ID およびステートメント・レベルの関連メトリックのリストも生成できるようになりました。

作業単位イベント・モニターを作成し、作業単位イベント・モニター・データを収集するには、DBADM または SQLADM 権限が必要です。

作業単位イベント・モニターの作成

DB2 バージョン 10.1 以降、作業単位イベント・モニターの出力を、未フォーマット・イベント (UE) 表または通常の表のどちらに書き込むか選択できます。最も適切な出力形式を選択する方法の詳細については、44 ページの『イベント・モニターの出力オプション』を参照してください。

どちらのタイプの表を使用するかにかかわらず、作業単位イベント・モニターを作成するときには、そのイベント・モニターの出力を格納する表を保管する表スペースを指定します。推奨されている手法は、表を保管するための専用の表スペースを構成する方法です。ただし、イベント・モニターを作成するときに、既存の表スペースを指定することもできます。表スペースを指定しなかった場合は、いずれかの表スペースが自動的に選択されます。

デフォルトおよびベスト・プラクティスを使用して作業単位イベント・モニターを作成するには、次の CREATE EVENT MONITOR ステートメントを使用します。次の例のステートメントは、デフォルトを使用できる箇所にはデフォルトを使用し、表スペース MY_EVMON_TABLESPACE 内の UE 表に出力を保管するように指定しています。

```
CREATE EVENT MONITOR MY_UOW_EVMON
  FOR UNIT OF WORK
   WRITE TO UNFORMATTED EVENT TABLE (IN MY_EVMON_TABLESPACE)
```

データ収集の構成

作業単位データについては、 4 つの異なる収集レベルを指定できます。

1. なし

2. 基本的な作業単位データ
   a. 作業単位内で実行されたパッケージに関する情報
   b. 作業単位内で実行されたステートメントの実行可能 ID のリスト

データベース構成パラメーターを使用すると、データベースでアクティブになっているすべての作業単位イベント・モニターの作業単位データ収集を制御できます。一方、特定のワークロード、サービス・クラス、または作業アクションに関する情報の収集を制御するには、該当するワークロード・オブジェクトに対して CREATE および ALTER ステートメントを使用します。

データベース・レベルでデータ収集を構成するには、**UPDATE DATABASE CONFIGURATION** コマンドを使用して、 **mon_uow_data** データベース構成パラメーターと、オプションで **mon_uow_pkglist** および **mon_uow_execlist** データベース構成パラメーターを設定します。これらのパラメーターに設定可能な値の組み合わせを、表 28 に示します。

表 28. 作業単位イベント・モニターの構成パラメーターに設定可能な値

収集するデータ	mon_uow_data	mon_uow_pkglist	mon_uow_execlist
作業単位データを収集しない	NONE (デフォルト)	OFF (デフォルト)	OFF (デフォルト)
基本的な作業単位データのみ収集する	BASE	OFF (デフォルト)	OFF (デフォルト)
パッケージ・リスト情報は収集するが、実行可能 ID に関する情報は収集しない	BASE	ON	OFF (デフォルト)
実行可能 ID に関する情報は収集するが、パッケージのリストは収集しない	BASE	OFF (デフォルト)	ON
基本的な作業単位データ、パッケージ・リスト情報、および実行可能 ID に関する情報を収集する	BASE	ON	ON

ヒント:

- これらの構成パラメーターのいずれも設定しない場合、作業単位データは収集されません。ただし、特定のワークロード・オブジェクトに対して収集を有効にしている場合は別です。例えば CREATE SERVICE CLASS、ALTER WORKLOAD ステートメントのような CREATE または ALTER ステートメントのいずれかを、該当するワークロード・オブジェクト・タイプに対して使用して、特定のワークロード・オブジェクトに関する収集を有効にすることができます。
- 基本的な作業単位データを収集して、パッケージ・リストおよび実行可能 ID 情報は収集しない場合、 **mon_uow_data** 構成パラメーターを BASE に設定し、**mon_uow_pkglist** および **mon_uow_execlist** 構成パラメーターは省略します。これらを明示的に設定しなければ、デフォルト値 OFF が使用されます。

- パッケージ・リストおよび実行可能 ID 情報の一方または両方を収集する場合は、 **mon_uow_data** 構成パラメーターを BASE に設定する必要もあります。 **mon_uow_data** 構成パラメーターを NONE に設定すると、 **mon_uow_pkglist** および **mon_uow_execlist** 構成パラメーターの設定にかかわらず、情報は収集されません。

特定のワークロード・オブジェクトのデータ収集を制御するには、対象となる特定のタイプのワークロード・オブジェクトに対する CREATE または ALTER ステートメントの COLLECT UNIT OF WORK DATA 節を使用してください。例えば、ワークロード REPORTS に関する基本的な作業単位イベント・データおよびパッケージ・リスト情報を収集するには、次のようなステートメントを実行すると良いでしょう。

```
ALTER WORKLOAD REPORTS COLLECT UNIT OF WORK DATA BASE INCLUDE PACKAGE LIST
```

作業単位内で実行されたステートメントのパッケージ・リスト情報および実行可能 ID リストの両方を収集するには、次のステートメントを使用すると良いでしょう。

```
ALTER WORKLOAD REPORTS COLLECT UNIT OF WORK DATA BASE INCLUDE PACKAGE LIST,
 EXECUTABLE LIST
```

221 ページの表 28 に示した設定は、特定のワークロードに対する設定を CREATE WORKLOAD または ALTER WORKLOAD ステートメントを使用してオーバーライドしていない限り、システム内で実行中のすべてのワークロードに適用されます。すべてのワークロードに関する基本的なレベルの情報と、特定のワークロードに関するパッケージ・リスト情報を収集する場合は、 **mon_uow_data** データベース構成パラメーターに BASE を設定します。そして、問題のワークロードに対して CREATE WORKLOAD または ALTER WORKLOAD ステートメントを使用して、レベルを BASE PACKAGE LIST に設定します。

デフォルトでは、作業単位イベント・モニターなどの、該当する表関数およびイベント・モニターは、要求メトリックを収集して報告します。このデフォルト設定は、以下のようにして変更できます。

- **mon_req_metrics** データベース構成パラメーターを使用する
- サービス・スーパークラスの CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントで COLLECT REQUEST METRICS 節を使用する。

デフォルトの設定を変更すると、要求メトリックを報告するすべての表関数またはイベント・モニターに影響を与えます。

作業単位イベント・モニターによりキャプチャーされたイベント・データへのアクセス

作業単位イベント・モニターは、通常の表にデータを書き込むことも、未フォーマット・イベント (UE) 表にバイナリー・フォーマットのデータを書き込むこともできます。通常の表のデータには SQL を使用してアクセスできます。

UE 表のデータにアクセスするには、次のいずれかの表関数を使用します。

EVMON_FORMAT_UE_TO_XML
未フォーマット・イベント表から XML 文書にデータを抽出します。

EVMON_FORMAT_UE_TO_TABLES
未フォーマット・イベント表から一連のリレーショナル表にデータを抽出します。

これらの表関数のいずれかを使用する場合、関数のパラメーターの 1 つとして SELECT ステートメントを含めることによって、どのデータを抽出するかを指定できます。 SELECT ステートメントにより提供される選択、順序付け、その他の面についてはすべて制御できます。

パッケージ・リスト情報を生成している場合、EVMON_FORMAT_UE_TO_XML 表関数を使用すると、基本的な作業単位イベント・モニター・データとパッケージ・リストの両方を含む 1 つの XML 文書を生成できます。
EVMON_FORMAT_UE_TO_TABLES プロシージャーは、基本的な作業単位イベント・モニター情報用に 1 つ、およびパッケージ・リスト情報用にもう 1 つ、合計 2 つの表を生成します。この 2 つの表は、MEMBER、APPLICATION_ID および UOW_ID 列の値を使用して結合することができます。

db2evmonfmt コマンドを使用して、以下のタスクの実行に役立てることもできます。

- イベント ID、イベント・タイプ、時間枠、アプリケーション、ワークロード、またはサービス・クラスの各属性に基づいて、関心対象のイベントを選択します
- テキスト・レポートまたはフォーマット済み XML 文書のどちらの形式で出力を受け取るかを選択します
- **db2evmonfmt** コマンドに提供されているものを使用する代わりに、独自の XSLT スタイル・シートを作成して、出力形式を制御します

例えば、以下のコマンドは、データベース SAMPLE で過去 24 時間に発生した作業単位イベントを抜粋した作業単位レポートを生成します。これらのイベント・レコードは、SAMPLE_UOW_EVENTS という未フォーマット・イベント表から取得します。このコマンドは、MyUOW.xsl スタイル・シートを使用してフォーマットしたテキスト出力を作成します。

```
java db2evmonfmt -d SAMPLE -ue SAMPLE_UOW_EVENTS -ftext -ss MyUOW.xsl -hours 24
```

作業単位イベント・モニターによって生成されるデータ

作業単位イベント・モニターは、システムで実行される作業単位 (トランザクション) に関するデータを生成します。作業単位イベント・モニターの出力先を通常の表にするか未フォーマット・イベント (UE) 表にするか選択できます。

UE 表にデータを書き込む場合、データを表示するにはデータに後処理を実行する必要があります。

選択した出力形式にかかわらず、作業単位イベント・データは、次の 4 つの論理グループのいずれかから取得されます。

- uow
- uow_metrics
- uow_package_list
- uow_exec_list

作業単位イベント・データを通常の表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

注: 指定しない限り、作業単位イベント・モニターで収集される唯一のモニター・エレメントは、要求メトリックです。基本的な作業単位イベント・データ、またはパッケージや実行リスト・データの生成を有効にするには、明示的にデータ収集を有効にする必要があります。詳しくは、168 ページの『イベント・モニターのデータ収集の有効化』を参照してください。

作業単位イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、作業単位イベント・モニターによって書き込まれる情報。

作業単位イベント・モニターの出力タイプとして WRITE TO TABLE を選択した場合、デフォルトでは、5 つの表が生成されます。各表には、1 つ以上の論理データ・グループのモニター・エレメントが入っています。

表 29. UNIT OF WORK 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
UOW_*evmon-name*	uow
UOW_METRICS_*evmon-name*	uow_metrics
UOW_PACKAGE_LIST_*evmon-name*	uow_package_list
UOW_EXECUTABLE_LIST_*evmon-name*	uow_executable_list
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

注: 5 つの表はすべてデフォルトで生成されますが、収集対象のロック情報の種類について、データ収集が有効になっているか確認する必要があります。有効になっていないと、一部の列にはヌル値がレポートされます。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 30. ロック・イベント・モニターに戻される情報 : デフォルトの表名: UOW_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

表 30. ロック・イベント・モニターに戻される情報 : デフォルトの表名: UOW_evmon-name (続き)

列名	データ・タイプ	説明
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル
APPLICATION_ID	VARCHAR(128)	842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
APPLICATION_NAME	VARCHAR(128)	847 ページの『appl_name アプリケーション名 : モニター・エレメント』
CLIENT_ACCTNG	VARCHAR(255)	client_acctng - クライアント・アカウンティング・ストリング
CLIENT_APPLNAME	VARCHAR(255)	client_applname - クライアント・アプリケーション名
CLIENT_HOSTNAME	VARCHAR(255)	client_hostname - クライアント・ホスト名
CLIENT_PID	BIGINT	client_pid クライアント・プロセス ID
CLIENT_PLATFORM	VARCHAR(12)	client_platform クライアント・オペレーティング・プラットフォーム
CLIENT_PORT_NUMBER	INTEGER	client_port_number - クライアント・ポート番号
CLIENT_PRODUCT_ID	VARCHAR(128)	900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』
CLIENT_PROTOCOL	VARCHAR(10)	client_protocol クライアント通信プロトコル
CLIENT_USERID	VARCHAR(255)	client_userid - クライアントのユーザー ID
CLIENT_WRKSTNNAME	VARCHAR(255)	client_wrkstnname - クライアント・ワークステーション名
COMPLETION_STATUS	VARCHAR(128)	completion_status 完了状況
CONNECTION_TIME	TIMESTAMP	924 ページの『connection_start_time - 接続開始時刻 : モニター・エレメント』
COORD_MEMBER	SMALLINT	coord_member - コーディネーター・メンバー
EVENT_ID	INTEGER NOT NULL	event_id - イベント ID モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP NOT NULL	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント

表 30. ロック・イベント・モニターに戻される情報 : デフォルトの表名: UOW_evmon-name (続き)

列名	データ・タイプ	説明
EXECUTABLE_LIST_SIZE	BIGINT	1021 ページの『executable_list_size - 実行可能リスト・サイズのモニター・エレメント』
EXECUTABLE_LIST_TRUNCATED	CHAR(3)	1022 ページの『executable_list_truncated - 実行可能リスト切り捨てのモニター・エレメント』
GLOBAL_TRANSACTION_ID	VARCHAR(40)	1061 ページの『global_transaction_id - グローバル・トランザクション ID のモニター・エレメント』
INTRA_PARALLEL_STATE	VARCHAR(128)	intra_parallel_state - パーティション内並列処理の現行状態
LOCAL_TRANSACTION_ID	VARCHAR(16)	1126 ページの『local_transaction_id - ローカル・トランザクション ID のモニター・エレメント』
MEMBER	SMALLINT	member - データベース・メンバー
MEMBER_ACTIVATION_TIME	TIMESTAMP	965 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』
METRICS	BLOB	
MON_INTERVAL_ID	VARCHAR(128)	mon_interval_id - モニター間隔 ID
PACKAGE_LIST_EXCEEDED	CHAR(3)	package_list_exceeded - パッケージ・リストの超過
PACKAGE_LIST_SIZE	INTEGER	1248 ページの『package_list_size - パッケージ・リスト・サイズのモニター・エレメント』
SERVICE_CLASS_ID	INTEGER	service_class_id サービス・クラス ID
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name サービス・サブクラス名
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name サービス・スーパークラス名
SESSION_AUTHID	VARCHAR(128)	1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』
START_TIME	TIMESTAMP	start_time イベント開始時刻

表 30. ロック・イベント・モニターに戻される情報 : デフォルトの表名: UOW_evmon-name (続き)

列名	データ・タイプ	説明
STOP_TIME	TIMESTAMP	stop_time イベント停止時刻
SYSTEM_AUTHID	VARCHAR(128)	1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』
UOW_ID	INTEGER	uow_id 作業単位 ID
UOW_LOG_SPACE_USED	BIGINT	uow_log_space_used 使用されている作業単位ログ・スペース
WORKLOAD_ID	INTEGER	workload_id ワークロード ID
WORKLOAD_NAME	VARCHAR(128)	workload_name ワークロード名
WORKLOAD_OCCURRENCE_ID	INTEGER	workload_occurrence_id ワークロード・オカレンス ID

表 31. 作業単位イベント・モニターに戻される情報: 表名: UOW_METRICS_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
APPLICATION_ID	VARCHAR (128)	842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
MEMBER	SMALLINT	member - データベース・メンバー
UOW_ID	INTEGER	uow_id 作業単位 ID
WLM_QUEUE_TIME_TOTAL	BIGINT	wlm_queue_time_total - ワークロード・マネージャー合計キュー時間
WLM_QUEUE_ASSIGNMENTS _TOTAL	BIGINT	wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て
FCM_TQ_RECV_WAIT_TIME	BIGINT	fcm_tq_recv_wait_time - FCM 表キュー受信待機時間
FCM_MESSAGE_RECV_WAIT_TIME	BIGINT	fcm_message_recv_wait_time - FCM メッセージの受信待機時間
FCM_TQ_SEND_WAIT_TIME	BIGINT	fcm_tq_send_wait_time - FCM 表キュー送信待機時間
FCM_MESSAGE_SEND_WAIT_TIME	BIGINT	fcm_message_send_wait_time - FCM メッセージの送信待機時間
AGENT_WAIT_TIME	BIGINT	agent_wait_time - エージェント待機時間
AGENT_WAITS_TOTAL	BIGINT	agent_waits_total - エージェント待機の合計
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCK_WAITS	BIGINT	lock_waits ロック待機数
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求

表 31. 作業単位イベント・モニターに戻される情報: 表名: UOW_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
LOG_BUFFER_WAIT_TIME	BIGINT	log_buffer_wait_time - ログ・バッファー待機時間
NUM_LOG_BUFFER_FULL	BIGINT	num_log_buffer_full フル・ログ・バッファーの回数
LOG_DISK_WAIT_TIME	BIGINT	log_disk_wait_time - ログ・ディスク待機時間
LOG_DISK_WAITS_TOTAL	BIGINT	log_disk_waits_total - ログ・ディスク待機の合計
TCPIP_RECV_WAIT_TIME	BIGINT	tcpip_recv_wait_time - TCP/IP 受信待機時間
TCPIP_RECVS_TOTAL	BIGINT	tcpip_recvs_total - TCP/IP 受信の合計
CLIENT_IDLE_WAIT_TIME	BIGINT	client_idle_wait_time - クライアントのアイドル待機時間
IPC_RECV_WAIT_TIME	BIGINT	ipc_recv_wait_time - プロセス間通信受信待機時間
IPC_RECVS_TOTAL	BIGINT	ipc_recvs_total - プロセス間通信受信の合計
IPC_SEND_WAIT_TIME	BIGINT	ipc_send_wait_time - プロセス間通信送信待機時間
IPC_SENDS_TOTAL	BIGINT	ipc_sends_total - プロセス間通信送信の合計
TCPIP_SEND_WAIT_TIME	BIGINT	tcpip_send_wait_time - TCP/IP 送信待機時間
TCPIP_SENDS_TOTAL	BIGINT	tcpip_sends_total - TCP/IP 送信の合計
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計
AUDIT_FILE_WRITE_WAIT_TIME	BIGINT	audit_file_write_wait_time - 監査ファイル書き込み待機時間
AUDIT_FILE_WRITES_TOTAL	BIGINT	audit_file_writes_total - 書き込まれた監査ファイルの合計
AUDIT_SUBSYSTEM_WAIT_TIME	BIGINT	audit_subsystem_wait_time - 監査サブシステム待機時間
AUDIT_SUBSYSTEM_WAITS_TOTAL	BIGINT	audit_subsystem_waits_total - 監査サブシステム待機の合計
DIAGLOG_WRITE_WAIT_TIME	BIGINT	diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間
DIAGLOG_WRITES_TOTAL	BIGINT	diaglog_writes_total - 診断ログ・ファイル書き込みの合計
FCM_SEND_WAIT_TIME	BIGINT	fcm_send_wait_time - FCM 送信待機時間
FCM_RECV_WAIT_TIME	BIGINT	fcm_recv_wait_time - FCM 受信待機時間
TOTAL_WAIT_TIME	BIGINT	total_wait_time - 合計待機時間
RQSTS_COMPLETED_TOTAL	BIGINT	rqsts_completed_total - 完了した要求の合計
TOTAL_RQST_TIME	BIGINT	total_rqst_time - 合計要求時間
APP_RQSTS_COMPLETED_TOTAL	BIGINT	app_rqsts_completed_total - 完了したアプリケーション要求の合計
TOTAL_APP_RQST_TIME	BIGINT	total_app_rqst_time - 合計アプリケーション要求時間
TOTAL_SECTION_SORT_PROC_TIME	BIGINT	total_section_sort_proc_time - セクションのソート処理時間の合計

表 31. 作業単位イベント・モニターに戻される情報: 表名: UOW_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_SECTION_SORTS	BIGINT	total_section_sorts - セクションのソートの合計
TOTAL_SECTION_SORT_TIME	BIGINT	total_section_sort_time - セクションのソート時間の合計
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_MODIFIED	BIGINT	rows_modified 変更行数
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
ACT_COMPLETED_TOTAL	BIGINT	act_completed_total - 完了したアクティビティーの合計
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
ROWS_RETURNED	BIGINT	rows_returned 戻り行数
DEADLOCKS	BIGINT	deadlocks デッドロック検出数
LOCK_TIMEOUTS	BIGINT	lock_timeouts ロック・タイムアウト数
LOCK_ESCALS	BIGINT	lock_escals ロック・エスカレーション数
FCM_SENDS_TOTAL	BIGINT	fcm_sends_total - FCM 送信の合計
FCM_RECVS_TOTAL	BIGINT	fcm_recvs_total - FCM 受信の合計

表 31. 作業単位イベント・モニターに戻される情報: 表名: UOW_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
FCM_SEND_VOLUME	BIGINT	fcm_send_volume - FCM 送信ボリューム
FCM_RECV_VOLUME	BIGINT	fcm_recv_volume - FCM 受信ボリューム
FCM_MESSAGE_SENDS_TOTAL	BIGINT	fcm_message_sends_total - FCM メッセージ送信の合計
FCM_MESSAGE_RECVS_TOTAL	BIGINT	fcm_message_recvs_total - FCM メッセージ受信の合計
FCM_MESSAGE_SEND_VOLUME	BIGINT	fcm_message_send_volume - FCM メッセージ送信ボリューム
FCM_MESSAGE_RECV_VOLUME	BIGINT	fcm_message_recv_volume - FCM メッセージ受信ボリューム
FCM_TQ_SENDS_TOTAL	BIGINT	fcm_tq_sends_total - FCM 表キュー送信の合計
FCM_TQ_RECVS_TOTAL	BIGINT	fcm_tq_recvs_total - FCM 表キュー受信の合計
FCM_TQ_SEND_VOLUME	BIGINT	fcm_tq_send_volume - FCM 表キュー送信ボリューム
FCM_TQ_RECV_VOLUME	BIGINT	fcm_tq_recv_volume - FCM 表キュー受信ボリューム
TQ_TOT_SEND_SPILLS	BIGINT	tq_tot_send_spills オーバーフローした表キュー・バッファーの合計数
TCPIP_SEND_VOLUME	BIGINT	tcpip_send_volume - TCP/IP 送信ボリューム
TCPIP_RECV_VOLUME	BIGINT	tcpip_recv_volume - TCP/IP 受信ボリューム
IPC_SEND_VOLUME	BIGINT	ipc_send_volume - プロセス間通信の送信ボリューム
IPC_RECV_VOLUME	BIGINT	ipc_recv_volume - プロセス間通信の受信ボリューム
POST_THRESHOLD_SORTS	BIGINT	post_threshold_sorts ポストしきい値ソート
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts ポスト共有しきい値ソート
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
AUDIT_EVENTS_TOTAL	BIGINT	audit_events_total - 監査イベントの合計
ACT_REJECTED_TOTAL	BIGINT	act_rejected_total - リジェクトされたアクティビティーの合計
ACT_ABORTED_TOTAL	BIGINT	act_aborted_total - 異常終了したアクティビティーの合計
TOTAL_SORTS	BIGINT	total_sorts ソート合計
TOTAL_ROUTINE_TIME	BIGINT	total_routine_time - 合計ルーチン時間
TOTAL_COMPILE_PROC_TIME	BIGINT	total_compile_proc_time - コンパイル処理時間の合計
TOTAL_COMPILATIONS	BIGINT	total_compilations - 合計コンパイル数
TOTAL_COMPILE_TIME	BIGINT	total_compile_time - 合計コンパイル時間
TOTAL_IMPLICIT_COMPILATIONS	BIGINT	total_implicit_compilations - 暗黙的なコンパイル数の合計
TOTAL_IMPLICIT_COMPILE_TIME	BIGINT	total_implicit_compile_time - 暗黙的なコンパイル時間の合計
TOTAL_RUNSTATS_PROC_TIME	BIGINT	total_runstats_proc_time - ランタイム統計処理時間の合計
TOTAL_RUNSTATS	BIGINT	total_runstats - ランタイム統計の合計

表 31. 作業単位イベント・モニターに戻される情報: 表名: UOW_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_RUNSTATS_TIME	BIGINT	total_runstats_time - ランタイム統計時間の合計
TOTAL_REORG_PROC_TIME	BIGINT	total_reorg_proc_time - 再編成処理時間の合計
TOTAL_REORGS	BIGINT	total_reorgs - 再編成の合計
TOTAL_REORG_TIME	BIGINT	total_reorg_time - 合計再編成時間
TOTAL_LOAD_PROC_TIME	BIGINT	total_load_proc_time - ロード処理時間の合計
TOTAL_LOADS	BIGINT	total_loads - ロード合計
TOTAL_LOAD_TIME	BIGINT	total_load_time - 合計ロード時間
TOTAL_SECTION_PROC_TIME	BIGINT	total_section_proc_time - セクション処理時間の合計
TOTAL_APP_SECTION_EXECUTIONS	BIGINT	total_app_section_executions - アプリケーションのセクション実行数の合計
TOTAL_SECTION_TIME	BIGINT	total_section_time - 合計セクション時間
TOTAL_COMMIT_PROC_TIME	BIGINT	total_commit_proc_time - コミット処理時間の合計
TOTAL_APP_COMMITS	BIGINT	total_app_commits - アプリケーションのコミットの合計数
TOTAL_COMMIT_TIME	BIGINT	total_commit_time - 合計コミット時間
TOTAL_ROLLBACK_PROC_TIME	BIGINT	total_rollback_proc_time - ロールバック処理時間の合計
TOTAL_APP_ROLLBACKS	BIGINT	total_app_rollbacks - アプリケーションの合計ロールバック数
TOTAL_ROLLBACK_TIME	BIGINT	total_rollback_time - 合計ロールバック時間
TOTAL_ROUTINE_USER_CODE_TIME	BIGINT	total_routine_user_code_time - ルーチンのユーザー・コード時間の合計
THRESH_VIOLATIONS	BIGINT	thresh_violations - しきい値違反の回数
NUM_LW_THRESH_EXCEEDED	BIGINT	num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数
TOTAL_ROUTINE_INVOCATIONS	BIGINT	total_routine_invocations - ルーチンの合計呼び出し数
INT_COMMITS	BIGINT	int_commits 内部コミット数
INT_ROLLBACKS	BIGINT	int_rollbacks 内部ロールバック数
CAT_CACHE_INSERTS	BIGINT	cat_cache_inserts カタログ・キャッシュ挿入数
CAT_CACHE_LOOKUPS	BIGINT	cat_cache_lookups カタログ・キャッシュ検索
PKG_CACHE_INSERTS	BIGINT	pkg_cache_inserts パッケージ・キャッシュ挿入
PKG_CACHE_LOOKUPS	BIGINT	pkg_cache_lookups パッケージ・キャッシュ検索
ACT_RQSTS_TOTAL	BIGINT	act_rqsts_total - アクティビティー要求の合計数
TOTAL_ACT_WAIT_TIME	BIGINT	total_act_wait_time - 合計アクティビティー待機時間
TOTAL_ACT_TIME	BIGINT	total_act_time - 合計アクティビティー時間
LOCK_WAIT_TIME_GLOBAL	BIGINT	lock_wait_time_global - グローバル・ロック待機時間
LOCK_WAITS_GLOBAL	BIGINT	lock_waits_global - グローバル・ロック待機
RECLAIM_WAIT_TIME	BIGINT	reclaim_wait_time - 再利用待機時間

表 31. 作業単位イベント・モニターに戻される情報: 表名: UOW_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
SPACEMAPPAGE_RECLAIM_WAIT_TIME	BIGINT	spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間
LOCK_TIMEOUTS_GLOBAL	BIGINT	lock_timeouts_global - グローバル・ロック・タイムアウト
LOCK_ESCALS_MAXLOCKS	BIGINT	lock_escals_maxlocks - maxlocks ロック・エスカレーション数
LOCK_ESCALS_LOCKLIST	BIGINT	lock_escals_locklist - locklist ロック・エスカレーション数
LOCK_ESCALS_GLOBAL	BIGINT	lock_escals_global - グローバル・ロック・エスカレーション数
CF_WAIT_TIME	BIGINT	cf_wait_time - クラスター・キャッシング機能待機時間
CF_WAITS	BIGINT	cf_waits - クラスター・キャッシング機能 DB2 pureScale サーバーの待機回数
POOL_DATA_GBP_L_READS	BIGINT	pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り
POOL_DATA_GBP_P_READS	BIGINT	pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り
POOL_DATA_LBP_PAGES_FOUND	BIGINT	pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ
POOL_DATA_GBP_INVALID_PAGES	BIGINT	pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ
POOL_INDEX_GBP_L_READS	BIGINT	pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り
POOL_INDEX_GBP_P_READS	BIGINT	pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り
POOL_INDEX_LBP_PAGES_FOUND	BIGINT	pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ
POOL_INDEX_GBP_INVALID_PAGES	BIGINT	pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ
POOL_XDA_GBP_L_READS	BIGINT	pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求
POOL_XDA_GBP_P_READS	BIGINT	pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求
POOL_XDA_LBP_PAGES_FOUND	BIGINT	pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ
POOL_XDA_GBP_INVALID_PAGES	BIGINT	pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ
EVMON_WAIT_TIME	BIGINT	evmon_wait_time - イベント・モニターの待機時間
EVMON_WAITS_TOTAL	BIGINT	evmon_waits_total - イベント・モニター合計待機回数
TOTAL_EXTENDED_LATCH_WAIT_TIME	BIGINT	total_extended_latch_wait_time - 拡張ラッチの合計待機時間
TOTAL_EXTENDED_LATCH_WAITS	BIGINT	total_extended_latch_waits - 拡張ラッチの合計待機回数
TOTAL_STATS_FABRICATIONS	BIGINT	total_stats_fabrications - 統計作成の合計回数

表 31. 作業単位イベント・モニターに戻される情報: 表名: UOW_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_STATS_FABRICATION_TIME	BIGINT	total_stats_fabrication_time - 統計作成の合計時間
TOTAL_SYNC_RUNSTATS_PROC_TIME	BIGINT	total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間
TOTAL_SYNC_RUNSTATS	BIGINT	total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数
TOTAL_SYNC_RUNSTATS_TIME	BIGINT	total_sync_runstats_time - 同期 RUNSTATS の合計時間
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間
POOL_QUEUED_ASYNC_DATA_REQS	BIGINT	pool_queued_async_data_reqs - データ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX_REQS	BIGINT	pool_queued_async_index_reqs - 索引プリフェッチ要求
POOL_QUEUED_ASYNC_XDA_REQS	BIGINT	pool_queued_async_xda_reqs - XDA プリフェッチ要求
POOL_QUEUED_ASYNC_OTHER_REQS	BIGINT	pool_queued_async_other_reqs - 非プリフェッチの要求
POOL_QUEUED_ASYNC_DATA_PAGES	BIGINT	pool_queued_async_data_pages - データ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX_PAGES	BIGINT	pool_queued_async_index_pages - 索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_XDA_PAGES	BIGINT	pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求
POOL_FAILED_ASYNC_DATA_REQS	BIGINT	pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求
POOL_FAILED_ASYNC_INDEX_REQS	BIGINT	pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求
POOL_FAILED_ASYNC_XDA_REQS	BIGINT	pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求
POOL_FAILED_ASYNC_OTHER_REQS	BIGINT	pool_failed_async_other_reqs - 失敗した非プリフェッチの要求
APP_ACT_COMPLETED_TOTAL	BIGINT	app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数
APP_ACT_ABORTED_TOTAL	BIGINT	app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数
APP_ACT_REJECTED_TOTAL	BIGINT	app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数
TOTAL_PEDS	BIGINT	total_peds - partial early distinct の合計回数
DISABLED_PEDS	BIGINT	1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』
POST_THRESHOLD_PEDS	BIGINT	post_threshold_peds - partial early distinct しきい値
TOTAL_PEAS	BIGINT	total_peas - partial early aggregation の合計回数

表 31. 作業単位イベント・モニターに戻される情報: 表名: UOW_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
POST_THRESHOLD_PEAS	BIGINT	post_threshold_peas - partial early aggregation しきい値
TQ_SORT_HEAP_REQUESTS	BIGINT	tq_sort_heap_requests - 表キュー・ソート・ヒープ要求
TQ_SORT_HEAP_REJECTIONS	BIGINT	tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否
TOTAL_CONNECT_REQUESTS	BIGINT	total_connect_requests - 接続要求またはユーザー切り替え要求
TOTAL_CONNECT_REQUEST_TIME	BIGINT	total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間
TOTAL_CONNECT _AUTHENTICATIONS	BIGINT	total_connect_authentications - 実行された接続またはユーザー切り替え認証
PREFETCH_WAIT_TIME	BIGINT	prefetch_wait_time プリフェッチ待ち時間
PREFETCH_WAITS	BIGINT	prefetch_waits - プリフェッチャーの待機カウント
POOL_DATA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1288 ページの『pool_data_gbp_indep_pages_found _in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
POOL_INDEX_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1326 ページの『pool_index_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
POOL_XDA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1393 ページの『pool_xda_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』

表 32. ロック・イベント・モニターに戻される情報: デフォルトの表名: UOW_PACKAGE_LIST_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
APPLICATION_ID	VARCHAR(128)	842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
INVOCATION_ID	INTEGER	invocation_id - 呼び出し ID
MEMBER	SMALLINT	member - データベース・メンバー
NESTING_LEVEL	INTEGER	nesting_level - ネスト・レベル
PACKAGE_ELAPSED_TIME	BIGINT	package_elapsed_time - パッケージ経過時間

表 32. ロック・イベント・モニターに戻される情報: デフォルトの表名: UOW_PACKAGE_LIST_evmon-name (続き)

列名	データ・タイプ	説明
PACKAGE_ID	BIGINT	package_id - パッケージ ID
ROUTINE_ID	INTEGER	routine_id - ルーチン ID
UOW_ID	INTEGER	uow_id 作業単位 ID

表 33. ロック・イベント・モニターに戻される情報: デフォルトの表名: UOW_EXECUTABLE_LIST_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
APPLICATION_ID	VARCHAR(128)	842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
EXECUTABLE_ID	VARCHAR(32)	executable_id 実行可能 ID
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCK_WAITS	BIGINT	lock_waits ロック待機数
MEMBER	SMALLINT	member - データベース・メンバー
NUM_EXECUTIONS	BIGINT	num_executions ステートメント実行回数
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts ポスト共有しきい値ソート
POST_THRESHOLD_SORTS	BIGINT	post_threshold_sorts ポストしきい値ソート
ROWS_READ	BIGINT	rows_read 読み取り行数
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
TOTAL_ACT_TIME	BIGINT	total_act_time - 合計アクティビティー時間
TOTAL_ACT_WAIT_TIME	BIGINT	total_act_wait_time - 合計アクティビティー待機時間
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
TOTAL_SORTS	BIGINT	total_sorts ソート合計
UOW_ID	INTEGER	uow_id 作業単位 ID

表 34. 作業単位イベント・モニターに戻される情報 : デフォルトの表名: CONTROL_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号

作業単位イベント・モニターの場合に EVMON_FORMAT_UE_TO_TABLES によってリレーショナル表に書き込まれる情報:

EVMON_FORMAT_UE_TO_TABLES 表関数から作業単位イベント・モニター用に書き込まれる情報。これは、`sqllib/misc/DB2EvmonUOW.xsd` ファイルにも記載されています。

表 35. 作業単位イベント・モニターに戻される情報: 表名: UOW_EVENT

列名	データ・タイプ	説明
EVENT_ID	BIGINT NOT NULL	event_id - イベント ID モニター・エレメント
TYPE	VARCHAR(128) NOT NULL	
EVENT_TIMESTAMP	TIMESTAMP NOT NULL	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー
COORD_MEMBER	SMALLINT	coord_member - コーディネーター・メンバー
COMPLETION_STATUS	VARCHAR(128)	completion_status 完了状況
START_TIME	TIMESTAMP	start_time イベント開始時刻
STOP_TIME	TIMESTAMP	stop_time イベント停止時刻
WORKLOAD_NAME	VARCHAR(128)	workload_name - ワークロード名
WORKLOAD_ID	INTEGER	workload_id ワークロード ID
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name サービス・スーパークラス名
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name サービス・サブクラス名
SERVICE_CLASS_ID	INTEGER	service_class_id サービス・クラス ID
UOW_ID	INTEGER	uow_id 作業単位 ID

表 35. 作業単位イベント・モニターに戻される情報: 表名: UOW_EVENT (続き)

列名	データ・タイプ	説明
WORKLOAD_OCCURRENCE_ID	INTEGER	workload_occurrence_id ワークロード・オカレンス ID
CONNECTION_TIME	TIMESTAMP	924 ページの『connection_start_time - 接続開始時刻 : モニター・エレメント』
MEMBER_ACTIVATION_TIME	TIMESTAMP	965 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』
APPLICATION_ID	VARCHAR(128)	842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル
APPLICATION_NAME	VARCHAR(128)	847 ページの『appl_name アプリケーション名 : モニター・エレメント』
SYSTEM_AUTHID	VARCHAR(128)	1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』
SESSION_AUTHID	VARCHAR(128)	1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』
CLIENT_PLATFORM	VARCHAR(12)	client_platform クライアント・オペレーティング・プラットフォーム
CLIENT_PID	BIGINT	client_pid クライアント・プロセス ID
CLIENT_PRODUCT_ID	VARCHAR(128)	900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』
CLIENT_PROTOCOL	VARCHAR(10)	client_protocol クライアント通信プロトコル
CLIENT_HOSTNAME	VARCHAR(255)	client_hostname - クライアントのホスト名
CLIENT_PORT_NUMBER	INTEGER	client_port_number - クライアントのポート番号
CLIENT_WRKSTNNAME	VARCHAR(255)	client_wrkstnname - クライアント・ワークステーション名
CLIENT_ACCTNG	VARCHAR(255)	client_acctng - クライアント・アカウンティング・ストリング
CLIENT_USERID	VARCHAR(255)	client_userid - クライアントのユーザー ID
CLIENT_APPLNAME	VARCHAR(255)	client_applname - クライアント・アプリケーション名

表 35. 作業単位イベント・モニターに戻される情報: 表名: UOW_EVENT (続き)

列名	データ・タイプ	説明
LOCAL_TRANSACTION_ID	VARCHAR(16)	1126 ページの『local_transaction_id - ローカル・トランザクション ID のモニター・エレメント』
GLOBAL_TRANSACTION_ID	VARCHAR(40)	1061 ページの『global_transaction_id - グローバル・トランザクション ID のモニター・エレメント』
UOW_LOG_SPACE_USED	BIGINT	uow_log_space_used 使用されている作業単位ログ・スペース
PACKAGE_LIST_SIZE	INTEGER	1248 ページの『package_list_size - パッケージ・リスト・サイズのモニター・エレメント』
PACKAGE_LIST_EXCEEDED	CHAR(3)	package_list_exceeded - パッケージ・リストの超過
EXECUTABLE_LIST_SIZE	BIGINT	1021 ページの『executable_list_size - 実行可能リスト・サイズのモニター・エレメント』
EXECUTABLE_LIST_TRUNCATED	CHAR(3)	1022 ページの『executable_list_truncated - 実行可能リスト切り捨てのモニター・エレメント』
METRICS	BLOB(1M)	メトリック関連のモニター・エレメントが含まれる XML 文書。この文書内のメトリックは、このトピックで後ほど取り上げられる UOW_METRICS 表で説明されているメトリックと同じです。詳しくは、 20 ページの『モニター・データを XML 文書で返すインターフェース』を参照してください。
INTRA_PARALLEL_STATE	VARCHAR(3)	intra_parallel_state - パーティション内並列処理の現行状態
MON_INTERVAL_ID	BIGINT	mon_interval_id - モニター間隔 ID

表 36. 作業単位イベント・モニターに戻される情報: 表名: UOW_PACKAGE_LIST

列名	データ・タイプ	説明
MEMBER	SMALLINT	member - データベース・メンバー
UOW_ID	INTEGER	uow_id 作業単位 ID
APPLICATION_ID	VARCHAR(128)	842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
PACKAGE_ID	BIGINT	package_id - パッケージ ID
NESTING_LEVEL	INTEGER	nesting_level - ネスト・レベル
ROUTINE_ID	BIGINT	routine_id - ルーチン ID

表 36. 作業単位イベント・モニターに戻される情報: 表名: UOW_PACKAGE_LIST (続き)

列名	データ・タイプ	説明
INVOCATION_ID	INTEGER	invocation_id - 呼び出し ID
PACKAGE_ELAPSED_TIME	BIGINT	package_elapsed_time - パッケージ経過時間

表 37. 作業単位イベント・モニターに戻される情報: 表名: UOW_EXECUTABLE_LIST

列名	データ・タイプ	説明
MEMBER	SMALLINT	member - データベース・メンバー
UOW_ID	INTEGER	uow_id 作業単位 ID
APPLICATION_ID	VARCHAR(128)	842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
EXECUTABLE_ID	VARCHAR(32) FOR BIT DATA	executable_id 実行可能 ID
NUM_EXECUTIONS	BIGINT	num_executions ステートメント実行回数
ROWS_READ	BIGINT	rows_read 読み取り行数
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
TOTAL_ACT_TIME	BIGINT	total_act_time - 合計アクティビティー時間
TOTAL_ACT_WAIT_TIME	BIGINT	total_act_wait_time - 合計アクティビティー待機時間
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCK_WAITS	BIGINT	lock_waits ロック待機数
TOTAL_SORTS	BIGINT	total_sorts ソート合計
POST_THRESHOLD_SORTS	BIGINT	post_threshold_sorts ポストしきい値ソート
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts ポスト共有しきい値ソート
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。

列名	データ・タイプ	説明
MEMBER	SMALLINT	member - データベース・メンバー
UOW_ID	INTEGER	uow_id 作業単位 ID
APPLICATION_ID	VARCHAR(128)	842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
ACT_ABORTED_TOTAL	BIGINT	act_aborted_total - 異常終了したアクティビティーの合計
ACT_COMPLETED_TOTAL	BIGINT	act_completed_total - 完了したアクティビティーの合計

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
ACT_REJECTED_TOTAL	BIGINT	act_rejected_total - リジェクトされたアクティビティーの合計
AGENT_WAIT_TIME	BIGINT	agent_wait_time - エージェント待機時間
AGENT_WAITS_TOTAL	BIGINT	agent_waits_total - エージェント待機の合計
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
CLIENT_IDLE_WAIT_TIME	BIGINT	client_idle_wait_time - クライアントのアイドル待機時間
DEADLOCKS	BIGINT	deadlocks デッドロック検出数
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求
FCM_RECV_VOLUME	BIGINT	fcm_recv_volume - FCM 受信ボリューム
FCM_RECVS_TOTAL	BIGINT	fcm_recvs_total - FCM 受信の合計
FCM_SEND_VOLUME	BIGINT	fcm_send_volume - FCM 送信ボリューム
FCM_SENDS_TOTAL	BIGINT	fcm_sends_total - FCM 送信の合計
FCM_RECV_WAIT_TIME	BIGINT	fcm_recv_wait_time - FCM 受信待機時間
FCM_SEND_WAIT_TIME	BIGINT	fcm_send_wait_time - FCM 送信待機時間
IPC_RECV_VOLUME	BIGINT	ipc_recv_volume - プロセス間通信の受信ボリューム
IPC_RECV_WAIT_TIME	BIGINT	ipc_recv_wait_time - プロセス間通信受信待機時間
IPC_RECVS_TOTAL	BIGINT	ipc_recvs_total - プロセス間通信受信の合計
IPC_SEND_VOLUME	BIGINT	ipc_send_volume - プロセス間通信の送信ボリューム
IPC_SEND_WAIT_TIME	BIGINT	ipc_send_wait_time - プロセス間通信送信待機時間
IPC_SENDS_TOTAL	BIGINT	ipc_sends_total - プロセス間通信送信の合計
LOCK_ESCALS	BIGINT	lock_escals ロック・エスカレーション数
LOCK_TIMEOUTS	BIGINT	lock_timeouts ロック・タイムアウト数
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCK_WAITS	BIGINT	lock_waits ロック待機数
LOG_BUFFER_WAIT_TIME	BIGINT	log_buffer_wait_time - ログ・バッファー待機時間
NUM_LOG_BUFFER_FULL	BIGINT	num_log_buffer_full フル・ログ・バッファーの回数
LOG_DISK_WAIT_TIME	BIGINT	log_disk_wait_time - ログ・ディスク待機時間
LOG_DISK_WAITS_TOTAL	BIGINT	log_disk_waits_total - ログ・ディスク待機の合計
RQSTS_COMPLETED_TOTAL	BIGINT	rqsts_completed_total - 完了した要求の合計
ROWS_MODIFIED	BIGINT	rows_modified 変更行数
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_RETURNED	BIGINT	rows_returned 戻り行数
TCPIP_RECV_VOLUME	BIGINT	tcpip_recv_volume - TCP/IP 受信ボリューム
TCPIP_SEND_VOLUME	BIGINT	tcpip_send_volume - TCP/IP 送信ボリューム
TCPIP_RECV_WAIT_TIME	BIGINT	tcpip_recv_wait_time - TCP/IP 受信待機時間
TCPIP_RECVS_TOTAL	BIGINT	tcpip_recvs_total - TCP/IP 受信の合計
TCPIP_SEND_WAIT_TIME	BIGINT	tcpip_send_wait_time - TCP/IP 送信待機時間

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
TCPIP_SENDS_TOTAL	BIGINT	tcpip_sends_total - TCP/IP 送信の合計
TOTAL_APP_RQST_TIME	BIGINT	total_app_rqst_time - 合計アプリケーション要求時間
TOTAL_RQST_TIME	BIGINT	total_rqst_time - 合計要求時間
WLM_QUEUE_TIME_TOTAL	BIGINT	wlm_queue_time_total - ワークロード・マネージャー合計キュー時間
WLM_QUEUE_ASSIGNMENTS _TOTAL	BIGINT	wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
TOTAL_WAIT_TIME	BIGINT	total_wait_time - 合計待機時間
APP_RQSTS_COMPLETED_TOTAL	BIGINT	app_rqsts_completed_total - 完了したアプリケーション要求の合計
TOTAL_SECTION_SORT_TIME	BIGINT	total_section_sort_time - セクションのソート時間の合計
TOTAL_SECTION_SORT_PROC _TIME	BIGINT	total_section_sort_proc_time - セクションのソート処理時間の合計
TOTAL_SECTION_SORTS	BIGINT	total_section_sorts - セクションのソートの合計
TOTAL_SORTS	BIGINT	total_sorts ソート合計
POST_THRESHOLD_SORTS	BIGINT	post_threshold_sorts ポストしきい値ソート
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts ポスト共有しきい値ソート
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
TOTAL_COMPILE_TIME	BIGINT	total_compile_time - 合計コンパイル時間
TOTAL_COMPILE_PROC_TIME	BIGINT	total_compile_proc_time - 合計コンパイル処理時間
TOTAL_COMPILATIONS	BIGINT	total_compilations - 合計コンパイル数
TOTAL_IMPLICIT_COMPILE_TIME	BIGINT	total_implicit_compile_time - 暗黙コンパイルの合計時間
TOTAL_IMPLICIT_COMPILE_PROC _TIME	BIGINT	total_implicit_compile_proc_time - 暗黙コンパイルの合計処理時間
TOTAL_IMPLICIT_COMPILATIONS	BIGINT	total_implicit_compilations - 合計暗黙コンパイル数
TOTAL_SECTION_TIME	BIGINT	total_section_time - 合計セクション時間
TOTAL_SECTION_PROC_TIME	BIGINT	total_section_proc_time - 合計セクション処理時間
TOTAL_APP_SECTION _EXECUTIONS	BIGINT	total_app_section_executions - アプリケーションのセクション実行数の合計
TOTAL_ACT_TIME	BIGINT	total_act_time - 合計アクティビティー時間
TOTAL_ACT_WAIT_TIME	BIGINT	total_act_wait_time - 合計アクティビティー待機時間
ACT_RQSTS_TOTAL	BIGINT	act_rqsts_total - 合計アクティビティー要求数

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
TOTAL_ROUTINE_TIME	BIGINT	total_routine_time - 合計ルーチン時間
TOTAL_ROUTINE_INVOCATIONS	BIGINT	total_routine_invocations - 合計ルーチン呼び出し数
TOTAL_COMMIT_TIME	BIGINT	total_commit_time - 合計コミット時間
TOTAL_COMMIT_PROC_TIME	BIGINT	total_commit_proc_time - 合計コミット処理時間
TOTAL_APP_COMMITS	BIGINT	total_app_commits - アプリケーションの合計コミット数
INT_COMMITS	BIGINT	int_commits 内部コミット数
TOTAL_ROLLBACK_TIME	BIGINT	total_rollback_time - 合計ロールバック時間
TOTAL_ROLLBACK_PROC_TIME	BIGINT	total_rollback_proc_time - 合計ロールバック処理時間
TOTAL_APP_ROLLBACKS	BIGINT	total_app_rollbacks - アプリケーションの合計ロールバック数
INT_ROLLBACKS	BIGINT	int_rollbacks 内部ロールバック数
TOTAL_RUNSTATS_TIME	BIGINT	total_runstats_time - 実行時統計の合計時間
TOTAL_RUNSTATS_PROC_TIME	BIGINT	total_runstats_proc_time - 実行時統計の合計処理時間
TOTAL_RUNSTATS	BIGINT	total_runstats - 合計実行時統計数
TOTAL_REORG_TIME	BIGINT	total_reorg_time - 合計再編成時間
TOTAL_REORG_PROC_TIME	BIGINT	total_reorg_proc_time - 合計再編成処理時間
TOTAL_REORGS	BIGINT	total_reorgs - 合計再編成数
TOTAL_LOAD_TIME	BIGINT	total_load_time - 合計ロード時間
TOTAL_LOAD_PROC_TIME	BIGINT	total_load_proc_time - 合計ロード処理時間
TOTAL_LOADS	BIGINT	total_loads - 合計ロード数
CAT_CACHE_INSERTS	BIGINT	cat_cache_inserts カタログ・キャッシュ挿入数
CAT_CACHE_LOOKUPS	BIGINT	cat_cache_lookups カタログ・キャッシュ検索
PKG_CACHE_INSERTS	BIGINT	pkg_cache_inserts パッケージ・キャッシュ挿入
PKG_CACHE_LOOKUPS	BIGINT	pkg_cache_lookups パッケージ・キャッシュ検索
THRESH_VIOLATIONS	BIGINT	thresh_violations - しきい値違反の合計数
NUM_LW_THRESH_EXCEEDED	BIGINT	num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数
FCM_TQ_RECV_WAIT_TIME	BIGINT	fcm_tq_recv_wait_time - FCM 表キュー受信待機時間
FCM_MESSAGE_RECV_WAIT_TIME	BIGINT	fcm_message_recv_wait_time - FCM メッセージの受信待機時間
FCM_TQ_SEND_WAIT_TIME	BIGINT	fcm_tq_send_wait_time - FCM 表キュー送信待機時間

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
FCM_MESSAGE_SEND_WAIT_TIME	BIGINT	fcm_message_send_wait_time - FCM メッセージの送信待機時間
AUDIT_FILE_WRITE_WAIT_TIME	BIGINT	audit_file_write_wait_time - 監査ファイル書き込み待機時間
AUDIT_FILE_WRITES_TOTAL	BIGINT	audit_file_writes_total - 書き込まれた監査ファイルの合計
AUDIT_SUBSYSTEM_WAIT_TIME	BIGINT	audit_subsystem_wait_time - 監査サブシステム待機時間
AUDIT_SUBSYSTEM_WAITS_TOTAL	BIGINT	audit_subsystem_waits_total - 監査サブシステム待機の合計
DIAGLOG_WRITE_WAIT_TIME	BIGINT	diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間
DIAGLOG_WRITES_TOTAL	BIGINT	diaglog_writes_total - 診断ログ・ファイル書き込みの合計
FCM_MESSAGE_SENDS_TOTAL	BIGINT	fcm_message_sends_total - FCM メッセージ送信の合計
FCM_MESSAGE_RECVS_TOTAL	BIGINT	fcm_message_recvs_total - FCM メッセージ受信の合計
FCM_MESSAGE_SEND_VOLUME	BIGINT	fcm_message_send_volume - FCM メッセージ送信ボリューム
FCM_MESSAGE_RECV_VOLUME	BIGINT	fcm_message_recv_volume - FCM メッセージ受信ボリューム
FCM_TQ_SENDS_TOTAL	BIGINT	fcm_tq_sends_total - FCM 表キュー送信の合計
FCM_TQ_RECVS_TOTAL	BIGINT	fcm_tq_recvs_total - FCM 表キュー受信の合計
FCM_TQ_SEND_VOLUME	BIGINT	fcm_tq_send_volume - FCM 表キュー送信ボリューム
FCM_TQ_RECV_VOLUME	BIGINT	fcm_tq_recv_volume - FCM 表キュー受信ボリューム
TQ_TOT_SEND_SPILLS	BIGINT	tq_tot_send_spills オーバーフローした表キュー・バッファーの合計数
AUDIT_EVENTS_TOTAL	BIGINT	audit_events_total - 監査イベントの合計
TOTAL_ROUTINE_USER_CODE _PROC_TIME	BIGINT	total_routine_user_code_proc_time - ユーザー・コードのルーチンの合計処理時間
TOTAL_ROUTINE_USER_CODE _TIME	BIGINT	total_routine_user_code_time - ユーザー・コードのルーチンの合計時間
LOCK_WAITS_GLOBAL	BIGINT	lock_waits_global - グローバル・ロック待機
LOCK_WAIT_TIME_GLOBAL	BIGINT	lock_wait_time_global - グローバル・ロック待機時間
LOCK_TIMEOUTS_GLOBAL	BIGINT	lock_timeouts_global - グローバル・ロック・タイムアウト
LOCK_ESCALS_MAXLOCKS	BIGINT	lock_escals_maxlocks - maxlocks ロック・エスカレーション数

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
LOCK_ESCALS_LOCKLIST	BIGINT	lock_escals_locklist - locklist ロック・エスカレーション数
LOCK_ESCALS_GLOBAL	BIGINT	lock_escals_global - グローバル・ロック・エスカレーション数
RECLAIM_WAIT_TIME	BIGINT	reclaim_wait_time - 再利用待機時間
SPACEMAPPAGE_RECLAIM_WAIT _TIME	BIGINT	spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間
CF_WAITS	BIGINT	cf_waits - クラスター・キャッシング機能 DB2 pureScale サーバーの待機回数
CF_WAIT_TIME	BIGINT	cf_wait_time - クラスター・キャッシング機能待機時間
POOL_DATA_GBP_L_READS	BIGINT	pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り
POOL_DATA_GBP_P_READS	BIGINT	pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り
POOL_DATA_LBP_PAGES_FOUND	BIGINT	pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ
POOL_DATA_GBP_INVALID _PAGES	BIGINT	pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ
POOL_INDEX_GBP_L_READS	BIGINT	pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り
POOL_INDEX_GBP_P_READS	BIGINT	pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り
POOL_INDEX_LBP_PAGES_FOUND	BIGINT	pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ
POOL_INDEX_GBP_INVALID_PAGES	BIGINT	pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ
POOL_XDA_GBP_L_READS	BIGINT	pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求
POOL_XDA_GBP_P_READS	BIGINT	pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求
POOL_XDA_LBP_PAGES_FOUND	BIGINT	pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ
POOL_XDA_GBP_INVALID_PAGES	BIGINT	pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ
EVMON_WAIT_TIME	BIGINT	evmon_wait_time - イベント・モニターの待機時間
EVMON_WAITS_TOTAL	BIGINT	evmon_waits_total - イベント・モニター合計待機回数
TOTAL_EXTENDED_LATCH_WAIT _TIME	BIGINT	total_extended_latch_wait_time - 拡張ラッチの合計待機時間

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
TOTAL_EXTENDED_LATCH_WAITS	BIGINT	total_extended_latch_waits - 拡張ラッチの合計待機回数
TOTAL_STATS_FABRICATION_TIME	BIGINT	total_stats_fabrication_time - 統計作成の合計時間
TOTAL_STATS_FABRICATION _PROC_TIME	BIGINT	total_stats_fabrication_proc_time - 統計作成の合計処理時間
TOTAL_STATS_FABRICATIONS	BIGINT	total_stats_fabrications - 統計作成の合計回数
TOTAL_SYNC_RUNSTATS_TIME	BIGINT	total_sync_runstats_time - 同期 RUNSTATS の合計時間
TOTAL_SYNC_RUNSTATS_PROC _TIME	BIGINT	total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間
TOTAL_SYNC_RUNSTATS	BIGINT	total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間
TOTAL_PEDS	BIGINT	total_peds - partial early distinct の合計回数
DISABLED_PEDS	BIGINT	1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』
POST_THRESHOLD_PEDS	BIGINT	post_threshold_peds - partial early distinct しきい値
TOTAL_PEAS	BIGINT	total_peas - partial early aggregation の合計回数
POST_THRESHOLD_PEAS	BIGINT	post_threshold_peas - partial early aggregation しきい値
TQ_SORT_HEAP_REQUESTS	BIGINT	tq_sort_heap_requests - 表キュー・ソート・ヒープ要求
TQ_SORT_HEAP_REJECTIONS	BIGINT	tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否
POOL_QUEUED_ASYNC_DATA _REQS	BIGINT	pool_queued_async_data_reqs - データ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX _REQS	BIGINT	pool_queued_async_index_reqs - 索引プリフェッチ要求
POOL_QUEUED_ASYNC_XDA _REQS	BIGINT	pool_queued_async_xda_reqs - XDA プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _DATA_REQS	BIGINT	pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _INDEX_REQS	BIGINT	pool_queued_async_temp_index_reqs - TEMPORARY 表スペースの索引プリフェッチ要求

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
POOL_QUEUED_ASYNC_TEMP_XDA_REQS	BIGINT	pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求
POOL_QUEUED_ASYNC_OTHER_REQS	BIGINT	pool_queued_async_other_reqs - 非プリフェッチの要求
POOL_QUEUED_ASYNC_DATA_PAGES	BIGINT	pool_queued_async_data_pages - データ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX_PAGES	BIGINT	pool_queued_async_index_pages - 索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_XDA_PAGES	BIGINT	pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_DATA_PAGES	BIGINT	pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES	BIGINT	pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_XDA_PAGES	BIGINT	pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求
POOL_FAILED_ASYNC_DATA_REQS	BIGINT	pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求
POOL_FAILED_ASYNC_INDEX_REQS	BIGINT	pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求
POOL_FAILED_ASYNC_XDA_REQS	BIGINT	pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_DATA_REQS	BIGINT	pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_INDEX_REQS	BIGINT	pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_XDA_REQS	BIGINT	pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求
POOL_FAILED_ASYNC_OTHER_REQS	BIGINT	pool_failed_async_other_reqs - 失敗した非プリフェッチの要求
PREFETCH_WAIT_TIME	BIGINT	prefetch_wait_time プリフェッチ待ち時間
PREFETCH_WAITS	BIGINT	prefetch_waits - プリフェッチャーの待機カウント
APP_ACT_COMPLETED_TOTAL	BIGINT	app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数

表 38. 作業単位イベント・モニターに戻される情報 : 表名: UOW_METRICS。この表のメトリックは、UOW_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
APP_ACT_ABORTED_TOTAL	BIGINT	app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数
APP_ACT_REJECTED_TOTAL	BIGINT	app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数
TOTAL_CONNECT_REQUEST_TIME	BIGINT	total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間
TOTAL_CONNECT_REQUEST _PROC_TIME	BIGINT	total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間
TOTAL_CONNECT_REQUESTS	BIGINT	total_connect_requests - 接続要求またはユーザー切り替え要求
TOTAL_CONNECT _AUTHENTICATION_TIME	BIGINT	total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間
TOTAL_CONNECT _AUTHENTICATION_PROC _TIME	BIGINT	total_connect_authentication_proc_time - 接続認証の合計処理時間
TOTAL_CONNECT _AUTHENTICATIONS	BIGINT	total_connect_authentications - 実行された接続またはユーザー切り替え認証
POOL_DATA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1288 ページの『pool_data_gbp_indep_pages_found _in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
POOL_INDEX_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1326 ページの『pool_index_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
POOL_XDA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1393 ページの『pool_xda_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』
COMM_EXIT_WAIT_TIME	BIGINT	comm_exit_wait_time - 通信バッファー出口待機時間モニター・エレメント
COMM_EXIT_WAITS	BIGINT	comm_exit_waits - 通信バッファー出口待機回数モニター・エレメント

作業単位イベント・モニターの場合に EVMON_FORMAT_UE_TO_XML によって XML に書き込まれる情報:

EVMON_FORMAT_UE_TO_XML 表関数から作業単位イベント・モニター用に書き込まれる情報。これは、`sqllib/misc/DB2EvmonUOW.xsd` ファイルにも記載されています。

db2_uow_event

作業単位イベントについて記述するメイン・スキーマ

エレメント・コンテンツ: (255 ページの『completion_status』、 255 ページの『start_time』、 256 ページの『stop_time』、 256 ページの『connection_time』、 256 ページの『application_name』、 256 ページの『application_handle』、 256 ページの『application_id』、 257 ページの『uow_id』 、 257 ページの『workload_occurrence_id』、 257 ページの『coord_member』、 257 ページの『member_activation_time 』、 257 ページの『workload_name』、 257 ページの『workload_id』、 258 ページの『service_superclass_name』 { 0 または 1 個 (?)} 、 258 ページの『service_subclass_name』 { 0 または 1 個 (?)} 、 258 ページの『service_class_id』 { 0 または 1 個 (?)} 、 258 ページの『session_authid』 { 0 または 1 個 (?)} 、 258 ページの『system_authid』、 258 ページの『client_pid』、 259 ページの『client_product_id』、 259 ページの『client_platform』、 259 ページの『client_protocol』 {0 または 1 個 (?)} 、 259 ページの『client_userid』 { 0 または 1 個 (?)} 、 259 ページの『client_wrkstnname』 { 0 または 1 個 (?)} 、 259 ページの『client_applname』 { 0 または 1 個 (?)} 、 260 ページの『client_acctng』 { 0 または 1 個 (?)} 、 260 ページの『local_transaction_id 』、 260 ページの『global_transaction_id 』、 260 ページの『system_metrics』、 260 ページの『client_hostname』、 260 ページの『client_port_number』 、 261 ページの『uow_log_space_used』、 261 ページの『package_list』、 261 ページの『executable_list』、 261 ページの『intra_parallel_state』、 ANY コンテント (検証しない (skip)) {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			必須	
type				必須	
timestamp	xs:dateTime			必須	
メンバー				必須	
release	xs:long			必須	
mon_interval_id	xs:long			必須	
ANY ネーム・スペースの ANY 属性					

package_id

詳しくは、モニター・エレメント 1247 ページの『package_id - パッケージ ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 251 ページの『package_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

package_elapsed_time

詳しくは、モニター・エレメント 1247 ページの『package_elapsed_time - パッケージ経過時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 251 ページの『package_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

invocation_id

詳しくは、モニター・エレメント 1110 ページの『invocation_id - 呼び出し ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 251 ページの『package_entry』

エレメント・コンテント :

タイプ	ファセット
xs:int	

routine_id

詳しくは、モニター・エレメント 1460 ページの『routine_id - ルーチン ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 251 ページの『package_entry』

エレメント・コンテント :

タイプ	ファセット
xs:int	

nesting_level

詳しくは、モニター・エレメント 1205 ページの『nesting_level - ネスト・レベル : モニター・エレメント』 を参照してください。

これを含むエレメント: 251 ページの『package_entry』

エレメント・コンテント :

タイプ	ファセット
xs:int	

package_entry

これを含むエレメント: 『package_list_entries』

エレメント・コンテント: (249 ページの『package_id』、 250 ページの『package_elapsed_time』、 250 ページの『invocation_id』、 250 ページの『routine_id』、 250 ページの『nesting_level』、 ANY コンテント (検証しない (skip)) {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
ANY ネーム・スペースの ANY 属性					

package_list_size

これを含むエレメント: 261 ページの『package_list』

エレメント・コンテント :

タイプ	ファセット
xs:int	

package_list_exceeded

詳しくは、モニター・エレメント 1248 ページの『package_list_exceeded - パッケージ・リストの超過 : モニター・エレメント』 を参照してください。

これを含むエレメント: 261 ページの『package_list』

package_list_entries

これを含むエレメント: 261 ページの『package_list』

エレメント・コンテント: (『package_entry』 {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
ANY ネーム・スペースの ANY 属性					

executable_id

詳しくは、モニター・エレメント 1020 ページの『executable_id 実行可能 ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

num_executions

詳しくは、モニター・エレメント 1211 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

rows_read

詳しくは、モニター・エレメント 1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_cpu_time

詳しくは、モニター・エレメント 1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_act_time

詳しくは、モニター・エレメント 1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_act_wait_time

詳しくは、モニター・エレメント 1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_wait_time

詳しくは、モニター・エレメント 1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_waits

詳しくは、モニター・エレメント 1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_sorts

詳しくは、モニター・エレメント 1680 ページの『total_sorts - ソート合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 254 ページの『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

post_threshold_sorts

詳しくは、モニター・エレメント 1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』 を参照してください。

これを含むエレメント: 『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

post_shrthreshold_sorts

詳しくは、モニター・エレメント 1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』 を参照してください。

これを含むエレメント: 『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

sort_overflows

詳しくは、モニター・エレメント 1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』 を参照してください。

これを含むエレメント: 『executable_entry』

エレメント・コンテント :

タイプ	ファセット
xs:long	

executable_entry

これを含むエレメント: 255 ページの『executable_list_entries』

エレメント・コンテント : (251 ページの『executable_id』, 252 ページの『num_executions』, 252 ページの『rows_read』, 252 ページの『total_cpu_time』, 252 ページの『total_act_time』, 253 ページの『total_act_wait_time』, 253 ページの『lock_wait_time』, 253 ページの『lock_waits』, 253 ページの『total_sorts』, 『post_threshold_sorts』, 『post_shrthreshold_sorts』, 『sort_overflows』, ANY コンテント (検証しない (skip)) {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
ANY ネーム・スペースの ANY 属性					

executable_list_size

これを含むエレメント: 261 ページの『executable_list』

エレメント・コンテント :

タイプ	ファセット
xs:int	

executable_list_truncated

これを含むエレメント: 261 ページの『executable_list』

executable_list_entries

これを含むエレメント: 261 ページの『executable_list』

エレメント・コンテント: (254 ページの『executable_entry』 {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
ANY ネーム・スペースの ANY 属性					

completion_status

作業単位の完了状況。可能な値は、
UNKNOWN、COMMIT、ROLLBACK、GLOBAL_COMMIT、GLOBAL ROLLBACK、XA_END、XA_PREPARE です。

これを含むエレメント: 249 ページの『db2_uow_event』

start_time

作業単位の開始時刻。詳しくは、モニター・エレメント 1714 ページの『uow_start_time - 作業単位開始タイム・スタンプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

stop_time

作業単位の停止時刻。詳しくは、モニター・エレメント 1715 ページの『uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

connection_time

アプリケーションがデータベース・メンバーに接続した時刻。詳しくは、モニター・エレメント 924 ページの『conn_time データベース接続時刻 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

application_name

クライアントで実行中のアプリケーションの名前。データベースが識別できる名前です。詳しくは、モニター・エレメント 847 ページの『appl_name アプリケーション名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

application_handle

システム全体での、アプリケーションのユニーク ID。詳しくは、モニター・エレメント 824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

application_id

アプリケーションがデータベース・マネージャーでデータベースに接続すると、この ID が生成されます。詳しくは、モニター・エレメント 842 ページの『appl_id - アプリケーション ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

uow_id

このアクティビティー・レコードが適用される作業単位 ID。詳しくは、モニター・エレメント 1711 ページの『uow_id 作業単位 ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

workload_occurrence_id

このアクティビティー・レコードが適用されるワークロード・オカレンス ID。詳しくは、モニター・エレメント 1740 ページの『workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

coord_member

これを含むエレメント: 249 ページの『db2_uow_event』

member_activation_time

このデータベース・メンバーがアクティブにされた時刻。詳しくは、モニター・エレメント 965 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

workload_name

作業単位が完了したワークロードの名前。詳しくは、モニター・エレメント 1739 ページの『workload_name ワークロード名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

workload_id

作業単位が完了したワークロードのワークロード ID。詳しくは、モニター・エレメント 1738 ページの『workload_id ワークロード ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

service_superclass_name

作業単位が完了したサービス・スーパークラスの名前。詳しくは、モニター・エレメント 1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

service_subclass_name

作業単位が完了したサービス・サブクラスの名前。詳しくは、モニター・エレメント 1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

service_class_id

作業単位が完了したサービス・クラスのサービス・クラス ID。詳しくは、モニター・エレメント 1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

session_authid

モニターされているアプリケーションを呼び出したユーザーのセッション許可 ID。詳しくは、モニター・エレメント 1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

system_authid

モニターされているアプリケーションを呼び出したユーザーのシステム許可 ID。詳しくは、モニター・エレメント 1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

client_pid

クライアントによってレポートされるプロセス ID。詳しくは、モニター・エレメント 898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

client_product_id

クライアントの製品 ID。詳しくは、モニター・エレメント 900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

client_platform

クライアントのプラットフォーム。詳しくは、モニター・エレメント 899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:short			オプション	

client_protocol

クライアントの製品 ID。詳しくは、モニター・エレメント 901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

client_userid

トランザクション・マネージャーが生成してサーバーに提供したクライアント・ユーザー ID。詳しくは、モニター・エレメント 902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

client_wrkstnname

この接続で sqleseti API が発行された場合に、クライアントのシステムまたはワークステーションを示します。詳しくは、モニター・エレメント 903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

client_applname

この接続で sqleseti API が発行された場合に、トランザクションを実行中のサーバー・トランザクション・プログラムを示します。詳しくは、モニター・エレメント 894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

client_acctng

この接続で sqleseti API が発行された場合に、ロギングおよび診断の目的でターゲット・データベースに渡されたデータです。詳しくは、モニター・エレメント 893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

local_transaction_id

作業単位のローカル・トランザクション ID

これを含むエレメント: 249 ページの『db2_uow_event』

global_transaction_id

作業単位のグローバル・トランザクション ID

これを含むエレメント: 249 ページの『db2_uow_event』

system_metrics

作業単位のメトリック。

これを含むエレメント: 249 ページの『db2_uow_event』

client_hostname

クライアントのホスト名。詳しくは、モニター・エレメント 896 ページの『client_hostname - クライアント・ホスト名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

client_port_number

クライアントのポート番号。詳しくは、モニター・エレメント 900 ページの『client_port_number - クライアント・ポート番号 : モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント :

タイプ	ファセット
xs:int	

uow_log_space_used

この作業単位で使用されたログ・スペースの量。詳しくは、モニター・エレメント 1713 ページの『uow_log_space_used - 使用されている作業単位ログ・スペース: モニター・エレメント』 を参照してください。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

package_list

作業単位のパッケージ・リスト。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント: (251 ページの『package_list_size』, 251 ページの『package_list_exceeded』, 251 ページの『package_list_entries』, ANY コンテント (検証しない (skip)) {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
ANY ネーム・スペースの ANY 属性					

executable_list

作業単位の実行可能リスト。

これを含むエレメント: 249 ページの『db2_uow_event』

エレメント・コンテント: (255 ページの『executable_list_size』, 255 ページの『executable_list_truncated』, 255 ページの『executable_list_entries』, ANY コンテント (検証しない (skip)) {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
ANY ネーム・スペースの ANY 属性					

intra_parallel_state

作業単位の現行のパーティション内並列処理の状態。可能な値は YES および NO です。

これを含むエレメント: 249 ページの『db2_uow_event』

作業単位イベント・モニターのパッケージ・リスト情報:

作業単位イベント・モニターは、作業単位の中で使われているパッケージのリストを収集することができます。この情報を使用すると、アプリケーションの中で、期待される実行時間よりも長くかかっていると思われるストアード・プロシージャーを判別できます。

DB2 バージョン 9.7 のフィックスパック 1 以降、イベント・モニターで収集されるデータの中に、作業単位内で使われているパッケージについての情報を含めることができるようになりました。この情報は、作業単位が終了したときに、作業単位イベント・モニターに選択した出力オプションによって、未フォーマット・イベント表または UOW_PACKAGE_LIST_*evmon-name* 表 (*evmon-name* は、イベント・モニターに割り当てられた名前です) に、イベントに関連する他の情報と一緒に書き込まれます。

この情報のキャプチャーを制御するには、以下の 2 つの方法があります。

- CREATE または ALTER WORKLOAD ステートメントの COLLECT UNIT OF WORK DATA 節の PACKAGE LIST オプションは、特定の ワークロードに関するこの情報の収集を制御します。このオプションを指定した場合、CREATE または ALTER WORKLOAD ステートメントで指定されたワークロードの下で実行される作業単位についての情報 (パッケージ・リスト情報を含む) が、アクティブな作業単位イベント・モニターに送られます。
- **mon_uow_pkglist** 構成パラメーターを ON に設定すると、データ・サーバーで実行されるすべての 作業単位についてのパッケージ・リスト情報を、アクティブな作業単位イベント・モニターに送ることができます。

  注: パッケージ・リスト情報を収集するには、**mon_uow_data** を BASE に設定する必要もあります。

パッケージ・リストに関する以下のデータが収集されます。

パッケージ ID (1247 ページの『package_id - パッケージ ID : モニター・エレメント』)
パッケージを識別する固有の ID。

ネスト・レベル (1205 ページの『nesting_level - ネスト・レベル : モニター・エレメント』)
ステートメントが実行されていた時点で有効であったネストまたは再帰のレベル。ネストの各レベルは、ネストされた (または再帰可能な) ストアード・プロシージャー呼び出しまたはユーザー定義関数 (UDF) 呼び出しに対応します。

ルーチン ID (1460 ページの『routine_id - ルーチン ID : モニター・エレメント』)
固有なルーチン ID。アクティビティーがルーチンの一部ではない場合、ゼロが戻されます。

呼び出し ID (1110 ページの『invocation_id - 呼び出し ID : モニター・エレメント』)
ルーチンの 1 つの呼び出しを、作業単位内の同じネスト・レベルの他の呼び出しと区別する ID。その ID は特定のネスト・レベルに関して作業単位内で固有です。

パッケージ経過時間 (1247 ページの『package_elapsed_time - パッケージ経過時間 : モニター・エレメント』)
パッケージ内のセクションの実行にかかった経過時間。

パッケージ・リストに関して収集される情報のリストが示唆するように、各パッケージに関する情報だけでなく、パッケージ内のルーチンのそれぞれの呼び出しに関する情報もキャプチャーされます。

経過時間もまた追跡されます。特定の呼び出しに関して計算される時間は、パッケージ内でのセクションの最初の実行から始まり、データベース・マネージャーが別のパッケージに切り替える時点で終了します。経過時間が追跡される方法についてさらに確認するには、 266 ページの『例』を参照してください。

パッケージ・リストが未フォーマット・イベント表に書き込まれる方法

パッケージ・リスト情報の収集を使用可能にすると、作業単位イベント・モニターは、各作業単位に関して未フォーマット・イベント (UE) 表に 2 つのレコードを書き込みます。最初のレコードには、基本的な作業単位イベント・モニター・データが含まれます。次のレコードには、パッケージ・リスト情報が含まれます。

パッケージ・リスト情報は UE 表の BLOB 列に保管されます。表スペースのページ・サイズが 4k (デフォルト) の場合、32 項目から成るリストを 1 つのインライン BLOB として保管できます。パッケージ・リストに書き込むことのできる項目の数は **mon_pkglist_sz** 構成パラメーターによって制御されます。このパラメーターのデフォルトは 32 で、これは最大 32 個の項目をパッケージ・リストに含めることができるという意味です。パッケージ・リストに含めることのできる項目数を増やす場合には、イベント・モニター出力の保管に使われる UE 表を、より大きなページ・サイズを持つ表スペースに必ず作成してください。パッケージ・リストのサイズを 32 増やすごとに、表スペースのページ・サイズを 4k 増やす必要があることを想定してください。したがって、例えばパッケージ・リスト内の最大項目数を 64 にした場合、表スペースのページ・サイズを少なくとも 8k にする必要があります。表スペースのページ・サイズを増やさずに **mon_pkglist_sz** を増やした場合、パッケージ・リストは引き続き作成されますが、表の中で BLOB はインラインでは保管されません。このためパフォーマンスが影響を受ける可能性があります。

注: パッケージ・リスト情報を含んでいる BLOB がインラインで保管されるかどうかを判別するには、ADMIN_IS_INLINED 管理関数を使用できます。

通常の表へのパッケージ・リストの書き込み

イベント・モニターの出力に通常の表を使用する場合、パッケージ・リスト情報は、 133 ページの『uow_package_list 論理データ・グループ』 の一部としてキャプチャーされます。各作業単位が完了すると、1 つ以上の行が UOW_PACKAGE_LIST_*evmon-name* 表に追加されます。この表には、論理データ・グループ内のモニター・エレメントごとに 1 つの列があります。この表に追加される行数は、作業単位の一部として実行されたパッケージの数に依存します。ただし、この表に追加できる行数の上限は、 **mon_pkglist_sz** 構成パラメーターで制御されます。このパラメーターのデフォルトは 32 で、これは最大 32 個の項目をパッケージ・リストに含めることができるという意味です。パッケージ・リストに含むことのできる項目数を増やすには、 **mon_pkglist_sz** を増やしてください。

パッケージ・リスト出力

前述のように、作業単位イベント・モニターが UE 表に書き込む場合、イベント・モニターは、パッケージ情報を収集すると 2 つのレコードを UE 表に書き込みます。 UE 表のデータを表示するための各インターフェースは、 2 つの UE 表レコードに含まれている情報を表示するメカニズムを備えています。例えば **db2evmonfmt** ツールは各レコードの情報を 1 つのレポートに結合します。EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用すると、結合可能なリレーショナル表が生成されます。パッケージ・リスト情報は、UOW_PACKAGE_LIST 表に入っています。 EVMON_FORMAT_UE_TO_XML は、両方のレコードの情報を含む 1 つの XML 文書を生成します。詳しくは、 222 ページの『作業単位イベント・モニターによりキャプチャーされたイベント・データへのアクセス』を参照してください。

イベント・モニターが直接リレーショナル表に書き込む場合、パッケージ・リスト情報は UOW_PACKAGE_LIST_*evmon-name* 表に書き込まれます。

注: パーティション・データベース環境では、コーディネーター・エージェント によって生成された作業単位イベントでのみパッケージ・リストが報告され、これは、そのエージェントが各パッケージ内で費やした時間のみを反映します。他のパーティションにある他のエージェントがこれらのパッケージ内で費やした時間は、まったく反映されません。

265 ページの図 5 は、作業単位イベント・モニターによって生成され、**db2evmonfmt** ツールによってフォーマット設定された情報を示しています。

```
------------------------------------------------------
Event ID               : 12
Event Type             : UOW
Event Timestamp        : 2009-12-08-14.44.39.162707
Member                 : 0
Release                : 9070200
------------------------------------------------------

Database Level Details
----------------------
Database Member Activation Time : 2009-12-08-14.41.55.089416
Coordinator Member              : 0

Connection Level Details
------------------------
Application ID                  : *LOCAL.gstager.091208194155
Application Handle              : 21
Application Name                : db2bp
Session Authorization ID        :
System Authorization ID         :
Connection Timestamp            : 2009-12-08-14.41.55.089416
Client Process ID               : 13043
Client Platform                 : LINUXX8664
Client Product ID               : SQL09072
Client Protocol                 : LOCAL
Client Hostname                 : HOSTX
Client Port Number              : 0

UOW Level Details
-------------------------
Start Time                      : 2009-12-08-14.44.39.160651
Stop Time                       : 2009-12-08-14.44.39.162707
Completion Status               : COMMIT
UOW ID                          : 12
Workoad Occurrence ID           : 1
Workload Name                   : SYSDEFAULTUSERWORKLOAD
Workoad ID                      : 1
Service Superclass Name         : SYSDEFAULTUSERCLASS
Service Subclass Name           : SYSDEFAULTSUBCLASS
Service Class ID                : 13
Client Userid                   :
Client Workstation Name         :
Client Application Name         :
Client Accounting String        :
Local Transaction ID            : 000000000000013B
Global Transaction ID           : 00000000000000000000000000000000000000000000
Log Space Used                  : 124

UOW Metrics
-------------------------
TOTAL_CPU_TIME            : 1591
TOTAL_WAIT_TIME           : 8363
ACT_ABORTED_TOTAL         : 0
ACT_COMPLETED_TOTAL       : 1
ACT_REJECTED_TOTAL:       : 0
AGENT_WAIT_TIME           : 87
AGENT_WAITS_TOTAL         : 1
APP_RQSTS_COMPLETED_TOTAL : 1
    .
    .
    .

Package List
-------------------------
Package List Size       : 2
Package List Exceeded   : no

PACKAGE_ID           NESTING_LEVEL ROUTINE_ID  INVOCATION_ID        PACKAGE_ELAPSED_TIME
-------------------- ------------- ----------- -------------------- --------------------
240                  0             0           0                    0
330                  1             66539       1                    1
```

注: 『UOW Metrics』 セクションのいくつかのメトリックは除外されています。

図 5. パッケージ・リスト情報を含む、作業単位イベント・モニターからの出力例

特定の作業単位に関するパッケージ・リストに示されるパッケージの数は、基本的な作業単位イベント・モニター・データに含まれる **package_list_count** モニター・エレメント (上記のレポートでは「Package List Size」) に反映されます。作業単位で使用されるパッケージの数が **mon_pkglist_sz** 構成パラメーターの指定値を

超えた場合、パッケージ・リストには追加のパッケージが含まれません。ただし、パッケージ・リストに収まる数よりも多いパッケージが存在したかどうかが、**package_list_exceeded** モニター・エレメントに示されます。このモニター・エレメントは、作業単位イベント・モニターの基本情報と共に戻されます (265 ページの図 5 では「Package List Exceeded」)。このモニター・エレメントの値が YES である場合、パッケージ・リストにより多くのパッケージを含めるために **mon_pkglist_sz** の値を増やすことができます。

例

以下に示すそれぞれの例は、パッケージ・リストに関して戻された情報を **db2evmonfmt** ツールで表示したものです。

例 1: 単一のパッケージ内の 1 つ以上のセクションを実行するアプリケーション

この例では、パッケージ ID 300 である 1 つのパッケージがこの作業単位に関して実行されました。

```
PACKAGE_ID  NESTING_LEVEL  ROUTINE_ID INVOCATION_ID ELAPSED_TIME
----------  -------------  ---------- ------------- ------------
300         0              0          0             100
```

この場合、パッケージ・リストに 1 つの項目があり、これはパッケージ内の 1 つ以上のセクション実行を反映しています。同じパッケージで実行されるすべてのセクションは、同じパッケージ呼び出しに含まれると見なされます。

例 2: パッケージ内でストアード・プロシージャーを呼び出すアプリケーション

この例では、パッケージ ID 300 であるパッケージが、806 という ID を持つストアード・プロシージャーを呼び出します。ストアード・プロシージャー内の 3 つのセクションが実行されます。

```
PACKAGE_ID  NESTING_LEVEL  ROUTINE_ID INOVATION_ID ELAPSED_TIME
----------  -------------  ---------- ------------ ------------
300         0              0          0            21
300         1              806        1            100
```

この出力は、2 つの項目をリストで示しています。 1 つはストアード・プロシージャー呼び出しに関する項目、もう 1 つはストアード・プロシージャー内の 3 つのセクション実行に関する項目です。リストの 2 番目の項目の NESTING_LEVEL は、別のパッケージからストアード・プロシージャーが呼び出されたことを表しています。

例 3: 異なる 2 つのパッケージのセクションを実行するアプリケーション

この例では、アプリケーションが 1 つのパッケージのセクションを実行した後、別のパッケージに移り、さらに最初のパッケージに戻ります。ストアード・プロシージャーは呼び出されません。以下の疑似コードは、この作業単位を表記したものです。

```
Application
    EXEC PACKAGEA
    EXEC PACKAGEB
    EXEC PACKAGEA
```

また、**PACKAGEA** の呼び出しには 100 ms、**PACKAGEB** の呼び出しには 25 ms、および **PACKAGEC** の呼び出しには 460 ms を要すると想定します。以下の出力は、パッケージ・リストがどのように表示されるかを示すものです。

```
PACKAGE_ID  NESTING_LEVEL  ROUTINE_ID INVOCATION_ID ELAPSED_TIME
----------  -------------  ---------- ------------- ------------
300         0              0          0             560
301         0              0          0             25
```

この場合、リストには 2 つの項目があります。パッケージ A (PACKAGE_ID 300) のセクションは、合計 560 ミリ秒にわたって実行されました。パッケージ B は 25 ミリ秒にわたって実行されました。パッケージ A は、それぞれの呼び出しの INVOCATION_ID および NESTING_LEVEL が同じであるため、1 行で表されています。どちらのパッケージでもストアード・プロシージャーが 1 つも呼び出されなかったため、INVOCATION_ID と NESTING_LEVEL は 0 のままです。

例 4: 複数のパッケージでセクションおよびストアード・プロシージャーを実行するアプリケーション

この例では、ID がそれぞれ 100、101、102 である 3 つのパッケージが存在します。アプリケーションはパッケージ 100 の中にあります。 2 つのストアード・プロシージャーが存在し、それぞれの ID は 201 および 202 です。最初のストアード・プロシージャー (SP1) はパッケージ 101 に、2 番目 (SP2) はパッケージ 102 にそれぞれ含まれています。以下の疑似コードは、この作業単位を表記したものです。

```
Application
   CALL SP1 a
      INSERT INTO T1 VALUES(7)  b
      CALL SP2 c
         INSERT INTO T2 VALUES(8)
      CALL SP2 d
         INSERT INTO T2 VALUES(8)
```

この作業単位のパッケージ・リストは、次のようになります。

```
PACKAGE_ID  NESTING_LEVEL  ROUTINE_ID INVOCATION_ID ELAPSED_TIME
----------  -------------  ---------- ------------- ------------
100         0              0          0             21
101         1 1            201        1             40
102         2 2            202        1 3           35
102         2              202        2 3           35
```

上記の出力には、次の 4 つの項目があります。

- 最初の項目は、最初のパッケージ内での SP1 呼び出しの実行に該当し、作業単位を表す疑似コードでは a という行です。
- 2 番目の項目は、パッケージ 101 内の 201 という ID を持つストアード・プロシージャー内のセクション実行に該当します。これらのセクションには行 b、c、および d が含まれます。ネスト・レベルは 1 に増えて、それが 1 で示されています。
- 3 番目の項目は、SP1 から呼び出される、SP2 内の最初の INSERT INTO T2 ステートメントの実行を表しています。ネスト・レベルが再び増加します (2)。
- リストの 4 番目の項目は、SP2 内の 2 番目の INSERT INTO T2 ステートメントの実行を表しています。その前の SP2 呼び出しと同様に、この

ストアード・プロシージャーは SP1 から呼び出されるため、ネスト・レベルは引き続き同じです。ただし、この 2 つのステートメントは別々のストアード・プロシージャー呼び出しの中で発生するため、両者はそれぞれ別個の呼び出し ID を持っています (**3**)。したがって、パッケージ・リストの中に 2 つの別個の項目があります。

実行可能リスト情報:

作業単位情報を収集する場合、オプションで、各作業単位の一部として実行されるステートメントの実行可能 ID のリストも収集することができます。

実行可能 ID に関する情報は、未フォーマット・イベント (UE) 表または通常の表に書き込まれます。この情報をキャプチャーするには、次の 2 つの方法があります。

- CREATE WORKLOAD または ALTER WORKLOAD ステートメントの COLLECT UNIT OF WORK DATA 節に EXECUTABLE LIST オプションを使用して、特定のワークロードについての情報を収集します。このステートメントで指定したワークロードの下で実行された作業単位についての情報が、実行可能 ID を含め、アクティブな作業単位 (UOW) イベント・モニターに送られます。
- 構成パラメーターを使用して、データ・サーバーで実行されたすべての作業単位についての情報を、実行可能情報を含め、アクティブな作業単位イベント・モニターに送ります。実行可能 ID 情報を収集するには、**mon_uow_data** 構成パラメーターに BASE を設定し、**mon_uow_execlist** 構成パラメーターに ON を設定します。

実行可能リストのために、次のデータが収集されます。

作業単位レベル

executable_list_size
ある特定の作業単位の実行可能 ID リスト内にある項目の数

executable_list_truncated
リストが切り捨てられたかどうかを示す YES または NO の値。処理中に実行可能リスト全体を格納するために十分なメモリーがない場合、リストは切り捨てられる可能性があります。

実行可能 ID リスト

executable_id (1020 ページの『executable_id 実行可能 ID : モニター・エレメント』)
実行された SQL ステートメント・セクションを一意的に識別する、データ・サーバーで生成された不透明なバイナリー・トークン。

num_executions (1211 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』)
SQL ステートメントが実行された回数。

rows_read (1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』)
表から読み取られた行の数。

total_cpu_time（1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』）
DB2 製品内で使用された CPU 時間の合計。ユーザーおよびシステム両方の CPU 時間の合計を表します。この値はマイクロ秒単位で示されます。

total_act_time（1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』）
アクティビティーの実行にかかった時間の合計。この値はミリ秒単位で示されます。

total_act_wait_time（1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』）
アクティビティーの処理中に、DB2 データベース・サーバー内での待機にかかった合計時間。値はミリ秒単位で示されます。

lock_wait_time（1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』）
ロック待機に費やされた合計経過時間。値はミリ秒単位で示されます。

lock_waits（1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』）
アプリケーションまたは接続がロックを待機した合計回数。

total_sorts（1680 ページの『total_sorts - ソート合計 : モニター・エレメント』） 実行されたソートの合計数。

post_threshold_sorts（1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』）
ソート・ヒープしきい値に達した後でヒープを要求したソートの数。

post_shrthreshold_sorts（1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』）
ソート・メモリー・スロットル・アルゴリズムによってスロットルして戻されたソートの合計数。スロットルされたソートとは、ソート・メモリー・マネージャーから要求されたメモリーよりも少ないメモリーが付与されたソートです。

sort_overflows（1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』）
ソート・ヒープを使い果たし、一時記憶用のディスク・スペースが必要になった可能性のあるソートの合計数。

UE 表への実行リストの書き込み

UOW イベント・モニターで基本的なデータと実行可能 ID リスト・データを収集する場合、 UE 表には少なくとも 2 つの別々のレコードが書き込まれます。最初のレコードには、基本的な UOW データを含む UOW イベントに関する情報が入っています。2 つ目のレコードには、実行可能 ID リストのデータを含む UOW_EXEC_LIST イベントです。この 2 つ目のレコードは、複数のレコードで構成される場合があります。これは、単一の UOW にユニークな実行可能 ID が多数存在する場合があるためです。各イベントが、使用可能なインライン LOB スペー

スに収まるように、これらのレコードは別々の行として UE 表に書き込まれます。UE 表のフォーマット用インターフェースを使用して、これらのイベントの情報をマージすることができます。実行可能 ID リストを収集しない場合、関連付けられたレコードが作成されることはありません。表に行は含まれません。

通常の表への実行リストの書き込み

イベント・モニターの出力に通常の表を使用する場合、実行可能リスト情報は、124 ページの『uow_executable_list 論理データ・グループ』 の一部としてキャプチャーされます。各作業単位が完了すると、1 つ以上の行が UOW_EXECUTABLE_LIST_*evmon_name* 表に追加されます。この表には、論理データ・グループ内のモニター・エレメントごとに 1 つの列があります。この表に追加される行数は、作業単位の一部として実行された固有な実行可能 ID の数に依存します。

実行可能リストの出力

作業単位イベント・モニターが UE 表に書き込む場合、イベント・モニターは、実行情報を収集すると 2 つのレコードを UE 表に書き込みます。 UE 表のデータを表示するための各インターフェースは、 2 つの UE 表レコードに含まれている情報を表示するメカニズムを備えています。 **db2evmonfmt** ツールは各レコードの情報を 1 つのレポートに結合します。EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用すると、結合可能なリレーショナル表が生成されます。実行可能リスト情報は、表 UOW_EXECUTABLE_LIST に入っています。EVMON_FORMAT_UE_TO_XML 表関数は、両方のレコードの情報を含む 1 つの XML 文書を生成します。詳しくは、222 ページの『作業単位イベント・モニターによりキャプチャーされたイベント・データへのアクセス』を参照してください。

イベント・モニターがリレーショナル表に直接書き込む場合、実行可能リスト情報は、表 UOW_EXECUTABLE_LIST_*evmon* に書き込まれます。

パーティション・データベース環境の場合、実行可能 ID リストは、コーディネーター・エージェント・メンバーやデータ・メンバーを含め、メンバーごとに生成されます。 DB2 pureScale 環境の場合、リストはコーディネーター・メンバーから生成されます。これは、非パーティション構成の場合と似ています。

例

以下のサンプルの情報は、 1 つの UOW 内で 5 つの異なる SQL ステートメント・セクションを実行するアプリケーションについて収集されたものです。この出力は、サンプルの列を使用して論理的な表示内容を示しています。実際の出力は、実行するツールまたは照会によって異なります。

```
EXECUTABLE_ID                           NUM_EXECUTIONS ROWS_READ TOTAL_CPU_TIME
--------------------------------------  -------------- --------- --------------
x'01007A0000002002008112617155495179l'              1     23456          76888
x'0100790000002002008112617153355112O'             55       345            768
x'01007C0000002002008112617172072899l'            234        67            232
x'01007B0000002002008112617165727291l'           3456       347           1223
x'01007D0000002002008112617240998771l'          22242      2244         432444
```

この例では、実行可能 ID リスト内に、実行された 5 つの異なるセクションに対応する 5 つのエントリーがあります。5 つのセクションの実行回数は、

NUM_EXECUTIONS 列に示されているようにそれぞれ異なりますが、ユニークなセクションごとにエントリーが 1 つのみあります。最初の行は、 1 回の実行だけで過度な CPU 時間が消費されているため、問題のあるアクティビティー・ステートメントを示している可能性があります。

作業単位イベント・データの収集とレポートの生成

作業単位イベント・モニターを使用することにより、チャージバックの目的で使用できるトランザクションに関するデータを収集できます。収集されたトランザクション・イベント・データは、未フォーマット・イベント表に読めない形式で入っています。このデータを使用して、判読可能なテキスト・レポートを作成することができます。

始める前に

作業単位イベント・モニター・データを収集するには、SYSADM または SYSCTRL 権限がなければなりません。

このタスクについて

このタスクでは、特定のワークロードに関する作業単位イベント・データを収集するための手順を説明します。

パッケージ・リストおよび実行リストの情報も、**mon_uow_pkglist** および **mon_uow_execlist** 構成パラメーターの両方を ON に設定した場合は収集されます。また、ALTER WORKLOAD ステートメントを以下のように変更することによって、 **mon_uow_pkglist** および **mon_uow_execlist** 構成パラメーターの設定にかかわらず、ワークロードに関するパッケージ・リストおよび実行リストの情報を収集することもできます。

- パッケージ・リストの情報の場合は、BASE オプションを BASE INCLUDE PACKAGE LIST オプションに置き換えます。
- 実行リスト情報の場合は、BASE オプションを BASE INCLUDE EXECUTABLE LIST オプションに置き換えます。
- パッケージ・リストと実行リスト情報の場合は、 BASE オプションを BASE INCLUDE PACKAGE LIST, EXECUTABLE LIST オプションに置き換えます。

作業単位イベント・モニターは、アプリケーション・トランザクションと対応 CPU 使用量を示す情報を収集します。作業単位イベント・モニターが収集するトランザクション・イベントの情報の例には、以下のようなものがあります。

- 合計 CPU 使用時間 (TOTAL_CPU_TIME モニター・エレメント)
- アプリケーション・ハンドル (APPLICATION_HANDLE モニター・エレメント)

制約事項

SYSADM または SYSCTRL 権限がない場合、入力データ値は表示できません。

手順

作業単位イベントに関する詳細情報を収集するには、以下のようにします。

1. 次の例に示すように、CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを発行して、UOWEVMON という作業単位イベント・モニターを作成します。

   ```
   CREATE EVENT MONITOR UOWEVMON FOR UNIT OF WORK
      WRITE TO UNFORMATTED EVENT TABLE
   ```

2. 以下のステートメントを発行して、UOWEVMON 作業単位イベント・モニターを活動化します。

   ```
   SET EVENT MONITOR UOWEVMON STATE 1
   ```

3. ステートメント履歴を指定した ALTER WORKLOAD ステートメントを発行して、ワークロード・レベルの作業単位イベント・データ収集を使用可能にします。 例えば、FINANCE および PAYROLL アプリケーションの作業単位データを収集するには、以下のステートメントを発行します。

   ```
   ALTER WORKLOAD finance COLLECT UNIT OF WORK DATA BASE
   ALTER WORKLOAD payroll COLLECT UNIT OF WORK DATA BASE
   ```

4. 作業単位トランザクション・イベントを収集するために、ワークロードを再実行します。
5. データベースに接続します。
6. 以下の例に示すように、XML パーサー・ツール **db2evmonfmt** を使用して、未フォーマット・イベント表に収集されたイベント・データに基づいたフラット・テキスト・レポートを生成します。

   ```
   java db2evmonfmt -d db_name -ue table_name -ftext -u user_id -p password
   ```

7. レポートを分析し、適切な課金を請求できるようにアプリケーションが使用している CPU 時間を判別します。
8. FINANCE と PAYROLL の両方のアプリケーションの作業単位データ収集をオフにする場合は、以下のステートメントを発行します。

   ```
   ALTER WORKLOAD finance COLLECT UNIT OF WORK DATA NONE
   ALTER WORKLOAD payroll COLLECT UNIT OF WORK DATA NONE
   ```

例

以下は、作業単位イベント・モニターが未フォーマット・イベント表に収集したデータを、 **db2evmonfmt** ツールを使用して変換することによって得られるレポートの例です。

```
------------------------------------------------------
Event ID               : 1
Event Type             : UOW
Event Timestamp        : 2008-10-31-13.29.04.130849
Member of detection    : 0
------------------------------------------------------

Database Level Details
----------------------
Member Activation Time     : 2008-10-31T13:28:48.538973
Coordinator Member         : 0

Connection Level Details
------------------------
Application ID             : *LOCAL.gstager.081031172848
Application Handle         : 20
Application Name           : db2bp
Session Authorization ID   : GSTAGER
System Authorization ID    : GSTAGER
Connection Timestamp       : 2008-10-31T13:28:48.538973
```

```
Client Process ID          : 28167
Client Platform            : 30
Client Product ID          : SQL09070
Client Hostname            : gilera
Client Port Number         : 30143

UOW Level Details
-----------------------
Start Time                 : 2008-10-31T13:28:51.560138
Stop Time                  : 2008-10-31T13:29:04.130849
Completion Status          : COMMIT
UOW ID                     : 5
Workoad Occurrence ID      : 1
Workload Name              : SYSDEFAULTUSERWORKLOAD
Workoad ID                 : 1
Client userid              :
Client Workstation Name    :
Client Application Name    :
Client Accounting String   :
Local Transaction ID       : 00000000000000EB
Global Transaction ID      : 00000000000000000000000000000000000000000000
Log Space Used             : 0

UOW Metrics
-----------------------
TOTAL_CPU_TIME                : 7459
TOTAL_WAIT_TIME               : 0
ACT_ABORTED_TOTAL             : 0
...
```

作業単位イベント・モニターを使用して、異なるアプリケーションまたはワークロードが使用した CPU 時間を計算する:

このトピックでは、日常のデータベース操作において作業単位イベント・モニターを使用する 1 つの方法が示されています。

一部のビジネス環境では、アプリケーションが使用する処理時間に応じて部門ごとに請求される場合があります。作業単位イベントを使用すると、異なるそれぞれのアプリケーション、ワークロード、またはサービス・クラスが使用する CPU 時間を記録できます。その後、この情報を、システム・リソースに関する請求を実行する会計アプリケーションで使用できます。

始める前に

CREATE EVENT MONITOR ステートメントでは、イベント・モニターによって生成される未フォーマット・イベント (UE) 表を格納するために最低 8 K のページ・サイズの表スペースが必要となります。表スペースを CREATE EVENT MONITOR ステートメントで明示的に指定しない限りは、データベースのデフォルトの表スペースが使用されます。

このタスクについて

このタスクでは、『チャージバック』アカウンティングの基本的なシナリオについて説明します。以下に記す例では、システムで実行されるすべての作業がトラッキングされます。収集されたデータからレポートが作成され、さまざまなアプリケーションによって使用された CPU 時間が表示されます。

組織の編成方法によっては、ワークロードに基づくシステム時間のトラッキングが適切な場合もあります。または、特定のワークロード、または種々のユーザーごとに、異なるサービス・スーパークラスで使用された CPU 時間を調べることもできます。データがリレーショナル表に書き込まれる場合、このタスクの例が示しているように、SQL を使用するとほとんど無限に近い方法でデータを照会し、提示することができます。

注: 作業単位内のアクティビティーは、さまざまなサービス・サブクラスで実行される可能性があります。したがって、サービス・サブクラスごとに作業単位情報を集約するのは適切ではありません。サービス・クラスごとに CPU 時間を集約する場合には、代わりにアクティビティー・イベント・モニターを使用してください。

手順

1. 作業単位イベント・モニターを作成し、終了した作業単位に関する情報をキャプチャーします。 例えば、TRACKWORK というイベント・モニターを作成するには、以下の SQL を使用できます。

   ```
   CREATE EVENT MONITOR TRACKWORK FOR UNIT OF WORK WRITE TO UNFORMATTED EVENT TABLE
   ```

   このステートメントにより、未フォーマット・イベント (UE) 表に書き込む作業単位イベント・モニターが作成されます。UE 表の名前は、イベント・モニター TRACKWORK 自体と同じで、この表はデフォルトの表スペースに格納されます。
2. 以下のコマンドを実行して、データベース上で実行されたすべての作業単位に関するイベント情報を収集することを、データベース・マネージャーに連絡します。

   ```
   UPDATE DATABASE CONFIGURATION FOR dbname USING MON_UOW_DATA BASE
   ```

   このコマンドにより、データ・サーバーで実行されたすべての作業単位に関する情報が、作業単位の完了時に、活動状態にある作業単位イベント・モニターに送信されます。収集される作業単位データの範囲を制御する方法について詳しくは、220 ページの『データ収集の構成』を参照してください。
3. 次に、イベント・モニターを活動状態にします。

   ```
   SET EVENT MONITOR TRACKWORK STATE 1
   ```

   注: デフォルトでは、このイベント・モニターはデータベースの活動化時に自動的に開始されます。AUTOSTART オプションがデフォルトで適用されるためです。ただし、このイベント・モニターは既にアクティブなデータベースで作成されるので、**SET EVENT MONITOR** コマンドを使用して手動で開始する必要があります。この時点以降、作業単位イベント・モニターは、各作業単位が実行されて完了する間に情報を収集します。それぞれの作業単位が完了すると、イベント・モニターは UE 表 TRACKWORK にイベントに関するレコードを追加します。
4. レポートを作成するためにデータを収集する準備が整ったなら、TRACKWORK UE 表からレコードを抽出する必要があります。

   この情報は XML 形式またはリレーショナル形式で表示できます。その際には、UE 表のデータを変換するために EVMON_FORMAT_UE_TO_XML プロシージャーまたは EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用します。または、**db2evmonfmt** ツールを使用して、イベント・モニターによって戻される

情報のテキスト・レポートを作成することも可能です。この例では、ご自分の必要にとってどの方法が最も適しているかを確認するために、EVMON_FORMAT_UE_TO_TABLES を使用してリレーショナル表を作成する方法が取り上げられています。

```
CALL EVMON_FORMAT_UE_TO_TABLES
   ('UOW', NULL, NULL, NULL, NULL, NULL, NULL, -1, 'SELECT * FROM TRACKWORK')
```

EVMON_FORMAT_UE_TO_TABLES プロシージャーは、イベント・モニターによって生成された UE 表 TRACKWORK を調べます。UE 表の各レコードを選択し、その中から、作業単位イベント・モニターが収集したデータが含まれる行を以下の 2 つのリレーショナル表に作成します。

- UOW_EVENT
- UOW_METRICS

1 番目の表には、キャプチャーされた各イベントと関連する、最も頻繁に使用されるモニター・エレメントとメトリックが含まれます。2 番目の表には、各イベントの詳細メトリックが入っています。

注:

- ステップ 2 (274 ページ) で **MON_UOW_DATA** 構成パラメーターに BASE ではなく PKGLIST を指定した場合、EVMON_FORMAT_UE_TO_TABLES プロシージャーによって UOW_PACKAGE_LIST という 3 番目の表が作成されます。この表には、作業単位に関連したパッケージ・リスト情報が入っています。ただし、この例の場合、基本的なモニター・エレメントだけが収集されるので (ステップ 2 (274 ページ) 参照)、この表には何もデータが入りません (パッケージ・リスト情報の使用法について詳しくは、262 ページの『作業単位イベント・モニターのパッケージ・リスト情報』を参照してください)。
- UOW_METRICS の列に入っている値は、UOW_EVENT 表の METRICS 列に含まれている XML 文書にもあります。列指向のアクセスを行うためにより便利な方法として、UOW_METRICS 表でも提供されています。

5. 前述のステップで生成された表を照会し、種々のアプリケーションによってどれほどの CPU 時間が使用されたかを確認します。以下のステートメントを使用すると、作業単位イベント・モニターが初期化されて以降、さまざまなユーザーがシステムで使用した合計 CPU 時間の明細が戻ります (この例では、クライアント・アプリケーションは sqleseti API によって、または IBM Rational® Application Developer for WebSphere® Software などのご使用になる可能性のあるアプリケーション開発環境によって、データベースに識別されると想定しています)。

```
SELECT SUBSTR(E.CLIENT_USERID,1,10) AS CLIENT_ID,
       SUBSTR(E.CLIENT_APPLNAME,1,80) AS CLIENT_APP,
       SUBSTR(E.CLIENT_WRKSTNNAME,1,10) AS WKSTN,
       SUM(M.TOTAL_CPU_TIME) AS CPU_TIME
       FROM UOW_EVENT E, UOW_METRICS M
WHERE  M.APPLICATION_ID = E.APPLICATION_ID
   AND M.UOW_ID = E.UOW_ID
   AND M.MEMBER = E.MEMBER
GROUP BY E.CLIENT_USERID, E.CLIENT_APPLNAME, E.CLIENT_WRKSTNNAME
ORDER BY CPU_TIME DESC;
```

この照会は、以下の結果を返します。

```
CLIENT_ID  CLIENT_APP                                                                                  WKSTN      CPU_TIME
---------- ------------------------------------------------------------------------------------------ ---------- --------------------
                                                                                                                          987770013
           DB2BATCH                                                                                                       249375000
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003021324173                               91181678
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1004201047173                               66097348
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003191536588                               28824420
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003191536434                               27555568
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003221122075                               16203116
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003221118191                               15759227
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003221531062                               15630121
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003221117466                               15236718
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003221116141                               14607249
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003251550366                               14427883
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003051054311                                1312500
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003051053301                                1296875
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003051139066                                1296875
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003051152281                                1281250
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003041230283                                1046875
                                                                                                      asrisk2           1031250
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003291503479                                 515625
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003251506219                                 484375
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003221444488                                 453125
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003021323249                                 406250
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003251544498                                 296875
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003171431559                                 171875
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003041227488                                 156250
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003221117188                                 109375
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003021333329                                  62500
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003191502148                                  62500
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003191527385                                  62500
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003191528492                                  62500
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003191530518                                  62500
           CLP C:¥DOCUME~1¥ALLUSE~1¥APPLIC~1¥IBM¥DB2¥DB2COPY1¥DB2¥TMP¥CCSCRIPT1003191533265                                  62500
           CLP C:¥Documents and Settings¥All Users¥Application Data¥IBM¥DB2¥DB2COPY1¥DB2DAS                                   62500
```

6. この時点では、作業単位イベント・モニター TRACKWORK は引き続き情報を収集しています。さまざまなアプリケーション、ユーザー、ワークロードが使用する CPU 時間をトラッキングする方法に応じて、以下のいずれかの行動方針を選択できます。

   - 毎日 CPU 使用量を計算する場合、この作業単位イベント・モニターを活動状態のままにできます。以下のように EVMON_FORMAT_UE_TO_TABLES プロシージャーを毎日実行し、前日の消費時間メトリックだけを取り出します。

     ```
     CALL EVMON_FORMAT_UE_TO_TABLES
          ('UOW', NULL, NULL, NULL, NULL, NULL, NULL, -1,
           'SELECT * FROM TRACKWORK
            WHERE (DATE(EVENT_TIMESTAMP)=(CURRENT DATE - 1 DAY))'
          )
     ```

     この方法を使用する場合、EVMON_FORMAT_UE_TO_TABLES プロシージャーによって生成される 3 つのリレーショナル表は、時間の経過とともに CPU の使用履歴によって徐々に大きくなり続けます。ステップ 5 (275 ページ) の照会を実行すると、これらの表が EVMON_FORMAT_UE_TO_TABLES プロシージャーによって最初に作成されて以降の CPU 時間の累積合計が戻ります。この照会に以下のように変更を加えて、前日の結果だけを表示することができます。

     ```
     SELECT SUBSTR(E.CLIENT_USERID,1,10) AS CLIENT_ID,
            SUBSTR(E.CLIENT_APPLNAME,1,80) AS CLIENT_APP,
            SUBSTR(E.CLIENT_WRKSTNNAME,1,10) AS WKSTN,
            SUM(M.TOTAL_CPU_TIME) AS CPU_TIME
            FROM UOW_EVENT E, UOW_METRICS M
     WHERE  M.APPLICATION_ID = E.APPLICATION_ID
        AND M.UOW_ID = E.UOW_ID
     ```

```
   AND M.MEMBER = E.MEMBER
   AND(DATE(E.EVENT_TIMESTAMP)=(CURRENT DATE - 1 DAY))
GROUP BY E.CLIENT_USERID, E.CLIENT_APPLNAME, E.CLIENT_WRKSTNNAME
ORDER BY CPU_TIME DESC;
```

ヒント: 毎日 CPU 使用量をトラッキングするものの、システム上で収集するデータ量を管理する場合には、リレーショナル表を更新してから、UE 表で不要になったデータを削除します。例えば、UE 表 TRACKWORK から前日に収集したデータを削除するには、以下のような DELETE ステートメントを使用します。

```
DELETE FROM TRACKWORK WHERE (DATE(EVENT_TIMESTAMP)=(CURRENT DATE - 1 DAY))
```

イベント・モニターがアクティブの間は、情報の書き込み先のすべての表に対して意図的排他 (IX) 表ロックを保持し、使用中にそうした表がドロップされることがないようにします。大量の行を削除している場合、DELETE ステートメントは大量の行ロックを取得します。この場合、ロック・エスカレーションが生じる可能性があります。行ロックが表ロックに変換される場合があるためです。このように表ロックが要求されると、DELETE ステートメントがハングする恐れがあります。イベント・モニターでは対象の表に既にロックがあるからです。

この状態を回避するため、DELETE ステートメントを発行する前に、以下のようにロック・タイムアウトを設定することを考慮してください。

```
SET CURRENT LOCK TIMEOUT 60
```

ロック・タイムアウト期間を増やしても問題が解決しない場合には、データの小サブセットのみ (例えば、6 時間か 12 時間などの限られた短期間のレコード) を削除してみてください。この方法ではロックがほとんど必要ではありません。そのため、ロック・エスカレーションが生じる可能性が低くなります。

また、ストレージ要件と履歴データを表示する必要性とのバランスを考慮に入れながら、EVMON_FORMAT_UE_TO_TABLES によって生成されるリレーショナル表を整理できます。

- CPU 時間の計算が終了したら、以下のステップを実行して、イベント・モニター情報の収集を停止し、イベント・モニターとその関連表をドロップできます。
  a. **SET EVENT MONITOR TRACKWORK STATE 0** コマンドを使用して、作業単位のこのイベント・モニター情報の収集を無効にします。
  b. DROP EVENT MONITOR ステートメントを使用して、イベント・モニター自体をドロップします。
  c. DROP TABLE ステートメントを使用して、対象のイベント・モニターに関連する表をドロップします。この場合、ドロップする表は以下のように、合計 4 つになります。
     – TRACKWORK。イベント・モニターから情報を収集するのに使用した UE 表
     – UOW_EVENT
     – UOW_METRICS
     – UOW_PACKAGE_LIST

d. オプション: アクティブなイベント・モニターが全く残っていない場合には、以下のコマンドを使用して、作業単位イベント情報が収集されないようにデータベース構成を更新できます。

```
UPDATE DATABASE CONFIGURATION FOR dbname USING MON_UOW_DATA NONE
```

変化形: 特定のワークロードのメトリック収集

これまでの例では、システムで実行されたすべての作業に関する作業単位メトリックをキャプチャーする方法について示されていました。**UPDATE DATABASE CONFIGURATION** コマンドを使用して収集されるデータの範囲を設定すると、必要以上の情報が収集される場合があります。例えば、特定のワークロードによって実行された 1 つの作業のみをトラッキングすることが必要な場合があるかもしれません。この場合、ステップ 2 (274 ページ) で示されていたようにデータベース全体に渡って作業単位情報の収集を有効にするのではなく、CREATE ステートメントまたは ALTER WORKLOAD ステートメントで COLLECT UNIT OF WORK DATA 節を指定できます。この節を使用すると、指定されたワークロードのデータだけが、イベント・モニターによって収集されます。例えば、PAYROLL というワークロードの作業単位データを収集するには、以下のステートメントを使用します。

```
ALTER WORKLOAD PAYROLL COLLNECT UNIT OF WORK DATA BASE
```

ALTER WORKLOAD ステートメントを毎回実行することによって、複数のワークロードのデータを収集できます。

残りのステップは同じです。ただし、ステップ 5 (275 ページ) は例外で、照会を以下のように変更します。

```
SELECT E.WORKLOAD_NAME,
       SUM(M.TOTAL_CPU_TIME) AS CPU_TIME
       FROM UOW_EVENT E, UOW_METRICS M
WHERE  M.APPLICATION_ID = E.APPLICATION_ID
   AND M.UOW_ID = E.UOW_ID
   AND M.MEMBER = E.MEMBER
GROUP  BY E.WORKLOAD_NAME
ORDER  BY CPU_TIME DESC
```

前述のステートメントによって、メトリック収集が有効になっていたそれぞれのワークロードの CPU 時間がレポートされます。

```
WORKLOAD                       CPU_TIME
------------------------------ --------------------
PAYROLL                                  2143292042
MARKETING                                492784916

  2 record(s) selected.
```

パッケージ・キャッシュ・ステートメントの追い出しイベントのモニター

パッケージ・キャッシュ・イベント・モニターは、データベースのパッケージ・キャッシュからフラッシュされたステートメント項目に関連するデータをキャプチャーします。このイベント・モニターによって、パッケージ・キャッシュの内容の履歴が取得できます。これは、SQL 照会のパフォーマンスおよび問題判別のための調査に役立ちます。

概説

パッケージ・キャッシュ・イベント・モニターは、
MON_GET_PKG_CACHE_STMT 表関数と同じ情報を収集します。これには、使用可能なアクティビティー・メトリック、および項目の実行可能セクション情報の全セットが含まれます。

バージョン 10.1 以降は、最も長く実行されているステートメントに関連した入力引数に関する情報を取得することができます。このステートメントは、モニター・エレメント **max_coord_stmt_exec_time** に関連するものです。このステートメントに関連する入力引数は、pkgcache_stmt_args 論理データ・グループの一部として記録されます。

CREATE EVENT MONITOR ステートメントの 2 つの制御メカニズムは、キャプチャーするデータ量を制限するのに役立ちます。この 2 つの制御メカニズムとは、以下の機能です。

1. 次の条件の 1 つ以上に基づく WHERE 節を使用して、項目をフィルタリングします。
   - 項目のメトリックの最終更新が、その項目がパッケージ・キャッシュから削除されるまでの特定の時間の後に発生したかどうか (UPDATED_SINCE_BOUNDARY_TIME)。メトリックが最後に更新された時間が、イベント・モニター用に定義された境界時間よりも新しい場合にのみ、項目が収集されます。イベント・モニターの境界時間は MON_GET_PKG_CACHE_STMT 表関数を使って設定可能です。イベント・モニターの境界時間がまだ設定されていない場合、UPDATED_SINCE_BOUNDARY_TIME 節を指定しても効果はありません。
   - 項目のセクションの実行回数 (NUM_EXECUTIONS)。
   - ステートメントの実行にかかった時間の合計 (STMT_EXEC_TIME)。
2. COLLECT DATA 節には以下のオプションがあります。
   - COLLECT BASE DATA

     MON_GET_PKG_CACHE_STMT 表関数と同じ情報、および使用可能なアクティビティー・メトリックの全セットを収集します。
   - COLLECT DETAILED DATA

     COLLECT BASE DATA 節で収集されるものと同じ情報を収集し、さらに項目の実行可能セクションに関する情報を含みます。

個々の SQL ステートメントの実行について調査する必要がある場合、MON_GET_PKG_CACHE_STMT 表関数を使用して、キャッシュされた項目の動作を他の項目の動作と比較できます (項目がまだパッケージ・キャッシュに存在する場合)。キャッシュされた項目についての実行メトリック、コンパイル環境、および詳細な説明は、診断目的で使用可能です。

項目が既にパッケージ・キャッシュからフラッシュされている場合、パッケージ・キャッシュ・イベント・モニターを使用して、パッケージ・キャッシュにキャッシュされていたフラッシュ済みの項目の履歴をレビューできます。履歴データには、MON_GET_PKG_CACHE_STMT 表関数によって得られる情報と同じものが含まれています。さらに、イベント・モニターでは、ステートメントの実行可能セクショ

ンについての情報が取得できます。動的 SQL および静的 SQL ステートメントの両方について取得できます。

パッケージ・キャッシュ・イベント・モニターの作成

パッケージ・キャッシュ・イベント・モニターを作成し、パッケージ・キャッシュ・イベント・モニター・データを収集するには、 DBADM または SQLADM 権限が必要です。

パッケージ・キャッシュ・イベント・モニターは、通常の表または未フォーマット・イベント表のどちらかに出力を書き込めます。

パッケージ・キャッシュ・イベント・モニターを作成する前に、イベント・モニターの出力を保管する表スペースを決定します。表スペースが指定されない場合、CREATE EVENT MONITOR ステートメントは、デフォルトの表スペースを使用します。ただし、推奨されている手法は、イベント・モニターに関連付けられている出力表を保管するための専用の表スペースを構成する方法です。未フォーマット・イベント表を使用する場合は、イベント・データが UE 表のインライン BLOB 列内に確実に入るように、少なくとも 8K のページ・サイズの表スペースにパッケージ・キャッシュ・イベント・モニターを作成してください。 BLOB 列がインライン化されていない場合は、未フォーマット・イベント表に対するイベントの書き込みと読み出しのパフォーマンスが効率的でない可能性があります。

デフォルトおよびベスト・プラクティスを使用してパッケージ・キャッシュ・イベント・モニターをセットアップするには、以下のステップを実行します。

- CREATE EVENT MONITOR ステートメントを発行してイベント・モニターを作成します。以下の例では、可能な場合はデフォルトを使用し、未フォーマット・イベント表を既存の表スペース MY_EVMON_TABLESPACE に保管することを指定します。

```
CREATE EVENT MONITOR MY_PKGCACHE_EVMON
  FOR PACKAGE CACHE
   WRITE TO UNFORMATTED EVENT TABLE (IN MY_EVMON_TABLESPACE)
```

データ収集の有効化

データ収集を有効にするには、 SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを活動化する必要があります。パッケージ・キャッシュ・イベント・モニターはパッシブ なイベント・モニターではありません。活動化された後、ステートメントがパッケージ・キャッシュからフラッシュされ、パッケージ・キャッシュ・イベント・モニターの作成時に設定されたフィルタリング基準を満たす度に、自動的にデータ収集を開始します。

パッケージ・キャッシュ・イベント・モニターによりキャプチャーされたイベント・データへのアクセス

作業単位イベント・モニターは、通常の表にデータを書き込むことも、未フォーマット・イベント (UE) 表にバイナリー・フォーマットのデータを書き込むこともできます。通常の表のデータには SQL を使用してアクセスできます。

UE 表のデータにアクセスするには、次のいずれかの表関数を使用します。

EVMON_FORMAT_UE_TO_XML
未フォーマット・イベント表から XML 文書にデータを抽出します。

EVMON_FORMAT_UE_TO_TABLES
未フォーマット・イベント表から一連のリレーショナル表にデータを抽出します。

これらの表関数のいずれかを使用する場合、関数のパラメーターの 1 つとして SELECT ステートメントを含めることによって、どのデータを抽出するかを指定できます。 SELECT ステートメントにより提供される選択、順序付け、その他の面についてはすべて制御できます。

スキーマ・ファイル ~/sqllib/misc/DB2EvmonPkgCache.xsd は、パッケージ・キャッシュ・イベント・モニター・レポートの期待される出力を、 XML 文書化するのに使用します。スキーマ・ファイルは、共通のモニター・スキーマ・ファイル (DB2MonCommon.xsd) を参照するため、共通の内容を重複させずにすみます。

XML スタイルシートは ~/sqllib/samples/jdbc/DB2EvmonPkgCache.xsl で提供されます。

これらの表関数を使用して、SELECT ステートメントを使用して抽出するデータを指定します。 SELECT ステートメントにより提供される選択、順序付け、その他の面についてはすべて制御できます。

db2evmonfmt コマンドを使用して、以下のタスクを実行することもできます。

- 実行可能 ID、セクション・タイプ、照会コストの見積もり、ステートメント・パッケージ・キャッシュ ID、またはフラッシュ時刻の各属性に基づいて、関心対象のイベントを選択します。
- テキスト・レポートまたはフォーマット済み XML 文書のどちらの形式で出力を受け取るかを選択します。
- **db2evmonfmt** コマンドに提供されているものを使用する代わりに、独自の XSLT スタイル・シートを作成して、出力形式を制御します。

例えば、以下のコマンドにより、次のようなパッケージ・キャッシュ・レポートが提供されます。

1. データベース SAMPLE で過去 24 時間に発生したパッケージ・キャッシュ・イベントを選択します。これらのイベント・レコードは、SAMPLE_PKGCACHE_EVENTS という未フォーマット・イベント表から取得します。
2. DB2EvmonPkgCache.xsl スタイル・シートを使用してフォーマット済みのテキスト出力を提供します。

```
java db2evmonfmt -d SAMPLE -ue SAMPLE_PKGCACHE_EVENTS -ftext -ss DB2EvmonPkgCache.xsl -hours 24
```

パッケージ・キャッシュ・イベント・モニターによって生成されるデータ

パッケージ・キャッシュ・イベント・モニターは、パッケージ・キャッシュから追い出されたパッケージに関するデータを生成します。パッケージ・キャッシュ・イベント・モニターの出力先を通常の表にするか、未フォーマット・イベント (UE) 表にするか選択できます。

UE 表にデータを書き込む場合、データを表示するにはデータに後処理を実行する必要があります。

選択した出力形式にかかわらず、すべてのパッケージ・キャッシュ・イベント・データは、次の 3 つの論理グループのいずれかから取得されます。

- pkgcache
- pkgcache_metrics
- pkgcache_stmt_args

パッケージ・キャッシュ・イベント・データを通常の表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

注: ロックおよび作業単位のイベント・モニターとは異なり、パッケージ・キャッシュ・イベント・モニターを作成した後はパッケージ・キャッシュ・イベント・データの生成を有効化する必要はありません。データ収集は、イベント・モニターが活動化されるとすぐに開始されます。

パッケージ・キャッシュ・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、パッケージ・キャッシュ・イベント・モニターによって書き込まれる情報。

パッケージ・キャッシュ・イベント・モニターの出力タイプとして WRITE TO TABLE を選択した場合、デフォルトでは、3 つの表が生成されます。各表には、1 つ以上の論理データ・グループのモニター・エレメントが入っています。

表 39. 表書き込みパッケージ・キャッシュ・イベント・モニターによって生成される表:
表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
PKGCACHE_*evmon-name*	pkgcache
PKGCACHE_METRICS_*evmon-name*	pkgcache_metrics
PKGCACHE_STMT_ARGS_*evmon-name*	121 ページの『pkgcache_stmt_args 論理データ・グループ』
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 40. パッケージ・キャッシュ・イベント・モニターに戻される情報: デフォルトの表名: PKGCACHE_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
COMP_ENV_DESC	BLOB	comp_env_desc コンパイル環境
EFFECTIVE_ISOLATION	CHARACTER (2)	effective_isolation - 有効な分離
EVENT_ID	BIGINT	event_id - イベント ID モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
EXECUTABLE_ID	VARCHAR (32)	executable_id 実行可能 ID
INSERT_TIMESTAMP	TIMESTAMP	insert_timestamp - 挿入タイムスタンプ
LAST_METRICS_UPDATE	TIMESTAMP	last_metrics_update - メトリック最終更新タイム・スタンプ
MAX_COORD_STMT_EXEC_TIME	BIGINT	max_coord_stmt_exec_time - コーディネーターの最大ステートメント実行時間
MAX_COORD_STMT_EXEC _TIMESTAMP	TIMESTAMP	max_coord_stmt_exec_timestamp - コーディネートされた最大ステートメント実行のタイム・スタンプ
MEMBER	SMALLINT	member - データベース・メンバー
METRICS	BLOB	
NUM_COORD_EXEC	BIGINT	num_coord_exec - コーディネーター・エージェントによる実行数
NUM_COORD_EXEC_WITH _METRICS	BIGINT	num_coord_exec_with_metrics - メトリック付きの、コーディネーター・エージェントによる実行数
NUM_EXEC_WITH_METRICS	BIGINT	num_exec_with_metrics メトリックが収集された実行数
NUM_EXECUTIONS	BIGINT	num_executions ステートメント実行回数
NUM_ROUTINES	BIGINT	num_routines - ルーチンの数
PACKAGE_NAME	VARCHAR (128)	package_name パッケージ名
PACKAGE_SCHEMA	VARCHAR (128)	package_schema - パッケージ・スキーマ
PACKAGE_VERSION_ID	VARCHAR (64)	package_version_id パッケージ・バージョン
PREP_TIME	BIGINT	prep_time 準備時間
QUERY_COST_ESTIMATE	BIGINT	query_cost_estimate 照会コストの見積もり
QUERY_DATA_TAG_LIST	VARCHAR (32)	query_data_tag_list - 照会データ・タグ・リスト
ROUTINE_ID	BIGINT	routine_id - ルーチン ID
SECTION_ENV	BLOB(0)	section_env セクション環境
SECTION_NUMBER	BIGINT	section_number セクション番号

表 40. パッケージ・キャッシュ・イベント・モニターに戻される情報: デフォルトの表名: PKGCACHE_evmon-name (続き)

列名	データ・タイプ	説明
SECTION_TYPE	CHARACTER (1)	section_type - セクション・タイプ標識
STMT_PKG_CACHE_ID	BIGINT	1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』
STMT_TEXT	CLOB	stmt_text SQL ステートメント・テキスト
STMT_TYPE_ID	VARCHAR (32)	stmt_type_id - ステートメント・タイプ ID
STMTNO	INTEGER	stmtno - ステートメント番号モニター・エレメント
TOTAL_STATS_FABRICATION _TIME	BIGINT	total_stats_fabrication_time - 統計作成の合計時間
TOTAL_STATS_FABRICATIONS	BIGINT	total_stats_fabrications - 統計作成の合計回数
TOTAL_SYNC_RUNSTATS	BIGINT	total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数
TOTAL_SYNC_RUNSTATS_TIME	BIGINT	total_sync_runstats_time - 同期 RUNSTATS の合計時間

表 41. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名 : PKGCACHE_METRICS_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
EVENT_ID	BIGINT	event_id - イベント ID モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー
WLM_QUEUE_TIME_TOTAL	BIGINT	wlm_queue_time_total - ワークロード・マネージャー合計キュー時間
WLM_QUEUE_ASSIGNMENTS _TOTAL	BIGINT	wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て
FCM_TQ_RECV_WAIT_TIME	BIGINT	fcm_tq_recv_wait_time - FCM 表キュー受信待機時間
FCM_MESSAGE_RECV_WAIT_TIME	BIGINT	fcm_message_recv_wait_time - FCM メッセージの受信待機時間
FCM_TQ_SEND_WAIT_TIME	BIGINT	fcm_tq_send_wait_time - FCM 表キュー送信待機時間
FCM_MESSAGE_SEND_WAIT_TIME	BIGINT	fcm_message_send_wait_time - FCM メッセージの送信待機時間
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCK_WAITS	BIGINT	lock_waits ロック待機数
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求
LOG_BUFFER_WAIT_TIME	BIGINT	log_buffer_wait_time - ログ・バッファー待機時間
NUM_LOG_BUFFER_FULL	BIGINT	num_log_buffer_full フル・ログ・バッファーの回数
LOG_DISK_WAIT_TIME	BIGINT	log_disk_wait_time - ログ・ディスク待機時間

表 41. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名 : PKGCACHE_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
LOG_DISK_WAITS_TOTAL	BIGINT	log_disk_waits_total - ログ・ディスク待機の合計
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計
AUDIT_FILE_WRITE_WAIT_TIME	BIGINT	audit_file_write_wait_time - 監査ファイル書き込み待機時間
AUDIT_FILE_WRITES_TOTAL	BIGINT	audit_file_writes_total - 書き込まれた監査ファイルの合計
AUDIT_SUBSYSTEM_WAIT_TIME	BIGINT	audit_subsystem_wait_time - 監査サブシステム待機時間
AUDIT_SUBSYSTEM_WAITS_TOTAL	BIGINT	audit_subsystem_waits_total - 監査サブシステム待機の合計
DIAGLOG_WRITE_WAIT_TIME	BIGINT	diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間
DIAGLOG_WRITES_TOTAL	BIGINT	diaglog_writes_total - 診断ログ・ファイル書き込みの合計
FCM_SEND_WAIT_TIME	BIGINT	fcm_send_wait_time - FCM 送信待機時間
FCM_RECV_WAIT_TIME	BIGINT	fcm_recv_wait_time - FCM 受信待機時間
TOTAL_ACT_WAIT_TIME	BIGINT	total_act_wait_time - 合計アクティビティー待機時間
TOTAL_SECTION_SORT_PROC_TIME	BIGINT	total_section_sort_proc_time - セクションのソート処理時間の合計
TOTAL_SECTION_SORTS	BIGINT	total_section_sorts - セクションのソートの合計
TOTAL_SECTION_SORT_TIME	BIGINT	total_section_sort_time - セクションのソート時間の合計
TOTAL_ACT_TIME	BIGINT	total_act_time - 合計アクティビティー時間
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_MODIFIED	BIGINT	rows_modified 変更行数
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み

表 41. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名 : PKGCACHE_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
ROWS_RETURNED	BIGINT	rows_returned 戻り行数
DEADLOCKS	BIGINT	deadlocks デッドロック検出数
LOCK_TIMEOUTS	BIGINT	lock_timeouts ロック・タイムアウト数
LOCK_ESCALS	BIGINT	lock_escals ロック・エスカレーション数
FCM_SENDS_TOTAL	BIGINT	fcm_sends_total - FCM 送信の合計
FCM_RECVS_TOTAL	BIGINT	fcm_recvs_total - FCM 受信の合計
FCM_SEND_VOLUME	BIGINT	fcm_send_volume - FCM 送信ボリューム
FCM_RECV_VOLUME	BIGINT	fcm_recv_volume - FCM 受信ボリューム
FCM_MESSAGE_SENDS_TOTAL	BIGINT	fcm_message_sends_total - FCM メッセージ送信の合計
FCM_MESSAGE_RECVS_TOTAL	BIGINT	fcm_message_recvs_total - FCM メッセージ受信の合計
FCM_MESSAGE_SEND_VOLUME	BIGINT	fcm_message_send_volume - FCM メッセージ送信ボリューム
FCM_MESSAGE_RECV_VOLUME	BIGINT	fcm_message_recv_volume - FCM メッセージ受信ボリューム
FCM_TQ_SENDS_TOTAL	BIGINT	fcm_tq_sends_total - FCM 表キュー送信の合計
FCM_TQ_RECVS_TOTAL	BIGINT	fcm_tq_recvs_total - FCM 表キュー受信の合計
FCM_TQ_SEND_VOLUME	BIGINT	fcm_tq_send_volume - FCM 表キュー送信ボリューム
FCM_TQ_RECV_VOLUME	BIGINT	fcm_tq_recv_volume - FCM 表キュー受信ボリューム
TQ_TOT_SEND_SPILLS	BIGINT	tq_tot_send_spills オーバーフローした表キュー・バッファーの合計数
POST_THRESHOLD_SORTS	BIGINT	post_threshold_sorts ポストしきい値ソート
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts ポスト共有しきい値ソート
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
AUDIT_EVENTS_TOTAL	BIGINT	audit_events_total - 監査イベントの合計
TOTAL_SORTS	BIGINT	total_sorts ソート合計
STMT_EXEC_TIME	BIGINT	stmt_exec_time - ステートメント実行時間
COORD_STMT_EXEC_TIME	BIGINT	coord_stmt_exec_time - コーディネーター・エージェントのステートメントの実行時間
TOTAL_ROUTINE_NON_SECT_PROC_TIME	BIGINT	total_routine_non_sect_proc_time - 非セクション処理時間
TOTAL_ROUTINE_NON_SECT_TIME	BIGINT	total_routine_non_sect_time - 非セクション・ルーチンの実行時間
TOTAL_SECTION_PROC_TIME	BIGINT	total_section_proc_time - セクション処理時間の合計
TOTAL_APP_SECTION_EXECUTIONS	BIGINT	total_app_section_executions - アプリケーションのセクション実行数の合計
TOTAL_SECTION_TIME	BIGINT	total_section_time - 合計セクション時間
TOTAL_ROUTINE_USER_CODE_PROC_TIME	BIGINT	total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計
TOTAL_ROUTINE_USER_CODE_TIME	BIGINT	total_routine_user_code_time - ルーチンのユーザー・コード時間の合計
TOTAL_ROUTINE_TIME	BIGINT	total_routine_time - 合計ルーチン時間

表 41. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名 : *PKGCACHE_METRICS_evmon-name* (続き)

列名	データ・タイプ	説明
THRESH_VIOLATIONS	BIGINT	thresh_violations - しきい値違反の回数
NUM_LW_THRESH_EXCEEDED	BIGINT	num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数
TOTAL_ROUTINE_INVOCATIONS	BIGINT	total_routine_invocations - ルーチンの合計呼び出し数
LOCK_WAIT_TIME_GLOBAL	BIGINT	lock_wait_time_global - グローバル・ロック待機時間
LOCK_WAITS_GLOBAL	BIGINT	lock_waits_global - グローバル・ロック待機
RECLAIM_WAIT_TIME	BIGINT	reclaim_wait_time - 再利用待機時間
SPACEMAPPAGE_RECLAIM_WAIT _TIME	BIGINT	spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間
LOCK_TIMEOUTS_GLOBAL	BIGINT	lock_timeouts_global - グローバル・ロック・タイムアウト
LOCK_ESCALS_MAXLOCKS	BIGINT	lock_escals_maxlocks - maxlocks ロック・エスカレーション数
LOCK_ESCALS_LOCKLIST	BIGINT	lock_escals_locklist - locklist ロック・エスカレーション数
LOCK_ESCALS_GLOBAL	BIGINT	lock_escals_global - グローバル・ロック・エスカレーション数
CF_WAIT_TIME	BIGINT	cf_wait_time - クラスター・キャッシング機能待機時間
CF_WAITS	BIGINT	cf_waits - クラスター・キャッシング機能 DB2 pureScale サーバーの待機回数
POOL_DATA_GBP_L_READS	BIGINT	pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り
POOL_DATA_GBP_P_READS	BIGINT	pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り
POOL_DATA_LBP_PAGES_FOUND	BIGINT	pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ
POOL_DATA_GBP_INVALID_PAGES	BIGINT	pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ
POOL_INDEX_GBP_L_READS	BIGINT	pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り
POOL_INDEX_GBP_P_READS	BIGINT	pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り
POOL_INDEX_LBP_PAGES_FOUND	BIGINT	pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ
POOL_INDEX_GBP_INVALID_PAGES	BIGINT	pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ
POOL_XDA_GBP_L_READS	BIGINT	pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求
POOL_XDA_GBP_P_READS	BIGINT	pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求
POOL_XDA_LBP_PAGES_FOUND	BIGINT	pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ
POOL_XDA_GBP_INVALID_PAGES	BIGINT	pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ
EVMON_WAIT_TIME	BIGINT	evmon_wait_time - イベント・モニターの待機時間
EVMON_WAITS_TOTAL	BIGINT	evmon_waits_total - イベント・モニター合計待機回数
TOTAL_EXTENDED_LATCH_WAIT _TIME	BIGINT	total_extended_latch_wait_time - 拡張ラッチの合計待機時間
TOTAL_EXTENDED_LATCH_WAITS	BIGINT	total_extended_latch_waits - 拡張ラッチの合計待機回数

表 41. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名 : PKGCACHE_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間
POOL_QUEUED_ASYNC_DATA _REQS	BIGINT	pool_queued_async_data_reqs - データ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX _REQS	BIGINT	pool_queued_async_index_reqs - 索引プリフェッチ要求
POOL_QUEUED_ASYNC_XDA_REQS	BIGINT	pool_queued_async_xda_reqs - XDA プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _DATA_REQS	BIGINT	pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _INDEX_REQS	BIGINT	pool_queued_async_temp_index_reqs - TEMPORARY 表スペースの索引プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _XDA_REQS	BIGINT	pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求
POOL_QUEUED_ASYNC_OTHER _REQS	BIGINT	pool_queued_async_other_reqs - 非プリフェッチの要求
POOL_QUEUED_ASYNC_DATA _PAGES	BIGINT	pool_queued_async_data_pages - データ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX _PAGES	BIGINT	pool_queued_async_index_pages - 索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_XDA _PAGES	BIGINT	pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _DATA_PAGES	BIGINT	pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _INDEX_PAGES	BIGINT	pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_XDA _PAGES	BIGINT	pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求
POOL_FAILED_ASYNC_DATA_REQS	BIGINT	pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求
POOL_FAILED_ASYNC_INDEX_REQS	BIGINT	pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求
POOL_FAILED_ASYNC_XDA_REQS	BIGINT	pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_DATA _REQS	BIGINT	pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求
POOL_FAILED_ASYNC_TEMP _INDEX_REQS	BIGINT	pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_XDA _REQS	BIGINT	pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求
POOL_FAILED_ASYNC_OTHER _REQS	BIGINT	pool_failed_async_other_reqs - 失敗した非プリフェッチの要求
TOTAL_PEDS	BIGINT	total_peds - partial early distinct の合計回数
DISABLED_PEDS	BIGINT	1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』
POST_THRESHOLD_PEDS	BIGINT	post_threshold_peds - partial early distinct しきい値
TOTAL_PEAS	BIGINT	total_peas - partial early aggregation の合計回数
POST_THRESHOLD_PEAS	BIGINT	post_threshold_peas - partial early aggregation しきい値
TQ_SORT_HEAP_REQUESTS	BIGINT	tq_sort_heap_requests - 表キュー・ソート・ヒープ要求

表 41. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名 : PKGCACHE_METRICS_evmon-name (続き)

列名	データ・タイプ	説明
TQ_SORT_HEAP_REJECTIONS	BIGINT	tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否
PREFETCH_WAIT_TIME	BIGINT	prefetch_wait_time プリフェッチ待ち時間
PREFETCH_WAITS	BIGINT	prefetch_waits - プリフェッチャーの待機カウント
POOL_DATA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1288 ページの『pool_data_gbp_indep_pages_found _in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
POOL_INDEX_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1326 ページの『pool_index_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
POOL_XDA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1393 ページの『pool_xda_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』

表 42. パッケージ・キャッシュ・イベント・モニターに戻される情報: デフォルトの表名: PKGCACHE_STMT_ARGS_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
EVENT_ID	BIGINT	event_id - イベント ID モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー
STMT_VALUE_DATA	CLOB	stmt_value_data 値データ
STMT_VALUE_INDEX	INTEGER	stmt_value_index 値索引
STMT_VALUE_ISNULL	INTEGER	stmt_value_isnull NULL 値の値
STMT_VALUE_ISREOPT	INTEGER	stmt_value_isreopt ステートメント再最適化に使用される変数
STMT_VALUE_TYPE	CHARACTER(16)	stmt_value_type 値タイプ

以下のデータ・タイプの項目は上記の表に記録されますが、引数の実際の値は STMT_VALUE_DATA エレメントに記録されません。

- BLOB
- CLOB
- REF
- BOOLEAN
- 構造化データ・タイプ
- DATALINK
- LONG VARGRAPHIC
- LONG VARCHAR
- XML タイプ
- DBCLOB
- ARRAY タイプ

- ROW タイプ
- ROWID
- CURSOR 変数

表 43. パッケージ・キャッシュ・イベント・モニターに戻される情報: デフォルトの表名: CONTROL_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号

パッケージ・キャッシュ・イベント・モニターの場合に EVMON_FORMAT_UE_TO_TABLES によってリレーショナル表に書き込まれる情報:

EVMON_FORMAT_UE_TO_TABLES 表関数からパッケージ・キャッシュ・イベント・モニター用に書き込まれる情報。これは、DB2EvmonPkgCache.xsd ファイルにも記載されています。

表 44. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_EVENT

列名	データ・タイプ	説明
XMLID	VARCHAR(256) NOT NULL	1744 ページの『xmlid - XML ID モニター・エレメント』
EVENT_ID	BIGINT NOT NULL	event_id - イベント ID モニター・エレメント
EVENT_TYPE	VARCHAR(128) NOT NULL	event_type - イベント・タイプ・モニター・エレメント
EVENT_TIMESTAMP	TIMESTAMP NOT NULL	event_timestamp - イベント・タイム・スタンプ・モニター・エレメント
MEMBER	SMALLINT NOT NULL	member - データベース・メンバー
SECTION_TYPE	CHAR(1)	section_type - セクション・タイプ標識
INSERT_TIMESTAMP	TIMESTAMP	insert_timestamp - 挿入タイムスタンプ
EXECUTABLE_ID	VARCHAR(32) FOR BIT DATA	executable_id 実行可能 ID
PACKAGE_SCHEMA	VARCHAR(128)	package_schema - パッケージ・スキーマ
PACKAGE_NAME	VARCHAR(128)	package_name パッケージ名
PACKAGE_VERSION_ID	VARCHAR(64)	package_version_id パッケージ・バージョン

表 44. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_EVENT (続き)

列名	データ・タイプ	説明
SECTION_NUMBER	BIGINT	section_number セクション番号
EFFECTIVE_ISOLATION	CHAR(2)	effective_isolation - 有効な分離
NUM_EXECUTIONS	BIGINT	num_executions ステートメント実行回数
NUM_EXEC_WITH_METRICS	BIGINT	num_exec_with_metrics メトリックが収集された実行数
PREP_TIME	BIGINT	prep_time 準備時間
LAST_METRICS_UPDATE	TIMESTAMP	last_metrics_update - メトリック最終更新タイム・スタンプ
NUM_COORD_EXEC	BIGINT	num_coord_exec - コーディネーター・エージェントによる実行回数
NUM_COORD_EXEC_WITH _METRICS	BIGINT	num_coord_exec_with_metrics - メトリックに関するコーディネーター・エージェントの実行回数
STMT_TYPE_ID	VARCHAR(32)	stmt_type_id - ステートメント・タイプ ID
QUERY_COST_ESTIMATE	BIGINT	query_cost_estimate 照会コストの見積もり
STMT_PKG_CACHE_ID	BIGINT	1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』
STMT_TEXT	CLOB(2M)	stmt_text SQL ステートメント・テキスト
COMP_ENV_DESC	BLOB(10K)	comp_env_desc コンパイル環境
METRICS	BLOB(1M)	メトリック関連のモニター・エレメントが含まれる XML 文書。この文書内のメトリックは、このトピックで後ほど取り上げられる PKGCACHE_METRICS 表で説明されるメトリックと同じです。詳しくは、 20 ページの『モニター・データを XML 文書で返すインターフェース』を参照してください。
SECTION_ENV	BLOB(150M)	section_env セクション環境
ROUTINE_ID	BIGINT	routine_id - ルーチン ID
QUERY_DATA_TAG_LIST	VARCHAR(32)	query_data_tag_list - 照会データ・タグ・リスト
TOTAL_STATS_FABRICATION_TIME	BIGINT	total_stats_fabrication_time - 統計作成の合計時間
TOTAL_STATS_FABRICATIONS	BIGINT	total_stats_fabrications - 統計作成の合計回数
TOTAL_SYNC_RUNSTATS_TIME	BIGINT	total_sync_runstats_time - 同期 RUNSTATS の合計時間

表 44. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_EVENT (続き)

列名	データ・タイプ	説明
TOTAL_SYNC_RUNSTATS	BIGINT	total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数
MAX_COORD_STMT_EXEC _TIMESTAMP	TIMESTAMP	max_coord_stmt_exec_timestamp - コーディネートされた最大ステートメント実行の タイム・スタンプ
MAX_COORD_STMT_EXEC_TIME	BIGINT	max_coord_stmt_exec_time - コーディネーターの最大ステートメント実行時間

表 45. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_METRICS。この表のメトリックは、PKGCACHE_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。

列名	データ・タイプ	説明
XMLID	VARCHAR(256) NOT NULL	1744 ページの『xmlid - XML ID モニター・エレメント』
TOTAL_ACT_TIME	BIGINT	total_act_time - 合計アクティビティー時間
TOTAL_ACT_WAIT_TIME	BIGINT	total_act_wait_time - 合計アクティビティー待機時間
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
TOTAL_SECTION_SORT_TIME	BIGINT	total_section_sort_time - セクションのソート時間の合計
TOTAL_SECTION_SORT_PROC _TIME	BIGINT	total_section_sort_proc_time - セクションのソート処理時間の合計
TOTAL_SECTION_SORTS	BIGINT	total_section_sorts - セクションのソートの合計
LOCK_ESCALS	BIGINT	lock_escals ロック・エスカレーション数
LOCK_WAITS	BIGINT	lock_waits ロック待機数
ROWS_MODIFIED	BIGINT	rows_modified 変更行数
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_RETURNED	BIGINT	rows_returned 戻り行数
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求

表 45. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_METRICS。この表のメトリックは、PKGCACHE_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
TOTAL_SORTS	BIGINT	total_sorts ソート合計
POST_THRESHOLD_SORTS	BIGINT	post_threshold_sorts ポストしきい値ソート
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts ポスト共有しきい値ソート
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
WLM_QUEUE_TIME_TOTAL	BIGINT	wlm_queue_time_total - ワークロード・マネージャー合計キュー時間
WLM_QUEUE_ASSIGNMENTS _TOTAL	BIGINT	wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て

表 45. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_METRICS。この表のメトリックは、PKGCACHE_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
DEADLOCKS	BIGINT	deadlocks デッドロック検出数
FCM_RECV_VOLUME	BIGINT	fcm_recv_volume - FCM 受信ボリューム
FCM_RECVS_TOTAL	BIGINT	fcm_recvs_total - FCM 受信の合計
FCM_SEND_VOLUME	BIGINT	fcm_send_volume - FCM 送信ボリューム
FCM_SENDS_TOTAL	BIGINT	fcm_sends_total - FCM 送信の合計
FCM_RECV_WAIT_TIME	BIGINT	fcm_recv_wait_time - FCM 受信待機時間
FCM_SEND_WAIT_TIME	BIGINT	fcm_send_wait_time - FCM 送信待機時間
LOCK_TIMEOUTS	BIGINT	lock_timeouts ロック・タイムアウト数
LOG_BUFFER_WAIT_TIME	BIGINT	log_buffer_wait_time - ログ・バッファー待機時間
NUM_LOG_BUFFER_FULL	BIGINT	num_log_buffer_full フル・ログ・バッファーの回数
LOG_DISK_WAIT_TIME	BIGINT	log_disk_wait_time - ログ・ディスク待機時間
LOG_DISK_WAITS_TOTAL	BIGINT	log_disk_waits_total - ログ・ディスク待機の合計
TOTAL_ROUTINE_TIME	BIGINT	total_routine_time - 合計ルーチン時間
TOTAL_ROUTINE_INVOCATIONS	BIGINT	total_routine_invocations - 合計ルーチン呼び出し数
COORD_STMT_EXEC_TIME	BIGINT	coord_stmt_exec_time - コーディネーター・エージェントによるステートメントの実行時間
STMT_EXEC_TIME	BIGINT	stmt_exec_time - ステートメントの実行時間
TOTAL_SECTION_TIME	BIGINT	total_section_time - 合計セクション時間
TOTAL_SECTION_PROC_TIME	BIGINT	total_section_proc_time - 合計セクション処理時間
TOTAL_ROUTINE_NON_SECT _TIME	BIGINT	total_routine_non_sect_time - 非セクション・ルーチン実行時間
TOTAL_ROUTINE_NON_SECT _PROC_TIME	BIGINT	total_routine_non_sect_proc_time - 非セクション処理時間
FCM_TQ_RECV_WAIT_TIME	BIGINT	fcm_tq_recv_wait_time - FCM 表キュー受信待機時間
FCM_MESSAGE_RECV_WAIT_TIME	BIGINT	fcm_message_recv_wait_time - FCM メッセージの受信待機時間
FCM_TQ_SEND_WAIT_TIME	BIGINT	fcm_tq_send_wait_time - FCM 表キュー送信待機時間
FCM_MESSAGE_SEND_WAIT_TIME	BIGINT	fcm_message_send_wait_time - FCM メッセージの送信待機時間

表 45. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_METRICS。この表のメトリックは、PKGCACHE_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
AUDIT_FILE_WRITE_WAIT_TIME	BIGINT	audit_file_write_wait_time - 監査ファイル書き込み待機時間
AUDIT_FILE_WRITES_TOTAL	BIGINT	audit_file_writes_total - 書き込まれた監査ファイルの合計
AUDIT_SUBSYSTEM_WAIT_TIME	BIGINT	audit_subsystem_wait_time - 監査サブシステム待機時間
AUDIT_SUBSYSTEM_WAITS_TOTAL	BIGINT	audit_subsystem_waits_total - 監査サブシステム待機の合計
DIAGLOG_WRITE_WAIT_TIME	BIGINT	diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間
DIAGLOG_WRITES_TOTAL	BIGINT	diaglog_writes_total - 診断ログ・ファイル書き込みの合計
FCM_MESSAGE_SENDS_TOTAL	BIGINT	fcm_message_sends_total - FCM メッセージ送信の合計
FCM_MESSAGE_RECVS_TOTAL	BIGINT	fcm_message_recvs_total - FCM メッセージ受信の合計
FCM_MESSAGE_SEND_VOLUME	BIGINT	fcm_message_send_volume - FCM メッセージ送信ボリューム
FCM_MESSAGE_RECV_VOLUME	BIGINT	fcm_message_recv_volume - FCM メッセージ受信ボリューム
FCM_TQ_SENDS_TOTAL	BIGINT	fcm_tq_sends_total - FCM 表キュー送信の合計
FCM_TQ_RECVS_TOTAL	BIGINT	fcm_tq_recvs_total - FCM 表キュー受信の合計
FCM_TQ_SEND_VOLUME	BIGINT	fcm_tq_send_volume - FCM 表キュー送信ボリューム
FCM_TQ_RECV_VOLUME	BIGINT	fcm_tq_recv_volume - FCM 表キュー受信ボリューム
TQ_TOT_SEND_SPILLS	BIGINT	tq_tot_send_spills オーバーフローした表キュー・バッファーの合計数
AUDIT_EVENTS_TOTAL	BIGINT	audit_events_total - 監査イベントの合計
TOTAL_APP_SECTION _EXECUTIONS	BIGINT	total_app_section_executions - アプリケーションのセクション実行数の合計
TOTAL_ROUTINE_USER_CODE _PROC_TIME	BIGINT	total_routine_user_code_proc_time - ユーザー・コードのルーチンの合計処理時間
TOTAL_ROUTINE_USER_CODE _TIME	BIGINT	total_routine_user_code_time - ユーザー・コードのルーチンの合計時間
THRESH_VIOLATIONS	BIGINT	thresh_violations - しきい値違反の合計数
NUM_LW_THRESH_EXCEEDED	BIGINT	num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数
LOCK_WAITS_GLOBAL	BIGINT	lock_waits_global - グローバル・ロック待機

表 45. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_METRICS。この表のメトリックは、PKGCACHE_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
LOCK_WAIT_TIME_GLOBAL	BIGINT	lock_wait_time_global - グローバル・ロック待機時間
LOCK_TIMEOUTS_GLOBAL	BIGINT	lock_timeouts_global - グローバル・ロック・タイムアウト
LOCK_ESCALS_MAXLOCKS	BIGINT	lock_escals_maxlocks - maxlocks ロック・エスカレーション数
LOCK_ESCALS_LOCKLIST	BIGINT	lock_escals_locklist - locklist ロック・エスカレーション数
LOCK_ESCALS_GLOBAL	BIGINT	lock_escals_global - グローバル・ロック・エスカレーション数
RECLAIM_WAIT_TIME	BIGINT	reclaim_wait_time - 再利用待機時間
SPACEMAPPAGE_RECLAIM_WAIT _TIME	BIGINT	spacemappage_reclaim_wait _time - スペース・マップ・ページ再利用の待機時間
CF_WAITS	BIGINT	cf_waits - クラスター・キャッシング機能 DB2 pureScale サーバーの待機回数
CF_WAIT_TIME	BIGINT	cf_wait_time - クラスター・キャッシング機能待機時間
POOL_DATA_GBP_L_READS	BIGINT	pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り
POOL_DATA_GBP_P_READS	BIGINT	pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り
POOL_DATA_LBP_PAGES_FOUND	BIGINT	pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ
POOL_DATA_GBP_INVALID_PAGES	BIGINT	pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ
POOL_INDEX_GBP_L_READS	BIGINT	pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り
POOL_INDEX_GBP_P_READS	BIGINT	pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り
POOL_INDEX_LBP_PAGES_FOUND	BIGINT	pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ
POOL_INDEX_GBP_INVALID _PAGES	BIGINT	pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ
POOL_XDA_GBP_L_READS	BIGINT	pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求
POOL_XDA_GBP_P_READS	BIGINT	pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求

表 45. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_METRICS。この表のメトリックは、PKGCACHE_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
POOL_XDA_LBP_PAGES_FOUND	BIGINT	pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ
POOL_XDA_GBP_INVALID_PAGES	BIGINT	pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ
EVMON_WAIT_TIME	BIGINT	evmon_wait_time - イベント・モニターの待機時間
EVMON_WAITS_TOTAL	BIGINT	evmon_waits_total - イベント・モニター合計待機回数
TOTAL_EXTENDED_LATCH_WAIT _TIME	BIGINT	total_extended_latch_wait_time - 拡張ラッチの合計待機時間
TOTAL_EXTENDED_LATCH_WAITS	BIGINT	total_extended_latch_waits - 拡張ラッチの合計待機回数
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間
TOTAL_PEDS	BIGINT	total_peds - partial early distinct の合計回数
DISABLED_PEDS	BIGINT	1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』
POST_THRESHOLD_PEDS	BIGINT	post_threshold_peds - partial early distinct しきい値
TOTAL_PEAS	BIGINT	total_peas - partial early aggregation の合計回数
POST_THRESHOLD_PEAS	BIGINT	post_threshold_peas - partial early aggregation しきい値
TQ_SORT_HEAP_REQUESTS	BIGINT	tq_sort_heap_requests - 表キュー・ソート・ヒープ要求
TQ_SORT_HEAP_REJECTIONS	BIGINT	tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否
POOL_QUEUED_ASYNC_DATA _REQS	BIGINT	pool_queued_async_data_reqs - データ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX _REQS	BIGINT	pool_queued_async_index_reqs - 索引プリフェッチ要求
POOL_QUEUED_ASYNC_XDA _REQS	BIGINT	pool_queued_async_xda_reqs - XDA プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _DATA_REQS	BIGINT	pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP _INDEX_REQS	BIGINT	pool_queued_async_temp_index_reqs - TEMPORARY 表スペースの索引プリフェッチ要求

表 45. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_METRICS。この表のメトリックは、PKGCACHE_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
POOL_QUEUED_ASYNC_TEMP_XDA_REQS	BIGINT	pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求
POOL_QUEUED_ASYNC_OTHER_REQS	BIGINT	pool_queued_async_other_reqs - 非プリフェッチの要求
POOL_QUEUED_ASYNC_DATA_PAGES	BIGINT	pool_queued_async_data_pages - データ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX_PAGES	BIGINT	pool_queued_async_index_pages - 索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_XDA_PAGES	BIGINT	pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_DATA_PAGES	BIGINT	pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES	BIGINT	pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_XDA_PAGES	BIGINT	pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求
POOL_FAILED_ASYNC_DATA_REQS	BIGINT	pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求
POOL_FAILED_ASYNC_INDEX_REQS	BIGINT	pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求
POOL_FAILED_ASYNC_XDA_REQS	BIGINT	pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_DATA_REQS	BIGINT	pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_INDEX_REQS	BIGINT	pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_XDA_REQS	BIGINT	pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求
POOL_FAILED_ASYNC_OTHER_REQS	BIGINT	pool_failed_async_other_reqs - 失敗した非プリフェッチの要求
PREFETCH_WAIT_TIME	BIGINT	prefetch_wait_time プリフェッチ待ち時間
PREFETCH_WAITS	BIGINT	prefetch_waits - プリフェッチャーの待機カウント

表 45. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_METRICS。この表のメトリックは、PKGCACHE_EVENT 表の METRICS モニター・エレメントで戻されるメトリックと同じです。 (続き)

列名	データ・タイプ	説明
POOL_DATA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1288 ページの『pool_data_gbp_indep_pages_found _in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
POOL_INDEX_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1326 ページの『pool_index_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
POOL_XDA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1393 ページの『pool_xda_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』

表 46. パッケージ・キャッシュ・イベント・モニターに戻される情報: 表名: PKGCACHE_STMT_ARGS。

列名	データ・タイプ	説明
XMLID	VARCHAR(256) NOT NULL	1744 ページの『xmlid - XML ID モニター・エレメント』
STMT_VALUE_INDEX	INTEGER NOT NULL	stmt_value_index 値索引
STMT_VALUE_ISREOPT	INTEGER	stmt_value_isreopt ステートメント再最適化に使用される変数
STMT_VALUE_ISNULL	INTEGER	stmt_value_isnull NULL 値の値
STMT_VALUE_TYPE	CHAR(16)	stmt_value_type 値タイプ
STMT_VALUE_DATA	CLOB (32K)	stmt_value_data 値データ

以下のデータ・タイプの項目は上記の表に記録されますが、引数の実際の値は STMT_VALUE_DATA エレメントに記録されません。

- BLOB
- CLOB
- REF
- BOOLEAN
- 構造化データ・タイプ
- DATALINK
- LONG VARGRAPHIC
- LONG VARCHAR
- XML タイプ
- DBCLOB
- ARRAY タイプ
- ROW タイプ

- ROWID
- CURSOR 変数

入力引数は、ステートメント内に最初に出現するものが最初に記録され、その後に出現するものが続いて記録されます。この表に記録できる入力パラメーターの数は、UE イベント・モニターがイベント情報の収集に使用する BLOB 文書のサイズの上限によってのみ、制約されます。実際には、収集される入力引数の数がこの上限に達することはまずないでしょう。

パッケージ・キャッシュ・イベント・モニター用に XML に書き込まれる情報:

EVMON_FORMAT_UE_TO_XML 表関数からパッケージ・キャッシュ・イベント・モニター用に書き込まれる情報。これは、`DB2EvmonPkgCache.xsd` ファイルにも記載されています。

db2_pkgcache_event

パッケージ・キャッシュ・イベントについて詳細に記述するメイン・スキーマ

エレメント・コンテント: (301 ページの『section_type』 、 301 ページの『insert_timestamp』 、 301 ページの『executable_id』 、 301 ページの『package_schema』, 301 ページの『package_name』 、 301 ページの『package_version_id』 、 302 ページの『section_number』 {0 または 1個 (?)}、 302 ページの『effective_isolation』、 302 ページの『num_executions』、 302 ページの『num_exec_with_metrics』、 303 ページの『prep_time』、 303 ページの『last_metrics_update』、 303 ページの『num_coord_exec』、 303 ページの『num_coord_exec_with_metrics』、 304 ページの『stmt_type_id』、 304 ページの『query_cost_estimate』、 304 ページの『stmt_pkg_cache_id』、 304 ページの『stmt_text』、 305 ページの『comp_env_desc』、 305 ページの『section_env』、 305 ページの『activity_metrics』、 305 ページの『routine_id』、 305 ページの『query_data_tag_list』、 305 ページの『total_stats_fabrication_time』、 306 ページの『total_stats_fabrications』、 306 ページの『total_sync_runstats_time』、 306 ページの『total_sync_runstats』、 306 ページの『max_coord_stmt_exec_timestamp』 {0 または 1 個 (?)}、 307 ページの『max_coord_stmt_exec_time_arg』 {0 個以上 (*)}、 307 ページの『max_coord_stmt_exec_time』、 ANY コンテント (検証しない (skip)) {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			必須	
type				必須	
timestamp	xs:dateTime			必須	
メンバー				必須	
release	xs:long			必須	
ANY ネーム・スペースの ANY 属性					

section_type

処理される SQL ステートメントのタイプ。 可能な値は D (動的) または S (静的) です。 詳しくは、モニター・エレメント 1479 ページの『section_type - セクション・タイプ標識 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

insert_timestamp

バリエーションまたはセクションがキャッシュに挿入された時刻。 詳しくは、モニター・エレメント 1102 ページの『insert_timestamp - 挿入タイムスタンプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

executable_id

実行された SQL ステートメント・セクションを一意的に識別する、データ・サーバーで生成されたバイナリー・トークン。詳しくは、モニター・エレメント 1020 ページの『executable_id 実行可能 ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

package_schema

SQL ステートメントに関連したパッケージのスキーマ名。 詳しくは、モニター・エレメント 1249 ページの『package_schema - パッケージ・スキーマ : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

package_name

現在実行中の SQL ステートメントが含まれているパッケージの名前。 詳しくは、モニター・エレメント 1248 ページの『package_name - パッケージ名 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

package_version_id

パッケージ・バージョンは、現在実行中の SQL ステートメントを含むパッケージのバージョン ID を示します。 詳しくは、モニター・エレメント 1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

section_number

現在処理中または最後に処理された SQL ステートメントのパッケージにある内部セクション番号。 詳しくは、モニター・エレメント 1477 ページの『section_number - セクション番号 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

effective_isolation

SQL ステートメントが実行されていたときに有効であった分離の値。 詳しくは、モニター・エレメント 1006 ページの『effective_isolation - 有効な分離 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	

num_executions

SQL ステートメントが実行された回数。詳しくは、モニター・エレメント 1211 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

num_exec_with_metrics

メトリック収集の対象になった SQL ステートメントの実行回数。 詳しくは、モニター・エレメント 1212 ページの『num_exec_with_metrics メトリックが収集された実行数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

prep_time

アクティビティーが SQL ステートメントである場合に SQL ステートメントを準備するために必要な時間 (ミリ秒単位)。 詳しくは、モニター・エレメント 1423 ページの『prep_time 準備時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

last_metrics_update

このキャッシュ項目で最後にメトリックが更新された時刻を示すタイム・スタンプ。 詳しくは、モニター・エレメント 1120 ページの『last_metrics_update - メトリック最終更新タイム・スタンプ : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

num_coord_exec

このセクションがコーディネーター・エージェントによって実行された回数。 詳しくは、モニター・エレメント 1210 ページの『num_coord_exec - コーディネーター・エージェントによる実行数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

num_coord_exec_with_metrics

このセクションがコーディネーター・エージェントによって実行され、モニター・メトリックがキャプチャーされた回数。 詳しくは、モニター・エレメント 1210 ページの『num_coord_exec_with_metrics - メトリック付きの、コーディネーター・エージェントによる実行数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_type_id

ステートメント・タイプの ID。 詳しくは、モニター・エレメント 1541 ページの『stmt_type_id - ステートメント・タイプ ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:long			オプション	

query_cost_estimate

SQL コンパイラーによって判別された照会の見積もりコスト。 詳しくは、モニター・エレメント 1433 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_pkg_cache_id

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_text

SQL ステートメントのテキスト。 詳しくは、モニター・エレメント 1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

comp_env_desc

908 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

これを含むエレメント: 300 ページの『db2_pkgcache_event』

section_env

SQL ステートメントのセクションを含む BLOB。詳しくは、モニター・エレメント 1477 ページの『section_env セクション環境 : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

activity_metrics

このキャッシュ項目のアクティビティー・メトリック。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

routine_id

CALL ステートメントの場合、このエレメントには、呼び出されたストアード・プロシージャーに関連付けられているルーチン ID が入っています。詳しくは、モニター・エレメント 1460 ページの『routine_id - ルーチン ID : モニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

query_data_tag_list

このキャッシュ項目のデータ・タグ・リスト。 詳しくは、モニター・エレメント 1434 ページの『query_data_tag_list - 見積もりの照会データ・タグ・リストのモニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

total_stats_fabrication_time

リアルタイム統計収集によって統計作成に費やされた合計時間 (ミリ秒単位)。 詳しくは、モニター・エレメント 1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_stats_fabrications

照会のコンパイル中にリアルタイム統計によって実行された統計作成の合計数。 詳しくは、モニター・エレメント 1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_sync_runstats_time

リアルタイム統計収集によってトリガーされた同期 RUNSTATS アクティビティーに費やされた合計時間 (ミリ秒単位)。詳しくは、モニター・エレメント 1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_sync_runstats

リアルタイム統計収集によってトリガーされた同期 RUNSTATS アクティビティーの合計数。 詳しくは、モニター・エレメント 1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

max_coord_stmt_exec_timestamp

詳しくは、モニター・エレメント 1179 ページの『max_coord_stmt_exec_timestamp - コーディネーターの最大ステートメント実行のタイム・スタンプのモニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:dateTime	

max_coord_stmt_exec_time_arg

max_coord_stmt_exec_time が記録されたときの、 このステートメントの入力変数を示します。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント: (『stmt_value_index』、 『stmt_value_isreopt』、 308 ページの『stmt_value_isnull』、 308 ページの『stmt_value_type』、 308 ページの『stmt_value_data』、 ANY コンテント (検証しない (skip)) {0 個以上 (*)})

max_coord_stmt_exec_time

詳しくは、モニター・エレメント 1176 ページの『max_coord_stmt_exec_time - コーディネーターの最大ステートメント実行時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 300 ページの『db2_pkgcache_event』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_value_index

このエレメントは、SQL ステートメントで使用される入力パラメーター・マーカーまたはホスト変数の位置を表します。詳しくは、モニター・エレメント 1543 ページの『stmt_value_index 値索引』 を参照してください。

これを含むエレメント: 『max_coord_stmt_exec_time_arg』

stmt_value_isreopt

このエレメントは、変数がステートメント最適化のやり直し中に使用されたかどうかを示します。 詳しくは、モニター・エレメント 1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 『max_coord_stmt_exec_time_arg』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			必須	

stmt_value_isnull

このエレメントは、SQL ステートメントに関連したデータ値が NULL 値かどうかを示します。 詳しくは、モニター・エレメント 1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』 を参照してください。

これを含むエレメント: 307 ページの『max_coord_stmt_exec_time_arg』

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			必須	

stmt_value_type

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

これを含むエレメント: 307 ページの『max_coord_stmt_exec_time_arg』

stmt_value_data

このエレメントは、SQL ステートメントに関連したデータ値のストリング表記です。 詳しくは、モニター・エレメント 1543 ページの『stmt_value_data 値データ』を参照してください。

これを含むエレメント: 307 ページの『max_coord_stmt_exec_time_arg』

パッケージ・キャッシュ・イベント・データの収集およびレポートの生成

パッケージ・キャッシュ・イベント・モニターを使用して、データベースのパッケージ・キャッシュからフラッシュされたステートメント項目に関するデータを収集します。未フォーマット・イベント表にパッケージ・キャッシュ・イベント・データが収集された後、このタスクの指示に従って、テキスト・レポートを取得します。

始める前に

パッケージ・キャッシュ・イベント・モニターのデータを収集するには、DBADM または SQLADM 権限が必要です。

このタスクについて

パッケージ・キャッシュ・イベント・モニターは、パッケージ・キャッシュ内にあったものについて関連履歴情報を収集します。これは、SQL ステートメントの照会のパフォーマンスおよび問題判別に役立ちます。パッケージ・キャッシュ・イベント・モニターが、データベース・パッケージ・キャッシュから収集する情報には、例えば以下の情報が含まれます。

- 実行可能 ID (EXECUTABLE_ID)
- 照会の見積コスト (QUERY_COST_ESTIMATE)
- 項目がパッケージ・キャッシュからフラッシュされた時刻 (Event Timestamp)

このタスクでは、パッケージ・キャッシュ・イベント・データを収集するための手順を説明します。

制約事項

DBADM または SQLADM 権限がない場合、入力データ値は表示できません。

手順

パッケージ・キャッシュ・イベントに関する詳細情報を収集するには、以下のステップを実行します。

1. 次の例に示すように、CREATE EVENT MONITOR FOR PACKAGE CACHE ステートメントを使用して、cachestmtevmon というパッケージ・キャッシュ・イベント・モニターを作成します。

   ```
   CREATE EVENT MONITOR cachestmtevmon FOR PACKAGE CACHE
      WRITE TO UNFORMATTED EVENT TABLE
   ```

2. 以下のステートメントを実行して、cachestmtevmon というパッケージ・キャッシュ・イベント・モニターを活動化します。

   ```
   SET EVENT MONITOR cachestmtevmon STATE 1
   ```

3. ロック・イベント・モニターおよび作業単位イベント・モニターとは異なり、パッケージ・キャッシュ・イベント・モニターは、イベント・モニターが活動化された後、自動的にデータの収集を開始します。
4. データベースに接続します。
5. イベント・モニター情報を収集するアプリケーション、ワークロード、または SQL ステートメントを実行します。
6. パッケージ・キャッシュのデータ収集をオフにするには、以下のコマンドを実行して、イベント・モニターを非アクティブ化します。

   ```
   SET EVENT MONITOR cachestmtevmon STATE 0
   ```

7. XML パーサー・ツール **db2evmonfmt** を使用して、パッケージ・キャッシュ・イベント・レポートを取得します。このツールは、未フォーマット・イベント表に収集されたイベント・データに基づいてフラット・テキスト・レポートを生成します。例えば、次のようにします。

   ```
   java db2evmonfmt -d db_name -ue table_name -ftext -u user_id -p password
   ```

8. レポートを分析します。

例

以下は、パッケージ・キャッシュ・イベント・モニターが未フォーマット・イベント表に収集したデータを、Java ベースのレポート・ツール **db2evmonfmt** を使用して変換することによって得られるレポートの例です。

```
------------------------------------------------------
Event ID              : 1
Event Type            : PKGCACHEBASE
Event Timestamp       : 2009-11-06-12.32.06.442020
Member                : 0
Release               : 9070100
```

```
------------------------------------------------------

Package Cache Details
---------------------
Section Type                          : D
Insert Timestamp                      : 2009-11-06-12.28.04.246930
Executable ID                         : 0100000000000000010000000000000000000000002002009110612280424693 2
Package Schema                        :
Package Name                          :
Package Version ID                    :
Section Number                        :
Effective Isolation                   : CS
Number Of Executions                  : 1
Number Of Executions With Metrics     : 1
Prep Time                             : 9
Last Metrics Update                   : 2009-11-06-12.28.07.905942
Executions By Coordinator             : 1
Executions By Coordinator With Metrics : 1
Statement Type                        : DDL, (not Set Constraints)
Query Cost Estimate                   : 1
Statement Package Cache ID            : 1151051235329
Statement Text                        : create event monitor cachestmtevmon for package cache write to unformatted event table
Compilation Environment               : 47454E5F434D504C010000000E000000800100000000000000000000000000010000000400000000100000
                                        0000000020000000400000008010000000000003000000080000001001000000000B0FA040000000100000018
                                        010000B804B0FA0500000001000000200100 0B80400000600000001000000280100 0B804000007000000010
                                        0000030010000B80400000800000001000000380100 0B80400000900000005000000040010000B80400000A00
                                        0000080000004801000 0B80400000B0000000B0000005001000 0B80400001000000000A0000006001000 0B8040
                                        0001300000004000000700100000000000000F0000000400000078010000000000001000000000000000500000
                                        0000000000000000000000000300000000000000030000000000000004E0000000000000004E0000000000000
                                        04E0000000000000031202020200000002020444444444444000000000000000000000C0000000002009110 6
                                        1228040000000000000000000000000000000000000100000000000000
Section Environment                   :

Metrics
-------------------
WLM_QUEUE_TIME_TOTAL                : 0
WLM_QUEUE_ASSIGNMENTS_TOTAL         : 0
FCM_TQ_RECV_WAIT_TIME               : 0
FCM_MESSAGE_RECV_WAIT_TIME          : 0
FCM_TQ_SEND_WAIT_TIME               : 0
FCM_MESSAGE_SEND_WAIT_TIME          : 0
LOCK_WAIT_TIME                      : 0
LOCK_WAITS                          : 0
DIRECT_READ_TIME                    : 0
DIRECT_READ_REQS                    : 0
DIRECT_WRITE_TIME                   : 3
DIRECT_WRITE_REQS                   : 1
LOG_BUFFER_WAIT_TIME                : 0
NUM_LOG_BUFFER_FULL                 : 0
LOG_DISK_WAIT_TIME                  : 0
LOG_DISK_WAITS_TOTAL                : 0
POOL_WRITE_TIME                     : 0
POOL_READ_TIME                      : 33
AUDIT_FILE_WRITE_WAIT_TIME          : 0
AUDIT_FILE_WRITES_TOTAL             : 0
AUDIT_SUBSYSTEM_WAIT_TIME           : 0
AUDIT_SUBSYSTEM_WAITS_TOTAL         : 0
DIAGLOG_WRITE_WAIT_TIME             : 0
DIAGLOG_WRITES_TOTAL                : 0
FCM_SEND_WAIT_TIME                  : 0
FCM_RECV_WAIT_TIME                  : 0
TOTAL_ACT_WAIT_TIME                 : 36
TOTAL_SECTION_SORT_PROC_TIME        : 0
TOTAL_SECTION_SORTS                 : 0
TOTAL_SECTION_SORT_TIME             : 0
TOTAL_ACT_TIME                      : 37
TOTAL_ROUTINE_TIME                  : 0
STMT_EXEC_TIME                      : 3658
COORD_STMT_EXEC_TIME                : 3658
TOTAL_ROUTINE_NON_SECTION_PROC_TIME : 0
TOTAL_ROUTINE_NON_SECTION_TIME      : 0
TOTAL_SECTION_PROC_TIME             : 1
TOTAL_SECTION_TIME                  : 37
TOTAL_ROUTINE_USER_CODE_PROC_TIME   : 0
TOTAL_ROUTINE_USER_CODE_TIME        : 0
ROWS_READ                           : 19
ROWS_MODIFIED                       : 3
POOL_DATA_L_READS                   : 42
POOL_INDEX_L_READS                  : 83
POOL_TEMP_DATA_L_READS              : 0
POOL_TEMP_INDEX_L_READS             : 0
POOL_XDA_L_READS                    : 0
POOL_TEMP_XDA_L_READS               : 0
TOTAL_CPU_TIME                      : 2243
POOL_DATA_P_READS                   : 13
POOL_TEMP_DATA_P_READS              : 0
POOL_XDA_P_READS                    : 0
POOL_TEMP_XDA_P_READS               : 0
POOL_INDEX_P_READS                  : 33
POOL_TEMP_INDEX_P_READS             : 0
```

```
POOL_DATA_WRITES                : 0
POOL_XDA_WRITES                 : 0
POOL_INDEX_WRITES               : 0
DIRECT_READS                    : 0
DIRECT_WRITES                   : 2
ROWS_RETURNED                   : 0
DEADLOCKS                       : 0
LOCK_TIMEOUTS                   : 0
LOCK_ESCALS                     : 0
FCM_SENDS_TOTAL                 : 0
FCM_RECVS_TOTAL                 : 0
FCM_SEND_VOLUME                 : 0
FCM_RECV_VOLUME                 : 0
FCM_MESSAGE_SENDS_TOTAL         : 0
FCM_MESSAGE_RECVS_TOTAL         : 0
FCM_MESSAGE_SEND_VOLUME         : 0
FCM_MESSAGE_RECV_VOLUME         : 0
FCM_TQ_SENDS_TOTAL              : 0
FCM_TQ_RECVS_TOTAL              : 0
FCM_TQ_SEND_VOLUME              : 0
FCM_TQ_RECV_VOLUME              : 0
TQ_TOT_SEND_SPILLS              : 0
POST_THRESHOLD_SORTS            : 0
POST_SHRTHRESHOLD_SORTS         : 0
SORT_OVERFLOWS                  : 0
AUDIT_EVENTS_TOTAL              : 0
TOTAL_SORTS                     : 0
THRESH_VIOLATIONS               : 0
NUM_LW_THRESH_EXCEEDED          : 0
TOTAL_ROUTINE_INVOCATIONS       : 0
```

パッケージ・キャッシュ情報を使用して、パフォーマンス調整の対象候補となるステートメントを識別する:

パッケージ・キャッシュ・イベント・モニターをメモリー内のメトリックと一緒に使用すると、パッケージ・キャッシュのどのステートメントの実行コストが高いかを識別できます。実行に長い時間がかかるステートメントを把握できたなら、それらに対してパフォーマンス調整を実行できます。

始める前に

CREATE EVENT MONITOR ステートメントでは、イベント・モニターによって生成される未フォーマット・イベント (UE) 表を格納するために最低 8 K のページ・サイズの表スペースが必要となります。表スペースを CREATE EVENT MONITOR ステートメントで明示的に指定しない限りは、データベースのデフォルトの表スペースが使用されます。

このタスクについて

このタスクでは、合計 CPU 時間の観点からコストが高いステートメントを検出するために、ある期間にシステムで実行されたすべての作業を調べる方法が示されています。パッケージ・キャッシュ・イベント・モニターを、(MON_GET_PKG_CACHE_STMT 表関数または MON_GET_PKG_CACHE_STMT_DETAILS 表関数によって戻される) メモリー内のモニター・エレメントに反映されるパッケージ・キャッシュ情報と一緒に使用するのが役立ちます。キャッシュ内のステートメント、およびキャッシュから追い出されたステートメントの両方を把握できるからです。コストの高いステートメントが識別できたら、これらのステートメントに対してパフォーマンス調整を実行できます。

注: 実行コストの高いステートメントを判別する際に使用するモニター・エレメントは、多数の中から選択できます。この例では、CPU 時間が使用されています (1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』)。この測定では、消費される実際の CPU リソースが表示され、ロック待機時間や、ステ

ートメント実行中の他の消費時間は反映されていません。代わりに、ステートメント実行時間 (1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』) を使用するよう選択することもできます。これには、セクション内のすべてのエージェントが費やした時間、および待機時間などが含まれます。また、パッケージ・キャッシュ・イベント・モニターによって戻される他の多数の消費時間エレメントから選択することも可能です。選択可能なモニター・エレメントについて詳しくは、 290 ページの『パッケージ・キャッシュ・イベント・モニターの場合に EVMON_FORMAT_UE_TO_TABLES によってリレーショナル表に書き込まれる情報』または 300 ページの『パッケージ・キャッシュ・イベント・モニター用に XML に書き込まれる情報』を参照してください。

制約事項

ここで使用されている例の場合、分析対象のステートメントの長さは 3000 文字に制限されます。この制約は、ステートメントで使用される GROUP BY 節に起因しています。この節を **stmt_text** モニター・エレメントなどの LOB 値と一緒に使用することはできません。

手順

1. ステートメントがパッケージ・キャッシュから削除される (追い出される) 際に、そうしてステートメントをキャプチャーするためのパッケージ・キャッシュ・イベント・モニターを作成します。 例えば、EXPENSIVESTMTS というイベント・モニターを作成するには、以下の SQL を使用できます。

   ```
   CREATE EVENT MONITOR EXPENSIVESTMTS FOR PACKAGE CACHE WRITE TO UNFORMATTED EVENT TABLE
   ```

   このステートメントは、イベント・モニター EXPENSIVESTMTS と同じ名前で、データベースのデフォルト表スペースにある UE 表に書き込むパッケージ・キャッシュ・イベント・モニターを作成します。TABLE *table-name* 節を使用して、UE 表のデフォルト名をオーバーライドできます。また、**IN** *tablespace-name* 節を使用すると、UE 表に使用される表スペースをオーバーライドできます。

   デフォルトでは、パッケージ・キャッシュから追い出されるすべてのステートメントは、パッケージ・キャッシュ・イベント・モニターによってキャプチャーされます。収集される情報量を制限するには、CREATE EVENT MONITOR ステートメントの一部として、収集される情報を制限するオプションを指定できます。詳しくは、CREATE EVENT MONITOR (パッケージ・キャッシュ) ステートメントに関する資料を参照してください。

2. 次に、イベント・モニターを活動状態にします。

   ```
   SET EVENT MONITOR EXPENSIVESTMTS STATE 1
   ```

   注: デフォルトでは、このイベント・モニターはデータベースの活動化時に自動的に開始されます。AUTOSTART オプションがデフォルトで適用されるためです。ただし、このイベント・モニターは既にアクティブなデータベースで作成されるので、**SET EVENT MONITOR** コマンドを使用して手動で開始する必要があります。

3. データベースに接続し、パフォーマンス分析の対象となるステートメント、ワークロード、またはアプリケーションを実行します。収集する情報量は、希望に応じて決めることができます。ただし、このタイプのパフォーマンス調整は、対象のアプリケーションやワークロードが定期的に実行される場合に最も効率的にな

ります。その他の場合には、以前に実行されたステートメントに対して加えた調整が、今後実行されるステートメントに対して全く影響を及ぼさないことがあります。

4. データの収集が終了したら、イベント・モニターを非アクティブ化状態にします。

```
SET EVENT MONITOR EXPENSIVESTMTS STATE 0
```

5. EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用して、イベント・モニターによりデータが追加された UE 表からデータを抽出します。

```
CALL EVMON_FORMAT_UE_TO_TABLES ('PKGCACHE', NULL, NULL, NULL, NULL, NULL,
 NULL, -1, 'SELECT * FROM EXPENSIVESTMTS')
```

このプロシージャーでは、イベント・モニターによって生成された UE 表 TRACKSTMTS を調べます。 UE 表のすべてのレコードが選択され、その中から、パッケージ・キャッシュ・イベント・モニターによって収集されたデータに基づいて以下の 2 つのリレーショナル表が作成されます。

- PKGCACHE_EVENT
- PCKCACHE_METRICS

1 番目の表には、キャプチャーされた各イベントと関連する、最も頻繁に使用されるモニター・エレメントとメトリックが含まれます。2 番目の表には、各イベントの詳細メトリックが入っています。

注: PKGCACHE_METRICS の列に入っている値は、PKGCACHE_EVENT 表の METRICS 列に含まれている XML 文書にもあります。列指向のアクセスを行うためにより便利な方法として、PKGCACHE_METRICS 表でも提供されています。

6. イベント・モニターからの出力を照会し、実行により時間のかかったステートメントを判別します。この例では、合計 CPU 時間 (1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』) が全体のコストを判別するために使用される消費時間モニター・エレメントです。

```
WITH STMTS AS
     (
1    SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME, EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT
     FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,NULL,-2)) AS T
     GROUP BY EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000)
     UNION ALL
2    SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME, EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT
     FROM PKGCACHE_EVENT E, PKGCACHE_METRICS M WHERE E.XMLID = M.XMLID
     GROUP BY EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000)
     )
SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME, STMT_TEXT, EXECUTABLE_ID
FROM STMTS
GROUP BY EXECUTABLE_ID, STMT_TEXT
ORDER BY TOTAL_EXEC_TIME DESC
FETCH FIRST 10 ROWS ONLY;
```

前述の例の場合、MON_GET_PKG_CACHE_STMT 表関数によって戻されたデータ (1 参照) と、パッケージ・キャッシュ・イベント・モニターによって戻されたデータ (2 参照) の両方が取り出されます。両方のデータ・セットを調べると、依然としてパッケージ・キャッシュ内に存在するステートメントのデータと、パッケージ・キャッシュから既に追い出されたステートメントのデータの両方を把握できます。これにより、どのステートメントの実行コストが高いかを評価する際に、ある特定の期間に実行されたすべてのステートメントを考慮に入れることができます。 この照会は、以下の結果を返します。

注: 出力上の目的のため、以下の出力サンプルを構成する文字のフォント・サイズは小さくなっています。以下の出力は、DB2 インフォメーション・センターにあるこのトピック (『パッケージ・キャッシュ情報を使用して、パフォーマンス調整の対象候補となるステートメントを識別する』) のオンライン・バージョンよりも読みやすい場合があります。

```
TOTAL_CPU_TIME       STMT_TEXT                     EXECUTABLE_ID
-------------------- ----------------------------- --------------------------------------------------------------------------------
              656250 CALL EVMON_FORMAT_UE_TO_TABLES x'0100000000000000070000000000000000000000002002010120712575922100
SQL0445W  Value "WITH STMTS AS ( SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME" has been truncated.  SQLSTATE=01004
              500000 CALL XSR_COMPLETE(?,?,NULL,1)  x'0100000000000000160000000000000000000000002002010120712575801112004'
              156250 CALL XSR_ADDSCHEMADOC(?,?,?,?, x'0100000000000000090000000000000000000000002002010120712575987700
0'
              156250 CREATE INDEX PKGCACHE_EVENT_IN x'0100000000000000120000000000000000000000002002010120712575800565003'
               93750 CALL XSR_REGISTER(?,?,?,?, NUL x'0100000000000000080000000000000000000000002002010120712575964300
0'
               93750 CALL XDB_DECOMP_XML_FROM_QUERY x'0100000000000000180000000000000000000000002002010120712575801862001'
               78125 CREATE INDEX PKGCACHE_METRICS_ x'0100000000000000140000000000000000000000002002010120712575800924000'
               46875 CREATE EVENT MONITOR EXPENSIVE x'0100000000000000010000000000000000000000002002010120712575758299000'
               46875 SET EVENT MONITOR EXPENSIVESTM x'0100000000000000050000000000000000000000002002010120712575758768001'
               46875 CALL SYSPROC.SYSINSTALLOBJECTS x'0100000000000000240000000000000000000000002002010120712575936286002'

  10 record(s) selected with 1 warning messages printed.
```

注: 見やすく表示するために、STMT_TEXT 列は省略されています。

次のタスク

ステップ 6 (313 ページ) に示されている照会からの出力を使用して、調整するステートメントを判別します。

パフォーマンス向上の可能性を探るためにパッケージ・キャッシュ情報と db2advis を使用する:

DB2 設計アドバイザーは SQL ステートメントを分析し、データベースのパフォーマンスを向上させる方法についての推奨を行うことができます。

パッケージ・キャッシュからのステートメント (パッケージ・キャッシュ・イベント・モニターがキャプチャーしたステートメントを含む) を設計アドバイザーへの入力として使用すると、指定のワークロードに関する (またはある期間に実行されるすべてのステートメントに関しても) パフォーマンスを向上させるために加えることができる変更を識別できます。

始める前に

- CREATE EVENT MONITOR ステートメントでは、イベント・モニターによって生成される未フォーマット・イベント (UE) 表を格納するために最低 8 K のページ・サイズの表スペースが必要となります。表スペースを CREATE EVENT MONITOR ステートメントで明示的に指定しない限りは、データベースのデフォルトの表スペースが使用されます。
- 設計アドバイザーが必要とする Explain 表が作成済みでなければなりません。

このタスクについて

このタスクでは、パッケージ・キャッシュ・イベント・モニターを使用して、ある期間にシステムで実行されるすべての作業をトラッキングする方法、および **db2advis** コマンドを使用してその期間に実行された実行コストの高いステートメントを分析する方法が示されています。db2advis コマンドの出力は、パッケージ・キャッシュ・イベント・モニターがアクティブだった間に実行されたステートメントに基づいて、データベースのパフォーマンスを向上させるためにデータベースに加えることができる調整や変更について提案します。当該のステートメントがパッケ

ージ・キャッシュにはもうない場合、パッケージ・キャッシュ・イベント・モニターを使用してこうしてステートメントをキャプチャーすると役立ちます。

制約事項

ここで使用されている例の場合、分析対象のステートメントの長さは 3000 文字に制限されます。この制約は、ステートメントで使用される GROUP BY 節に起因しています。この節を **stmt_text** モニター・エレメントなどの LOB 値と一緒に使用することはできません。

手順

1. ステートメントがパッケージ・キャッシュから削除される (追い出される) 際に、そうしてステートメントをキャプチャーするためのパッケージ・キャッシュ・イベント・モニターを作成します。 例えば、TRACKSTMTS というイベント・モニターを作成するには、以下の SQL を使用することができます。

   ```
   CREATE EVENT MONITOR TRACKSTMTS FOR PACKAGE CACHE WRITE TO UNFORMATTED EVENT TABLE
   ```

   このステートメントは、イベント・モニター TRACKSTMTS と同じ名前で、UE 表に書き込むパッケージ・キャッシュ・イベント・モニターを作成します。
2. 次に、イベント・モニターを活動状態にします。

   ```
   SET EVENT MONITOR TRACKSTMTS STATE 1
   ```
3. データベースに接続し、パフォーマンス分析の対象となるステートメント、ワークロード、またはアプリケーションを実行します。収集する情報量は、希望に応じて決めることができます。ただし、このタイプのパフォーマンス調整は、対象のアプリケーションやワークロードが定期的に実行される場合に最も効率的になります。その他の場合には、以前に実行されたステートメントに対して加えた調整が、今後実行されるステートメントに対して全く影響を及ぼさないことがあります。
4. データの収集が終了したら、イベント・モニターを非アクティブ化状態にします。

   ```
   SET EVENT MONITOR TRACKSTMTS STATE 0
   ```
5. EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用して、イベント・モニターによりデータが追加された UE 表からデータを抽出します。

   ```
   CALL EVMON_FORMAT_UE_TO_TABLES
       ('PKGCACHE', NULL, NULL, NULL, NULL, NULL, NULL, -1,
        'SELECT * FROM TRACKSTMTS')
   ```

   このプロシージャーにより、パッケージ・キャッシュ・イベント・モニターが収集したデータに基づいて以下の 2 つのリレーショナル表が作成されます。

   - PKGCACHE_EVENT
   - PCKCACHE_METRICS

   1 番目の表には、キャプチャーされた各イベントと関連する、最も頻繁に使用されるモニター・エレメントとメトリックが含まれます。2 番目の表には、各イベントの詳細メトリックが入っています。

注: PKGCACHE_METRICS の列に入っている値は、PKGCACHE_EVENT 表の METRICS 列に含まれている XML 文書にもあります。列指向のアクセスを行うためにより便利な方法として、PKGCACHE_METRICS 表でも提供されています。

6. イベント・モニターからの出力を照会し、実行により時間のかかったステートメントを判別します。この例では、ステートメント実行時間 (1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』) が全体のコストを判別するために使用される消費時間モニター・エレメントです。このモニター・エレメントは、すべてのデータベース・パーティションで合算されます。

   ヒント: 照会からの出力をテキスト・ファイルに保存します。次のステップでこのファイルを使用します。

```
WITH STMTS AS
(
  SELECT SUM(TOTAL_STMT_EXEC_TIME)/SUM(TOTAL_NUM_COORD_EXEC_WITH_METRICS) AS AVG_TIME_PER_EXEC,
         STMT_TEXT, SUM(NUM_EXECUTIONS) AS NUM_EXECUTIONS, STMT_TYPE_ID
  FROM (
        (
         SELECT    SUM(STMT_EXEC_TIME) AS TOTAL_STMT_EXEC_TIME,
                   SUM(NUM_COORD_EXEC_WITH_METRICS) AS TOTAL_NUM_COORD_EXEC_WITH_METRICS,
                   SUM(NUM_COORD_EXEC) AS NUM_EXECUTIONS,
                   VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT,
                   STMT_TYPE_ID
         FROM      PKGCACHE_EVENT AS E, PKGCACHE_METRICS AS M
         WHERE     E.XMLID = M.XMLID
          AND      NUM_COORD_EXEC_WITH_METRICS > 0
         GROUP BY  VARCHAR(STMT_TEXT, 3000),STMT_TYPE_ID
         ORDER BY TOTAL_NUM_COORD_EXEC_WITH_METRICS DESC
         FETCH FIRST 50 ROWS ONLY
        )
        UNION ALL
        (
         SELECT    SUM(STMT_EXEC_TIME) AS TOTAL_STMT_EXEC_TIME,
                   SUM(NUM_COORD_EXEC_WITH_METRICS) AS TOTAL_NUM_COORD_EXEC_WITH_METRICS,
                   SUM(NUM_COORD_EXEC) AS NUM_EXECUTIONS,
                   VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT,
                   STMT_TYPE_ID
         FROM      TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,NULL,-2)) AS T
         WHERE     NUM_COORD_EXEC_WITH_METRICS > 0
         GROUP BY  VARCHAR(STMT_TEXT, 3000),STMT_TYPE_ID
         ORDER BY TOTAL_NUM_COORD_EXEC_WITH_METRICS DESC
         FETCH FIRST 50 ROWS ONLY
        )
       ) AS Q_UA
  GROUP BY STMT_TEXT, STMT_TYPE_ID
)
SELECT      '--# SET FREQUENCY ' || NUM_EXECUTIONS || X'0A' || STMT_TEXT || ';'
FROM        STMTS  WHERE STMT_TYPE_ID LIKE 'DML, Select%' OR STMT_TYPE_ID LIKE 'DML, Insert%' 1
ORDER BY    AVG_TIME_PER_EXEC DESC
FETCH FIRST 50 ROWS ONLY;
```

前述のサンプル・ステートメントでは、パッケージ・キャッシュ・イベント・モニターからのデータと、MON_GET_PKG_CACHE_STMT 表関数からのメモリー内の情報の両方が取り出されます。両方のデータ・セットを調べると、パッケージ・キャッシュから追い出されたステートメントのデータと、依然としてパッケージ・キャッシュ内に存在するステートメントのデータを把握できます。このようにすると、実行コストの高いステートメントを評価する際に、キャッシュからまだ追い出されていないステートメントも対象として含めることができます。この照会は実行ごとに、ステートメントが実行された回数に基づいて、アクティブ・パッケージ・キャッシュとパッケージ・キャッシュ・イベント・モニターの両方から上位 50 個のステートメントを取り出します。その後、これらのステー

トメントのうち、ステートメントの実行時間の平均長さに基づいて、上位 50 個の SELECT ステートメントまたは INSERT ステートメントが選択されます (**1**)。

注: 実行コストの高いステートメントを判別する際に使用するモニター・エレメントは、多数の中から選択できます。この例では、ステートメント実行時間が使用されています。この測定には、このセクションを実行しているすべてのメンバーとエージェントが実行に費やした時間と、待機時間などが含まれています。代わりに CPU 時間 (1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』) を使用するよう選択することもできます。これは、ステートメントの処理に CPU が費やした時間だけをレポートします。また、パッケージ・キャッシュ・イベント・モニターによって戻される他の多数の消費時間エレメントから選択することも可能です。選択可能なモニター・エレメントについて詳しくは、 290 ページの『パッケージ・キャッシュ・イベント・モニターの場合に EVMON_FORMAT_UE_TO_TABLES によってリレーショナル表に書き込まれる情報』または 300 ページの『パッケージ・キャッシュ・イベント・モニター用に XML に書き込まれる情報』を参照してください。

さらに、この照会には `--# SET FREQUENCY` というフォーマットの出力があり、これは設計アドバイザーが分析で使用します。 前述の照会によって、以下のような結果が戻ります。

```
------------------------------------------------------------------------------------------------------------------------------
--# SET FREQUENCY 1
WITH STMTS AS ( SELECT SUM(TOTAL_STMT_EXEC_TIME)/SUM(TOTAL_NUM_COORD_EXEC_WITH_METRICS) AS AVG_TIME_PER_EXEC, STMT
--# SET FREQUENCY 2
WITH STMTS AS ( SELECT SUM(TOTAL_CPU_TIME) AS TOTAL_CPU_TIME, EXECUTABLE_ID, VARCHAR(STMT_TEXT, 3000) AS STMT_TEXT
--# SET FREQUENCY 1055
SELECT POLICY FROM SYSTOOLS.POLICY WHERE MED='DB2CommonMED' AND DECISION='NOP' AND NAME='CommonPolicy';
--# SET FREQUENCY 99
SELECT CREATOR, NAME, CTIME FROM SYSIBM.SYSTABLES WHERE TYPE='T' OR TYPE='S' OR TYPE='N' WITH UR;
--# SET FREQUENCY 1
UPDATE SYSTOOLS.HMON_ATM_INFO SET STATS_LOCK = 'N', REORG_LOCK = 'N';
--# SET FREQUENCY 1
UPDATE SYSTOOLS.HMON_ATM_INFO AS ATM SET STATS_FLAG = 'N', REORG_FLAG = 'N' WHERE (ATM.SCHEMA, ATM.NAME) IN   (SEL
--# SET FREQUENCY 1
SELECT POLICY FROM SYSTOOLS.POLICY WHERE MED='DB2TableMaintenanceMED' AND DECISION='TableRunstatsDecision' AND NAM
--# SET FREQUENCY 83
WITH JTAB(JSCHEMA,JNAME) AS (VALUES(TABLE_SCHEMA(CAST(? AS varchar(128)), CAST(? AS varchar(128))), TABLE_NAME (CA
--# SET FREQUENCY 122
WITH VTYPED (NAME, SCHEMA) AS (VALUES(TABLE_NAME (CAST(? AS varchar(128)), CAST(? AS varchar(128))), TABLE_SCHEMA(
--# SET FREQUENCY 1210
SELECT COLNAME, TYPENAME FROM  SYSCAT.COLUMNS WHERE TABNAME='POLICY' AND TABSCHEMA='SYSTOOLS';
--# SET FREQUENCY 105
SELECT TABNAME FROM SYSCAT.TABLES  WHERE TABNAME='HMON_ATM_INFO' AND  TABSCHEMA='SYSTOOLS';
--# SET FREQUENCY 104
DELETE FROM SYSTOOLS.HMON_ATM_INFO AS ATM WHERE NOT EXISTS ( SELECT * FROM SYSIBM.SYSTABLES AS IBM WHERE ATM.NAME
--# SET FREQUENCY 1118
VALUES(SUBSTR(:H00003      ,:H00014,:H00015 ))         INTO :H00009:H00017    ;
--# SET FREQUENCY 274
INSERT INTO "ASRISK"."PKGCACHE_EVENT"("EVENT_ID","XMLID","EVENT_TYPE","EVENT_TIMESTAMP","MEMBER","SECTION_TYPE","I
--# SET FREQUENCY 1
SELECT IBM.TID, IBM.FID FROM SYSIBM.SYSTABLES AS IBM, SYSTOOLS.HMON_ATM_INFO AS ATM WHERE ATM.STATS_FLAG <> 'Y' AN
--# SET FREQUENCY 115
VALUES(SUBSTR(CAST(? AS CLOB(162)),CAST(? AS INTEGER),CAST(? AS INTEGER)));
--# SET FREQUENCY 8227

⋮

-# SET FREQUENCY 532
SELECT TBNAME, TBCREATOR FROM "ASRISK  ".SYSINDEXES WHERE NAME = 'INDCOLUMNS01' AND CREATOR = 'SYSIBM  ';
--# SET FREQUENCY 105
SELECT TABNAME FROM SYSCAT.TABLES  WHERE TABNAME='HMON_COLLECTION' AND  TABSCHEMA='SYSTOOLS';
--# SET FREQUENCY 4091
SELECT STATS_LOCK, REORG_LOCK FROM SYSTOOLS.HMON_ATM_INFO WHERE SCHEMA = ? AND NAME = ? AND CREATE_TIME = ? FOR UP
--# SET FREQUENCY 17100
SELECT CREATE_TIME FROM SYSTOOLS.HMON_ATM_INFO WHERE SCHEMA = ? AND NAME = ? FOR UPDATE;
--# SET FREQUENCY 524
SELECT COUNT(*) FROM "SYSIBM".SYSTABLES WHERE NAME = 'SYSDATAPARTITIONEXPRESSION' AND CREATOR = 'SYSIBM  ' AND TYP
--# SET FREQUENCY 532
SELECT COUNT(*) FROM "SYSIBM".SYSTABLES WHERE NAME = 'SYSCOLUMNS' AND CREATOR = 'SYSIBM  ' AND TYPE = 'S';

  47 record(s) selected
```

注: 前述の出力サンプル内の行は、見やすく表示するために省略されています。

7. ステップ 6 (316 ページ) で照会によって戻されたステートメントを使用して、**db2advis** コマンドの入力ファイルを作成します (**db2advis** コマンドの入力ファイルの作成について詳しくは、このコマンドの参照資料をご覧ください)。
8. ステップ 7 で作成された入力ファイルを使用して、**db2advis** コマンドを実行します。 例えば、作成した入力ファイルが pkgcache_stmts.txt という名前の場合、以下のようなコマンドを実行します。

   ```
   db2advis -d customer -i pkgcache_stmts.txt -m MICP
   ```

   where
   - **-d CUSTOMER** は、推奨を受ける対象のデータベース名です。
   - **-i pkgcache_stmts.txt** は、**db2advis** の入力ファイル名です。
   - **-m MICP** は、パフォーマンスを向上させるための以下の推奨を生成する db2advis コマンドに対するディレクティブです。
     - **M** 新規マテリアライズ照会表
     - **I** 新規索引
     - **C** 標準表のマルチディメンション・クラスタリング表 (MDC) への変換
     - **P** 既存の索引の再パーティショニング

タスクの結果

設計アドバイザーによって、以下のような推奨が戻されます。

```
execution started at timestamp 2010-03-16-14.25.57.562000
Using the default table space name USERSPACE1
found [47] SQL statements from the input file
excluding statement [0] from the workload.
excluding statement [1] from the workload.
excluding statement [19] from the workload.
excluding statement [39] from the workload.
Recommending indexes...
Recommending MQTs...
Recommending Multi-Dimensional Clusterings...
Found 19 user defined views in the catalog table
Found [17] candidate MQTs
Getting cost of workload with MQTs
total disk space needed for initial set [   0.159] MB
total disk space constrained to         [  69.215] MB
  2  indexes in current solution
  0  MQTs in current solution
total disk space needed for initial set [   0.024] MB
total disk space constrained to         [ 103.822] MB
No useful Multi-dimensional Clustering dimensions for this workload
 [5651.8281] timerons  (without recommendations)
 [5519.8281] timerons  (with current solution)
 [2.34%] improvement


--
--
-- LIST OF MODIFIED CREATE-TABLE STATEMENTS WITH RECOMMENDED PARTITIONING KEYS AND TABLESPACES AND/OR RECOMMENDED MULTI-DIMENSIONAL CLUSTERINGS
-- ===========================
--  No new partitioning keys or tablespaces are recommended for this workload.


--
--
-- LIST OF RECOMMENDED MQTs
-- ===========================


--
--
-- RECOMMENDED EXISTING MQTs
-- ===========================


--
--
-- UNUSED EXISTING MQTs
-- ===========================
-- DROP TABLE "ASRISK  "."ADEFUSR";
```

```
--
--
-- RECOMMENDED CLUSTERING INDEXES
-- ============================


--
--
-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1],    0.024MB
   CREATE INDEX "ASRISK  "."IDX0031618305300000" ON "ASRISK  "."SYSINDEXES"
   ("CREATOR" ASC, "NAME" ASC, "TBCREATOR" ASC, "TBNAME"
   ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS;
   COMMIT WORK ;


--
--
-- RECOMMENDED EXISTING INDEXES
-- ============================
-- RUNSTATS ON TABLE "SYSTOOLS"."POLICY" FOR SAMPLED DETAILED INDEX "SYSTOOLS"."POLICY_UNQ" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSTOOLS"."HMON_ATM_INFO" FOR SAMPLED DETAILED INDEX "SYSTOOLS"."ATM_UNIQ" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM  "."SYSDATAPARTITIONS" FOR SAMPLED DETAILED INDEX "SYSIBM  "."INDDATAPARTITIONS03" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM  "."SYSTABLES" FOR SAMPLED DETAILED INDEX "SYSIBM  "."INDTABLES01" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM  "."SYSTABLESPACES" FOR SAMPLED DETAILED INDEX "SYSIBM  "."INDTABLESPACES04" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM  "."SYSCOLUMNS" FOR SAMPLED DETAILED INDEX "SYSIBM  "."INDCOLUMNS01" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM  "."SYSINDEXES" FOR SAMPLED DETAILED INDEX "SYSIBM  "."INDINDEXES02" ;
-- COMMIT WORK ;
-- RUNSTATS ON TABLE "SYSIBM  "."SYSTRIGGERS" FOR SAMPLED DETAILED INDEX "SYSIBM  "."INDTRIGGERS02" ;
-- COMMIT WORK ;


--
--
-- UNUSED EXISTING INDEXES
-- ===========================
-- DROP INDEX "ASRISK  "."PKGCACHE_EVENT_IND1";
-- ===========================
--

-- ====ADVISOR DETAILED XML OUTPUT=============
-- ==(Benefits do not include clustering recommendations)==
```

⋮

注: 設計アドバイザーからの出力は、見やすく表示するために省略されています。

次のタスク

パフォーマンスを向上させるためにデータベースにどのような変更を加えるかを判別する際に、設計アドバイザーからの出力を使用すると役立ちます。

アクティビティー・イベント・モニター

アクティビティー・イベント・モニターは、システムで実行されるアクティビティーに関するデータをキャプチャーします。このイベント・モニターを使用すると、ステートメントのパフォーマンスと動作、ならびにシステム全般のロードのパフォーマンスと動作をより深く理解するために役立つデータを収集できます。

アクティビティー・イベント・モニターは、システムで各アクティビティーが完了した後に情報を記録します。一方、作業単位イベント・モニターは、各トランザクションの完了時にデータを記録します。アクティビティー・イベント・モニターを使用すると、個々のステートメントの実行に関するモニター・エレメントを調べることができます。

アクティビティー・イベント・モニターが戻すデータは、以下の表関数が戻すデータを補完するものです。

- MON_GET_ACTIVITY_DETAILS
- MON_GET_PKG_CACHE_STMT
- MON_GET_PKG_CACHE_STMT_DETAILS

このイベント・モニターは、システムで実行されたアクティビティーの履歴情報を戻しますが、これらの表関数は、システムで実行完了したアクティビティー、または最近実行されたアクティビティーの情報を提供します。

アクティビティー・イベント・モニターの用途

他のイベント・モニターと一緒に使用する

アクティビティー・イベント・モニターは、他のイベント・モニターと一緒に使用すると特に有用です。例えば、定義したしきい値に違反するステートメントの実行に関する情報をキャプチャーする必要があるとします。この場合、次の手順を実行します。

1. CREATE THRESHOLD ステートメントを使用してしきい値を定義します。しきい値を定義する一環として、COLLECT ACTIVITY DATA 節を指定して、アクティブなアクティビティー・イベント・モニターでアクティビティー・データを記録するように指定します。
2. しきい値がいつ違反されたのかを示す詳細情報を、その時にシステムで何が発生していたのかを示す他のデータと一緒にキャプチャーする、しきい値違反イベント・モニターを作成します。
3. しきい値違反によって生成されるアクティビティー情報をキャプチャーするアクティビティー・イベント・モニターを作成します。
4. アプリケーションまたはワークロードを実行します。
5. イベント・モニターの出力を照会して、しきい値が違反された時に何が発生していたのかを示す情報を参照します。しきい値違反イベント・モニターのデータと、アクティビティー・イベント・モニターのデータの結合を実行すれば、違反発生時に実行されていたステートメントを特定できます。

このプロセスについては、338 ページの『例: ステートメントの実行に関連したアクティビティー情報のキャプチャー』で詳しく説明しています。

アクティビティー・イベント・モニターには他にも次のような用途があります。

- 実行時間が長い照会に関する情報をキャプチャーします。この場合は、アクティビティーが終了する前に、WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを実行してアクティビティーに関する情報の収集を実施しておきます。このプロシージャーの実行は、長時間実行されているステートメントを強制終了するだけでなく、そのステートメントに関する情報をキャプチャーする必要がある場合に有用です。
- 特定のワークロード内のどのステートメント・アプリケーションが実行されているのかを調べます。

コマンド、プロシージャー、またはツールへの入力

アクティビティー・イベント・モニターによって生成されたデータを、次に示すものを含め、さまざまなツールおよびストアード・プロシージャーの入力値として使用することができます。

db2advis - DB2 設計アドバイザー・コマンド

db2advis コマンドは、アクティビティー・イベント・モニターの出力を使用して、以下の項目およびアクティビティーに関する推奨事項を生成します。

- マテリアライズ照会表 (MQT)
- 索引
- 表の再パーティション化
- マルチディメンション・クラスタリング (MDC) 表への変換
- 使用されていないオブジェクトの削除

db2expln - SQL および XQuery Explain コマンド

db2expln は、アクティビティー・イベント・モニターから得たセクション情報を使用して、セクションに関連するステートメントのアクセス・プランを作成することができます。

EXPLAIN_FROM_ACTIVITY ストアード・プロシージャー

EXPLAIN_FROM_ACTIVITY ストアード・プロシージャーは、アクティビティー・イベント・モニターから得たセクションの内容を使用して、特定のステートメントの実行を Explain します。Explain の出力は Explain 表に格納され、Explain ツール (例えば **db2exfmt** コマンド) を使ってこれを処理できます。 Explain の出力には、存在する場合、アクセス・プランとセクション実行時統計(アクセス・プランの演算子に関するランタイム統計) がどちらも含まれます。

ワークロード管理履歴分析ツール

`wlmhist.pl` および `wlmhistrep.pl` の Perl スクリプトは、アクティビティー・イベント・モニターによってキャプチャーされた情報を使用して、履歴分析を実行します。

セクション実行時統計

アクティビティー・イベント・モニターのもう 1 つの用途は、セクション実行時統計をキャプチャーすることです。このデータを使用すると、アクティビティー・イベント・モニターでキャプチャーされた実際の値と、アクセス・プラン内の見積もりコストとを比較することができます。この比較によって、アクセス・プランがまだ有効かどうかを確認できます。

アクティビティー・イベント・モニターの作成

アクティビティー・イベントをキャプチャーするイベント・モニターを作成するには、CREATE EVENT MONITOR FOR ACTIVITIES ステートメントを使用します。

始める前に

アクティビティー・イベント・モニターを作成するには、DBADM 権限か SQLADM 権限を持っている必要があります。

手順

アクティビティー・イベント・モニターを作成するには、以下のようにします。

1. CREATE EVENT MONITOR FOR ACTIVITIES ステートメントを作成します。例えば、myactevmon というイベント・モニターを作成するために、次のようなステートメントを使用できます。

   ```
   CREATE EVENT MONITOR myactevmon FOR ACTIVITIES
       WRITE TO TABLE
   ```

2. ステートメントを実行します。 この例では、以下のような特性を持つ、myactevmon という名のアクティビティー・イベント・モニターが作成されます。
   - アクティビティー・イベント・モニターに適用可能な、すべての論理データ・グループのモニター・エレメントが収集されます。
   - 出力は、リレーショナル表に書き込まれます。それぞれの表に、デフォルトの表名が割り当てられます。
   - イベント・モニターは、データベースが最初に活動化されたときに自動的に開始するよう構成されます。
   -

   このようになるのは、アクティビティー・イベント・モニターのデフォルトの設定を使用したからです。必要に応じて、これらのデフォルトをオーバーライドすることができます。

3. イベント・モニターを活動化します。 イベント・モニターは自動的に開始するよう構成されますが、そのイベント・モニターの作成後、データベースが初めて活動化されるまで、自動的に開始することはありません。イベント・モニターがデータの収集をただちに始めるように強制するには、次の例に示すような SET EVENT MONITOR STATE ステートメントを使用します。

   ```
   SET EVENT MONITOR myactevmon STATE 1
   ```

次のタスク

必要なデータを収集するようにイベント・モニターを構成します。

アクティビティー・モニターのデータ収集の構成

アクティビティーに関連したイベント・データを収集できるようにするには、その前にデータ収集を構成する必要があります。使用できる構成オプションは、データを収集する目的に応じて異なります。

このタスクについて

アクティビティー・イベント・モニターのデータ収集を構成するには、以下の 2 つの方法があります。

- CREATE THRESHOLD ステートメントを使用して作成したしきい値に違反すると、必ずアクティビティー・イベント・データが生成されるようにすることができます。
- WLM オブジェクトに関する CREATE または ALTER ステートメントに、COLLECT ACTIVITY METRICS 節を指定することができます。

手順

アクティビティーに関連したイベント・データを収集するには、以下のようにします。

1. データを収集する対象のアクティビティーを決定します。 次の表に、オプションの概略を示します。

どのアクティビティーに関するデータを収集するか。	構成を制御する要素
しきい値違反に関連したアクティビティー	しきい値違反に関連したアクティビティーの場合は、 COLLECT ACTIVITY DATA 節を指定して CREATE THRESHOLD または ALTER THRESHOLD ステートメントを実行します。 COLLECT ACTIVITY DATA 節を指定すると、しきい値が違反された場合には必ず、アクティビティー・データがアクティブなアクティビティー・イベント・モニターに送信されます。オプションで、WITH DETAILS、WITH DETAILS、SECTION、または AND VALUES 節のいずれかを追加で指定します。 **ヒント: `mon_act_metrics`** 構成パラメーターの値は `NONE` に変更することを検討してください。そうすると、イベント・データを収集する対象のアクティビティーが、しきい値違反に関連するアクティビティーだけに制限されます。

どのアクティビティーに関するデータを収集するか。	構成を制御する要素
特定のワークロード管理オブジェクトに関連するアクティビティー	ワークロード管理に関連するアクティビティーの場合は、COLLECT ACTIVITY DATA 節を指定して以下のステートメントのいずれかを実行します。オプションで、WITH DETAILS、WITH DETAILS、SECTION、WITH SECTION INCLUDE ACTUALS BASE、または AND VALUES 節のいずれかを追加で指定します。 • CREATE WORKLOAD • ALTER WORKLOAD • CREATE WORK ACTION SET • ALTER WORK ACTION SET • CREATE SERVICE CLASS • ALTER SERVICE CLASS 上記の各ステートメントに COLLECT ACTIVITY DATA 節を指定すると、そのステートメントで参照されるワークロード・オブジェクトのアクティビティー・データが、アクティブなアクティビティー・イベント・モニターに送信されます。 例えば、PAYROLL ワークロードに関するアクティビティー・イベント・データを収集するように、次のステートメントを実行してワークロードを変更することができます。 `ALTER WORKLOAD PAYROLL COLLECT ACTIVITY DATA WITH SECTION INCLUDE ACTUALS BASE` この例では、セクション実行時統計のデータとともにアクティビティー・データが収集されます。

2. 前のステップで説明した構成オプションのいずれかを使用して、収集レベルを設定します。 例えば、PAYROLL ワークロードに関するアクティビティー・イベント・データを収集するためには、次のステートメントを使用してワークロードを変更することができます。

   ```
   ALTER WORKLOAD PAYROLL COLLECT ACTIVITY DATA WITH SECTION INCLUDE ACTUALS BASE
   ```

   この最後の例では、セクション実行時統計のデータとともにアクティビティー・データが収集されます。

タスクの結果

データ収集が構成されます。

デフォルト・ユーザー・ワークロードのアクティビティー・データの収集

特定のワークロードにマップされないアクティビティーのアクティビティー・イベント・データを収集する場合は、次のようなステートメントを実行して、デフォルト・ユーザー・ワークロードのデータを収集できます。

```
ALTER WORKLOAD SYSDEFAULTUSERWORKLOAD COLLECT ACTIVITY DATA ON COORDINATOR
 WITH DETAILS
```

このステートメントにより、ユーザー定義のワークロード・オブジェクトに関連しないアクティビティーの詳細なアクティビティー・データが収集されます。デフォルトでは、ユーザー定義のワークロード・オブジェクトがない場合、すべてのデータベース接続が SYSDEFAULTUSERWORKLOAD ワークロードに関連付けられます。

次のタスク

アプリケーションまたはワークロードを実行します。収集の基準を満たすアクティビティーが実行されると、イベント・データはアクティブな任意のアクティビティー・イベント・モニターに送信されます。

アクティビティー・イベント・モニターによって生成されるデータ

アクティビティー・イベント・モニターの出力は、通常の表、ファイル、または名前付きパイプに書き込むか選択できます。

選択した出力形式にかかわらず、アクティビティー・イベント・モニターによってキャプチャーされる全データは、次の 4 つの論理データ・グループのいずれかから取得されます。

- event_activity
- event_activitymetrics
- event_activitystmt
- event_activityvals

統計イベント・モニターのデータを通常の表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

アクティビティー・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、アクティビティー・イベント・モニターによって書き込まれる情報。

アクティビティー・イベント・モニターの出力タイプとして WRITE TO TABLE を選択した場合、デフォルトでは、5 つの表が生成されます。各表には、1 つ以上の論理データ・グループのモニター・エレメントが入っています。

表 47. ACTIVITIES 表書き込みイベント・モニターによって生成される表: 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
ACTIVITY_*evmon-name*	event_activity

表 47. ACTIVITIES 表書き込みイベント・モニターによって生成される表 (続き): 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
ACTIVITYSTMT_*evmon-name*	event_activitystmt
ACTIVITYVALS_*evmon-name*	event_activityvals
ACTIVITYMETRICS_*evmon-name*	event_activitymetrics
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 48. アクティビティー・イベント・モニターに戻される情報 : デフォルトの表名: ACTIVITY_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACT_EXEC_TIME	BIGINT	act_exec_time アクティビティー実行時間
ACTIVATE_TIMESTAMP	TIMESTAMP	activate_timestamp タイム・スタンプの活動化
ACTIVITY_ID	BIGINT	activity_id アクティビティー ID
ACTIVITY_SECONDARY_ID	SMALLINT	activity_secondary_id アクティビティー 2 次 ID
ACTIVITY_TYPE	VARCHAR(64)	activity_type アクティビティー・タイプ
ADDRESS	VARCHAR(128)	address - 接続の開始元となった IP アドレス
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID)
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
APPL_NAME	VARCHAR(255)	appl_name アプリケーション名
ARM_CORRELATOR	BLOB(0)	arm_correlator アプリケーション応答測定相関関係子
COORD_PARTITION_NUM	INTEGER	coord_partition_num コーディネーター・パーティション番号
DB_WORK_ACTION_SET_ID	INTEGER	db_work_action_set_id データベース作業アクション・セット ID
DB_WORK_CLASS_ID	INTEGER	db_work_class_id データベース作業クラス ID
DETAILS_XML	BLOB(0)	
MON_INTERVAL_ID	BIGINT	mon_interval_id - モニター間隔 ID
NUM_REMAPS	BIGINT	num_remaps 再マップ数
PARENT_ACTIVITY_ID	BIGINT	parent_activity_id 親アクティビティー ID

表 48. アクティビティー・イベント・モニターに戻される情報 : デフォルトの表名: *ACTIVITY_evmon-name (続き)*

列名	データ・タイプ	説明
PARENT_UOW_ID	INTEGER	parent_uow_id 親作業単位 ID
PARTIAL_RECORD	SMALLINT	partial_record 部分レコード
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
PREP_TIME	BIGINT	prep_time 準備時間
QUERY_ACTUAL_DEGREE	INTEGER	query_actual_degree - パーティション内並列処理の実際の実行時の多重度
QUERY_CARD_ESTIMATE	BIGINT	query_card_estimate 照会行数の見積もり
QUERY_COST_ESTIMATE	BIGINT	query_cost_estimate 照会コストの見積もり
QUERY_DATA_TAG_LIST	VARCHAR(32)	query_data_tag_list - 照会データ・タグ・リスト
ROWS_FETCHED	BIGINT	rows_fetched フェッチ行数
ROWS_MODIFIED	BIGINT	rows_modified 変更行数
ROWS_RETURNED	BIGINT	rows_returned 戻り行数
SC_WORK_ACTION_SET_ID	INTEGER	sc_work_action_set_id サービス・クラス作業アクション・セット ID
SC_WORK_CLASS_ID	INTEGER	sc_work_class_id サービス・クラス作業クラス ID
SECTION_ACTUALS	BLOB(0)	section_actuals - セクション実行時統計
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name サービス・サブクラス名
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name サービス・スーパークラス名

表 48. アクティビティー・イベント・モニターに戻される情報 : デフォルトの表名: *ACTIVITY_evmon-name (続き)*

列名	データ・タイプ	説明
SESSION_AUTH_ID	VARCHAR(128)	session_auth_id セッション許可 ID
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
SQLCABC	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLCAID	CHARACTER(8)	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLCODE	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD1	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD2	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD3	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD4	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD5	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD6	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRM	VARCHAR(72)	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRP	CHARACTER(8)	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLSTATE	CHARACTER(5)	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLWARN	CHARACTER(11)	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SYSTEM_CPU_TIME	BIGINT	system_cpu_time システム CPU 時間
TIME_COMPLETED	TIMESTAMP	time_completed 完了時刻
TIME_CREATED	TIMESTAMP	time_created 作成時刻
TIME_STARTED	TIMESTAMP	time_started 開始時刻
TOTAL_SORT_TIME	BIGINT	total_sort_time ソート時間合計
TOTAL_SORTS	BIGINT	total_sorts ソート合計
TOTAL_STATS_FABRICATION_TIME	BIGINT	total_stats_fabrication_time - 統計作成の合計時間
TOTAL_STATS_FABRICATIONS	BIGINT	total_stats_fabrications - 統計作成の合計回数
TOTAL_SYNC_RUNSTATS	BIGINT	total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数
TOTAL_SYNC_RUNSTATS_TIME	BIGINT	total_sync_runstats_time - 同期 RUNSTATS の合計時間
TPMON_ACC_STR	VARCHAR(200)	tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング

表 48. アクティビティー・イベント・モニターに戻される情報 : デフォルトの表名: *ACTIVITY_evmon-name (続き)*

列名	データ・タイプ	説明
TPMON_CLIENT_APP	VARCHAR(255)	tpmon_client_app TP モニター・クライアント・アプリケーション名
TPMON_CLIENT_USERID	VARCHAR(255)	tpmon_client_userid TP モニター・クライアント・ユーザー ID
TPMON_CLIENT_WKSTN	VARCHAR(255)	tpmon_client_wkstn TP モニター・クライアント・ワークステーション名
UOW_ID	INTEGER	uow_id 作業単位 ID
USER_CPU_TIME	BIGINT	user_cpu_time ユーザー CPU 時間
WL_WORK_ACTION_SET_ID	INTEGER	wl_work_action_set_id - ワークロード作業アクション・セット ID
WL_WORK_CLASS_ID	INTEGER	wl_work_class_id - ワークロード作業クラス ID
WORKLOAD_ID	INTEGER	workload_id ワークロード ID
WORKLOAD_OCCURRENCE_ID	INTEGER	workload_occurrence_id ワークロード・オカレンス ID

表 49. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYSTMT_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACTIVATE_TIMESTAMP	TIMESTAMP	activate_timestamp タイム・スタンプの活動化
ACTIVITY_ID	BIGINT	activity_id アクティビティー ID
ACTIVITY_SECONDARY_ID	SMALLINT	activity_secondary_id アクティビティー 2 次 ID
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
COMP_ENV_DESC	BLOB(0)	comp_env_desc コンパイル環境
CREATOR	VARCHAR(128)	creator アプリケーション作成者
EFF_STMT_TEXT	CLOB	eff_stmt_text - 有効なステートメント・テキスト
EXECUTABLE_ID	VARCHAR(32)	executable_id 実行可能 ID
NUM_ROUTINES	BIGINT	num_routines - ルーチンの数
PACKAGE_NAME	VARCHAR(128)	package_name パッケージ名
PACKAGE_VERSION_ID	VARCHAR(64)	package_version_id パッケージ・バージョン
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
ROUTINE_ID	BIGINT	routine_id - ルーチン ID
SECTION_ENV	BLOB(0)	section_env セクション環境
SECTION_NUMBER	BIGINT	section_number セクション番号
STMT_FIRST_USE_TIME	TIMESTAMP	stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ
STMT_INVOCATION_ID	BIGINT	stmt_invocation_id ステートメント呼び出し ID
STMT_ISOLATION	BIGINT	stmt_isolation ステートメント分離
STMT_LAST_USE_TIME	TIMESTAMP	stmt_last_use_time - ステートメント最終使用時タイム・スタンプ

表 49. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYSTMT_evmon-name (続き)*

列名	データ・タイプ	説明
STMT_LOCK_TIMEOUT	INTEGER	stmt_lock_timeout ステートメント・ロック・タイムアウト
STMT_NEST_LEVEL	BIGINT	stmt_nest_level ステートメント・ネスト・レベル
STMT_PKGCACHE_ID	BIGINT	stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID
STMT_QUERY_ID	BIGINT	stmt_query_id ステートメント照会 ID
STMT_SOURCE_ID	BIGINT	stmt_source_id ステートメント・ソース ID
STMT_TEXT	CLOB	stmt_text SQL ステートメント・テキスト
STMT_TYPE	BIGINT	stmt_type ステートメント・タイプ
STMTNO	INTEGER	stmtno - ステートメント番号モニター・エレメント
UOW_ID	INTEGER	uow_id 作業単位 ID

表 50. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYMETRICS_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACTIVITY_ID	BIGINT	activity_id アクティビティー ID
ACTIVITY_SECONDARY_ID	SMALLINT	activity_secondary_id アクティビティー 2 次 ID
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
UOW_ID	INTEGER	uow_id 作業単位 ID
WLM_QUEUE_TIME_TOTAL	BIGINT	wlm_queue_time_total - ワークロード・マネージャー合計キュー時間
WLM_QUEUE_ASSIGNMENTS_TOTAL	BIGINT	wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て
FCM_TQ_RECV_WAIT_TIME	BIGINT	fcm_tq_recv_wait_time - FCM 表キュー受信待機時間
FCM_MESSAGE_RECV_WAIT_TIME	BIGINT	fcm_message_recv_wait_time - FCM メッセージの受信待機時間
FCM_TQ_SEND_WAIT_TIME	BIGINT	fcm_tq_send_wait_time - FCM 表キュー送信待機時間
FCM_MESSAGE_SEND_WAIT_TIME	BIGINT	fcm_message_send_wait_time - FCM メッセージの送信待機時間
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCK_WAITS	BIGINT	lock_waits ロック待機数
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求
LOG_BUFFER_WAIT_TIME	BIGINT	log_buffer_wait_time - ログ・バッファー待機時間
NUM_LOG_BUFFER_FULL	BIGINT	num_log_buffer_full フル・ログ・バッファーの回数

表 50. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
LOG_DISK_WAIT_TIME	BIGINT	log_disk_wait_time - ログ・ディスク待機時間
LOG_DISK_WAITS_TOTAL	BIGINT	log_disk_waits_total - ログ・ディスク待機の合計
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計
AUDIT_FILE_WRITE_WAIT_TIME	BIGINT	audit_file_write_wait_time - 監査ファイル書き込み待機時間
AUDIT_FILE_WRITES_TOTAL	BIGINT	audit_file_writes_total - 書き込まれた監査ファイルの合計
AUDIT_SUBSYSTEM_WAIT_TIME	BIGINT	audit_subsystem_wait_time - 監査サブシステム待機時間
AUDIT_SUBSYSTEM_WAITS_TOTAL	BIGINT	audit_subsystem_waits_total - 監査サブシステム待機の合計
DIAGLOG_WRITE_WAIT_TIME	BIGINT	diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間
DIAGLOG_WRITES_TOTAL	BIGINT	diaglog_writes_total - 診断ログ・ファイル書き込みの合計
FCM_SEND_WAIT_TIME	BIGINT	fcm_send_wait_time - FCM 送信待機時間
FCM_RECV_WAIT_TIME	BIGINT	fcm_recv_wait_time - FCM 受信待機時間
TOTAL_ACT_WAIT_TIME	BIGINT	total_act_wait_time - 合計アクティビティー待機時間
TOTAL_SECTION_SORT_PROC_TIME	BIGINT	total_section_sort_proc_time - セクションのソート処理時間の合計
TOTAL_SECTION_SORTS	BIGINT	total_section_sorts - セクションのソートの合計
TOTAL_SECTION_SORT_TIME	BIGINT	total_section_sort_time - セクションのソート時間の合計
TOTAL_ACT_TIME	BIGINT	total_act_time - 合計アクティビティー時間
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_MODIFIED	BIGINT	rows_modified 変更行数
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間

表 50. アクティビティー・イベント・モニターに戻される情報: 表名: ACTIVITYMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
ROWS_RETURNED	BIGINT	rows_returned 戻り行数
DEADLOCKS	BIGINT	deadlocks デッドロック検出数
LOCK_TIMEOUTS	BIGINT	lock_timeouts ロック・タイムアウト数
LOCK_ESCALS	BIGINT	lock_escals ロック・エスカレーション数
FCM_SENDS_TOTAL	BIGINT	fcm_sends_total - FCM 送信の合計
FCM_RECVS_TOTAL	BIGINT	fcm_recvs_total - FCM 受信の合計
FCM_SEND_VOLUME	BIGINT	fcm_send_volume - FCM 送信ボリューム
FCM_RECV_VOLUME	BIGINT	fcm_recv_volume - FCM 受信ボリューム
FCM_MESSAGE_SENDS_TOTAL	BIGINT	fcm_message_sends_total - FCM メッセージ送信の合計
FCM_MESSAGE_RECVS_TOTAL	BIGINT	fcm_message_recvs_total - FCM メッセージ受信の合計
FCM_MESSAGE_SEND_VOLUME	BIGINT	fcm_message_send_volume - FCM メッセージ送信ボリューム
FCM_MESSAGE_RECV_VOLUME	BIGINT	fcm_message_recv_volume - FCM メッセージ受信ボリューム
FCM_TQ_SENDS_TOTAL	BIGINT	fcm_tq_sends_total - FCM 表キュー送信の合計
FCM_TQ_RECVS_TOTAL	BIGINT	fcm_tq_recvs_total - FCM 表キュー受信の合計
FCM_TQ_SEND_VOLUME	BIGINT	fcm_tq_send_volume - FCM 表キュー送信ボリューム
FCM_TQ_RECV_VOLUME	BIGINT	fcm_tq_recv_volume - FCM 表キュー受信ボリューム

表 50. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
TQ_TOT_SEND_SPILLS	BIGINT	tq_tot_send_spills オーバーフローした表キュー・バッファーの合計数
POST_THRESHOLD_SORTS	BIGINT	post_threshold_sorts ポストしきい値ソート
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts ポスト共有しきい値ソート
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
AUDIT_EVENTS_TOTAL	BIGINT	audit_events_total - 監査イベントの合計
TOTAL_SORTS	BIGINT	total_sorts ソート合計
STMT_EXEC_TIME	BIGINT	stmt_exec_time - ステートメント実行時間
COORD_STMT_EXEC_TIME	BIGINT	coord_stmt_exec_time - コーディネーター・エージェントのステートメントの実行時間
TOTAL_ROUTINE_NON_SECT_PROC_TIME	BIGINT	total_routine_non_sect_proc_time - 非セクション処理時間
TOTAL_ROUTINE_NON_SECT_TIME	BIGINT	total_routine_non_sect_time - 非セクション・ルーチンの実行時間
TOTAL_SECTION_PROC_TIME	BIGINT	total_section_proc_time - セクション処理時間の合計
TOTAL_APP_SECTION_EXECUTIONS	BIGINT	total_app_section_executions - アプリケーションのセクション実行数の合計
TOTAL_SECTION_TIME	BIGINT	total_section_time - 合計セクション時間
TOTAL_ROUTINE_USER_CODE_PROC_TIME	BIGINT	total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計
TOTAL_ROUTINE_USER_CODE_TIME	BIGINT	total_routine_user_code_time - ルーチンのユーザー・コード時間の合計
TOTAL_ROUTINE_TIME	BIGINT	total_routine_time - 合計ルーチン時間
THRESH_VIOLATIONS	BIGINT	thresh_violations - しきい値違反の回数
NUM_LW_THRESH_EXCEEDED	BIGINT	num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数
TOTAL_ROUTINE_INVOCATIONS	BIGINT	total_routine_invocations - ルーチンの合計呼び出し数
LOCK_WAIT_TIME_GLOBAL	BIGINT	lock_wait_time_global - グローバル・ロック待機時間
LOCK_WAITS_GLOBAL	BIGINT	lock_waits_global - グローバル・ロック待機
RECLAIM_WAIT_TIME	BIGINT	reclaim_wait_time - 再利用待機時間
SPACEMAPPAGE_RECLAIM_WAIT_TIME	BIGINT	spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間
LOCK_TIMEOUTS_GLOBAL	BIGINT	lock_timeouts_global - グローバル・ロック・タイムアウト
LOCK_ESCALS_MAXLOCKS	BIGINT	lock_escals_maxlocks - maxlocks ロック・エスカレーション数
LOCK_ESCALS_LOCKLIST	BIGINT	lock_escals_locklist - locklist ロック・エスカレーション数
LOCK_ESCALS_GLOBAL	BIGINT	lock_escals_global - グローバル・ロック・エスカレーション数

表 50. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
CF_WAIT_TIME	BIGINT	cf_wait_time - クラスター・キャッシング機能待機時間
CF_WAITS	BIGINT	cf_waits - クラスター・キャッシング機能 DB2 pureScale サーバーの待機回数
POOL_DATA_GBP_L_READS	BIGINT	pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り
POOL_DATA_GBP_P_READS	BIGINT	pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り
POOL_DATA_LBP_PAGES_FOUND	BIGINT	pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ
POOL_DATA_GBP_INVALID_PAGES	BIGINT	pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ
POOL_INDEX_GBP_L_READS	BIGINT	pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り
POOL_INDEX_GBP_P_READS	BIGINT	pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り
POOL_INDEX_LBP_PAGES_FOUND	BIGINT	pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ
POOL_INDEX_GBP_INVALID_PAGES	BIGINT	pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ
POOL_XDA_GBP_L_READS	BIGINT	pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求
POOL_XDA_GBP_P_READS	BIGINT	pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求
POOL_XDA_LBP_PAGES_FOUND	BIGINT	pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ
POOL_XDA_GBP_INVALID_PAGES	BIGINT	pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ
EVMON_WAIT_TIME	BIGINT	evmon_wait_time - イベント・モニターの待機時間
EVMON_WAITS_TOTAL	BIGINT	evmon_waits_total - イベント・モニター合計待機回数
TOTAL_EXTENDED_LATCH_WAIT_TIME	BIGINT	total_extended_latch_wait_time - 拡張ラッチの合計待機時間
TOTAL_EXTENDED_LATCH_WAITS	BIGINT	total_extended_latch_waits - 拡張ラッチの合計待機回数
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間
POOL_QUEUED_ASYNC_DATA_REQS	BIGINT	pool_queued_async_data_reqs - データ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX_REQS	BIGINT	pool_queued_async_index_reqs - 索引プリフェッチ要求
POOL_QUEUED_ASYNC_XDA_REQS	BIGINT	pool_queued_async_xda_reqs - XDA プリフェッチ要求

表 50. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYMETRICS_evmon-name* (続き)

列名	データ・タイプ	説明
POOL_QUEUED_ASYNC_TEMP_DATA_REQS	BIGINT	pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_INDEX_REQS	BIGINT	pool_queued_async_temp_index_reqs - TEMPORARY 表スペースの索引プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_XDA_REQS	BIGINT	pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求
POOL_QUEUED_ASYNC_OTHER_REQS	BIGINT	pool_queued_async_other_reqs - 非プリフェッチの要求
POOL_QUEUED_ASYNC_DATA_PAGES	BIGINT	pool_queued_async_data_pages - データ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_INDEX_PAGES	BIGINT	pool_queued_async_index_pages - 索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_XDA_PAGES	BIGINT	pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_DATA_PAGES	BIGINT	pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES	BIGINT	pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求
POOL_QUEUED_ASYNC_TEMP_XDA_PAGES	BIGINT	pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求
POOL_FAILED_ASYNC_DATA_REQS	BIGINT	pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求
POOL_FAILED_ASYNC_INDEX_REQS	BIGINT	pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求
POOL_FAILED_ASYNC_XDA_REQS	BIGINT	pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_DATA_REQS	BIGINT	pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_INDEX_REQS	BIGINT	pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求
POOL_FAILED_ASYNC_TEMP_XDA_REQS	BIGINT	pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求
POOL_FAILED_ASYNC_OTHER_REQS	BIGINT	pool_failed_async_other_reqs - 失敗した非プリフェッチの要求
TOTAL_PEDS	BIGINT	total_peds - partial early distinct の合計回数
DISABLED_PEDS	BIGINT	1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』
POST_THRESHOLD_PEDS	BIGINT	post_threshold_peds - partial early distinct しきい値
TOTAL_PEAS	BIGINT	total_peas - partial early aggregation の合計回数
POST_THRESHOLD_PEAS	BIGINT	post_threshold_peas - partial early aggregation しきい値

表 50. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
TQ_SORT_HEAP_REQUESTS	BIGINT	tq_sort_heap_requests - 表キュー・ソート・ヒープ要求
TQ_SORT_HEAP_REJECTIONS	BIGINT	tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否
PREFETCH_WAIT_TIME	BIGINT	prefetch_wait_time プリフェッチ待ち時間
PREFETCH_WAITS	BIGINT	prefetch_waits - プリフェッチャーの待機カウント
POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP	BIGINT	1288 ページの『pool_data_gbp_indep_pages_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP	BIGINT	1326 ページの『pool_index_gbp_indep_pages_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP	BIGINT	1393 ページの『pool_xda_gbp_indep_pages_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』

表 51. アクティビティー・イベント・モニターに戻される情報: 表名: *ACTIVITYVALS_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACTIVATE_TIMESTAMP	TIMESTAMP	activate_timestamp タイム・スタンプの活動化
ACTIVITY_ID	BIGINT	activity_id アクティビティー ID
ACTIVITY_SECONDARY_ID	SMALLINT	activity_secondary_id アクティビティー 2 次 ID
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
STMT_VALUE_DATA	CLOB	stmt_value_data 値データ
STMT_VALUE_INDEX	INTEGER	stmt_value_index 値索引
STMT_VALUE_ISNULL	INTEGER	stmt_value_isnull NULL 値の値
STMT_VALUE_ISREOPT	INTEGER	stmt_value_isreopt ステートメント再最適化に使用される変数
STMT_VALUE_TYPE	CHARACTER(16)	stmt_value_type 値タイプ
UOW_ID	INTEGER	uow_id 作業単位 ID

表 52. アクティビティー・イベント・モニターに戻される情報 : デフォルトの表名: *CONTROL_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ

表 52. アクティビティー・イベント・モニターに戻される情報 : デフォルトの表名: *CONTROL_evmon-name (続き)*

列名	データ・タイプ	説明
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号

アクティビティー・イベント・モニターによって表に書き込まれたイベント情報へのアクセス

アクティビティー・イベント・モニターは、その出力を表、ファイル、およびパイプに書き込むことができます。

ファイルおよびパイプに書き込まれたデータの使用について詳しくは、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

始める前に

アクティビティー・イベント・モニターを作成し、活動化して、データ収集を有効にしておく必要があります。

このタスクについて

アクティビティー・イベント・モニターが生成する表については、 134 ページの『ターゲット表、コントロール表、およびイベント・モニター表の管理』に記載しています。 DB2 バージョン 10.1 より前は、メトリック・モニター・エレメントは ACTIVITIES 表の DETAIL_XML 列の XML 文書に書き込まれていました。この XML 文書のスキーマは、 sqllib/misc/DB2MonCommon.xsd ファイルにあります。また、スキーマの最上位エレメントは activity_metrics です。以前は details_xml XML 文書からしか使用できなかったメトリックは、バージョン 10.1 以降、アクティビティー・イベント・モニターにより生成された ACTIVITYMETRICS 表でも使用できるようになりました。

手順

アクティビティー・イベント・モニターにより生成されたデータにアクセスするには、以下のようにします。

1. 調べようとする列を返す照会を作成します。 例えば、特定の作業単位に関連するステートメントのテキストに関する情報に関心がある場合、次のような照会を作成することができます。

   ```
   SELECT UOW_ID, SUBSTR(STMT_TEXT, 1,70) AS STMT_TEXT FROM ACTIVITYSTMT_ACTEVMON
     WHERE UOW_ID=11
   ```

   この場合、イベント・モニターの名前は actevmon です。
2. この照会を実行します。 前述の照会から、次のような結果が返されることがあります。

   ```
   UOW_ID    STMT_TEXT
   --------- ------------------------------------------------------------------
          11 select * from gosaleshr.employee_expense_detail order by expense_date

     1 record(s) selected.
   ```

タスクの結果

例

ACTIVITY 表の DETAILS_XML 列のデータにアクセスする場合、DB2 製品とともにこの目的で提供されているインターフェースをどれでも使用することができます。例えば、作業単位に関してアクティビティー・イベント・モニターにより収集されたメトリック情報を調べる場合、次のようなステートメントを使用できます。

```
SELECT  SUBSTR(B.METRIC_NAME, 1, 20) METRIC_NAME, B.VALUE
   FROM ACTIVITY_ACTEVMON AS A,
   TABLE(MON_FORMAT_XML_METRICS_BY_ROW(A.DETAILS_XML))AS B
   WHERE UOW_ID=23
   ORDER BY B.VALUE DESC
```

このステートメントは、UOW_ID が 23 の作業単位に関して収集されたアクティビティー・メトリックをすべて返します。

```
METRIC_NAME          VALUE
-------------------- --------------------
TOTAL_CPU_TIME                     140625
ROWS_READ                             977
TOTAL_ACT_TIME                        880
STMT_EXEC_TIME                        880
COORD_STMT_EXEC_TIME                  880
TOTAL_SECTION_PROC_T                  880
TOTAL_SECTION_TIME                    880

⋮

⋮
FCM_TQ_SEND_WAITS_TO                    0
FCM_MESSAGE_SEND_WAI                    0
FCM_SEND_WAITS_TOTAL                    0
FCM_RECV_WAITS_TOTAL                    0

  92 record(s) selected.
```

イベント・モニターから返される XML データの処理に関する詳細は、20 ページの『モニター・データを XML 文書で返すインターフェース』を参照してください。

例: ステートメントの実行に関連したアクティビティー情報のキャプチャー

実行に長い時間を要しているステートメントを特定する場合は、あるしきい値を定義して、そのしきい値を超えた場合にそのステートメントの実行に関する情報をアクティビティー・イベント・モニターにキャプチャーさせることができます。

その後、ステートメントの実行情報を、アクティビティー・イベント・モニターによって収集された情報と関連付けて、スローダウンの原因となった可能性のある事柄について詳しく理解するのに役立つアクティビティー・メトリックを表示することができます。

始める前に

アクティビティー情報をキャプチャーする前に、問題のステートメントを特定する必要があります。例えば、ユーザーまたはアプリケーション開発者が、ある特定のステートメントの実行時間が想定より長いと訴えてくる場合もあるでしょう。または、パッケージ・キャッシュ・イベント・モニターを使用して、実行に時間がかか

っているステートメントを自分で特定する場合もあるでしょう。

このタスクについて

以下の例の場合、調査対象の照会は、アプリケーションの一部として実行されています。照会は以下のとおりです。

```
SELECT DISTINCT PARTS_BIN FROM STOCK WHERE PART_NUMBER = ?
```

スローダウンの理由として考えられることの 1 つとして、データ分布が好ましくない場合があります。例えば、STOCK 表にはほとんどの部品番号に関しては数行しか含まれておらず、特定の 1 つの部品番号については数千行にも及ぶので、この SELECT ステートメントの実行に時間がかかる場合があります。以下の例では、上記の照会と関連したアクティビティーにおいてパラメーター・マーカー (?) として処理された実際の値を取得する方法を示します。

手順

照会の実行が低下した原因がデータ分布が適切でないためであるという推測をテストするには、該当のステートメントにしきい値を作成できます。その後、このしきい値とアクティビティー・イベント・モニターを使用して、そのステートメントの実行に関する情報をキャプチャーできます。この情報から、想定よりも長く実行された照会によって処理された実際の値を判定できます。

1. 該当のステートメントのしきい値を作成します。その際、ステートメントが 10 秒を超えて実行されると、しきい値違反イベントが発生するように指定します。

   ```
   CREATE THRESHOLD TH1
       FOR STATEMENT TEXT 'SELECT DISTINCT PARTS_BIN
       FROM STOCK WHERE PART_NUMBER = ?' ACTIVITIES
       ENFORCEMENT DATABASE
       WHEN ACTIVITYTOTALTIME > 10 SECONDS
       COLLECT ACTIVITY DATA WITH DETAILS, SECTION AND VALUES
       CONTINUE
   ```

2. しきい値違反を記録するためのしきい値イベント・モニターを作成します。

   ```
   CREATE EVENT MONITOR STMT_THRESH_VIOLATIONS
     FOR THRESHOLD VIOLATIONS
     WRITE TO TABLE
     AUTOSTART
   ```

3. 詳細なアクティビティー情報を記録するためのアクティビティー・イベント・モニターを作成します。

   ```
   CREATE EVENT MONITOR ACTIVITIES
      FOR ACTIVITIES
      WRITE TO TABLE
   ```

4. 新しいイベント・モニターを使用可能にします。

   ```
   SET EVENT MONITOR ACTIVITIES STATE 1
   SET EVENT MONITOR STMT_THRESH_VIOLATIONS STATE 1
   ```

5. ステートメントを実行するアプリケーションを実行します。しきい値違反が生じると、しきい値違反についての情報がしきい値違反イベント・モニター STMT_THRESH_VIOLATIONS によって記録され、しきい値違反に関連したアクティビティーについての情報がアクティビティー・イベント・モニター ACTIVITIES によって記録されます。

6. しきい値違反が発生したかどうかを判別するには、ステップ 1 で定義したしきい値 TH1 のしきい値イベント・モニターにより記録された違反番号を照会しま

す。この照会を実行するには、ビュー SYSCAT.THRESHOLDS を、しきい値違反情報が含まれているしきい値イベント・モニターによって生成された表と結合してください。しきい値名 TH1 は SYSCAT.THRESHOLDS に保持されているため、この結合が必要です。

```
SELECT COUNT(1) NUM_VIOLATIONS
        FROM THRESHOLDVIOLATIONS_DB2THRESHOLDVIOLATIONS T
        JOIN SYSCAT.THRESHOLDS S ON T.THRESHOLDID = S.THRESHOLDID
        WHERE S.THRESHOLDNAME = 'TH1';

NUM_VIOLATIONS
--------------
             1

  1 record(s) selected.
```

この場合、しきい値違反が 1 回ありました。ステップ 1 (339 ページ) で指定したステートメントの実行時間が 10 秒を超えた実行が 1 回あったということです。

7. 1 (339 ページ) で指定したステートメントのパラメーター・マーカー (?) で表されていたデータ (部品番号) を調べます。 次の例では、SELECT ステートメントによって、アクティビティー・イベント・モニターが生成した ACTIVITYVALS 表の 1 つから、パラメーター・マーカーの値 (SQL 内の後ろの STMT_VALUE_DATA によって表される) が取得されます。

```
SELECT SUBSTR(V.STMT_VALUE_DATA, 1, 80) PARAM_MARKER_VALUE
        FROM ACTIVITYVALS_ACTIVITIES V
        JOIN THRESHOLDVIOLATIONS_STMT_THRESH_VIOLATIONS T
          ON T.APPL_ID = V.APPL_ID
          AND T.UOW_ID = V.UOW_ID
          AND T.ACTIVITY_ID = V.ACTIVITY_ID
        JOIN SYSCAT.THRESHOLDS S
          ON T.THRESHOLDID = S.THRESHOLDID
        WHERE S.THRESHOLDNAME = 'TH1';
```

前述の例の場合、select ステートメントによって、アクティビティー・イベント・モニターによって生成された表の 1 つから、パラメーター・マーカーの値 (STMT_VALUE_DATA) が取得されます。

```
PARAM_MARKER_VALUE
--------------------------------------------------------------------------------
475299
```

8. これで、長期間実行されていたステートメントに関連した PART_NUMBER の値が分かったので、STOCK 表を調べて、照会の時間を長くした原因となっている可能性のあるその部品番号が表に出現するかどうかを確認できます。例えば、PART_NUBMER の値が 475299 である行数が (その他の部品番号の行数と比較して) 多ければ、それが原因で、この値が使用された場合に照会の実行時間が長くなっている可能性があります。

変化形: 実行可能 ID を使用しているステートメントに対するしきい値の定義

前述の例の場合、しきい値は、ステップ 1 (339 ページ) でステートメントの実際のテキストを使用して明示的に識別されました。しきい値を間接的に定義することも可能で、その場合にはパッケージ・キャッシュに含まれるステートメントの実行可能 ID を指定します。例えば、しきい値を次のように定義できます。

```
CREATE THRESHOLD TH1
   FOR STATEMENT REFERENCE
     x'0100000000000000002000000000000000000000000020020100304162158584850' ACTIVITIES
   ENFORCEMENT DATABASE
   WHEN ACTIVITYTOTALTIME > 10 SECONDS
   COLLECT ACTIVITY DATA WITH DETAILS, SECTION AND VALUES
   CONTINUE;
```

この例では、STATEMENT REFERENCE というキーワードに続く実行可能 ID を使用して、パッケージ・キャッシュ内で対応するステートメント・テキストを検索しています。ステートメントの実行可能 ID は、パッケージ・キャッシュを調べることにより判別できます。ステートメントの実行可能 ID などの、パッケージ・キャッシュに入っている情報の調査方法について詳しくは、311 ページの『パッケージ・キャッシュ情報を使用して、パフォーマンス調整の対象候補となるステートメントを識別する』を参照してください。

パッケージ・キャッシュでその実行可能 ID が検出される場合、パッケージ・キャッシュから関連するステートメント・テキストが取得されて、ステートメントしきい値の定義に使用されます。静的 SQL セクションのステートメントの場合、実行可能 ID がパッケージ・キャッシュに存在しない場合、そのステートメント・テキストはシステム・カタログから取得されます。動的 SQL セクションのステートメントの場合、PREPARE ステートメントを使用して、そのステートメント・ストリングから準備済みステートメントを作成することを考慮してください。実行可能 ID がパッケージ・キャッシュにもシステム・カタログにも見つからない場合は、エラー (SQL4721N) が戻されます。

統計イベント・モニター

統計イベント・モニターは、システムの稼働状況をさまざまな側面から調べるために使用できるデータをキャプチャーします。

統計イベント・モニターを使用して、次の 2 つのタイプの情報を収集することができます。

メトリック
メトリックとは、システムに関する測定情報をキャプチャーする要求モニター・エレメントのことです。これらのメトリックには、ロックを待機した時間などの消費時間モニター・エレメントと、デッドロック発生回数などのカウンター・モニター・エレメントが含まれます。これらのメトリックの収集は、**mon_req_metrics** データベース構成パラメーターを使用してデータベース全体に対して構成することも、CREATE ステートメントまたは ALTER SERVICE CLASS ステートメントの COLLECT REQUEST METRICS 節を使用して特定のサービス・クラスに対して構成することもできます。一部のシステム・メトリックは、 EVENT_SCSTATS 論理データ・グループおよび EVENT_WLSTATS 論理データ・グループの **details_xml** モニター・エレメントおよび **metrics** モニター・エレメントの一部として収集されます。これらのモニター・エレメントは両方とも XML 文書です。

統計 統計は、ワークロード・マネージャー・オブジェクトに関して保持されます。ワークロード・マネージャー・オブジェクトには、サービス・クラス、作業クラス、ワークロード、およびしきい値キューが含まれます。これらの統計には、TEMPORARY 表スペースの使用量などの最高水準点モニター・エレメントと、見積もりコストの平均値などの計算されたモニター・エレメ

ントが含まれます。この統計はメモリー内に常駐しており、ワークロード・マネージャーの統計表関数を使用してリアルタイムで表示することが可能です。また、統計を収集して統計イベント・モニターに送信し、後で履歴分析を行うときに表示することもできます。デフォルトでは、各ワークロード管理オブジェクトについて、最小セットの統計が収集されます。統計収集の範囲は、さまざまなワークロード管理オブジェクトに対する CREATE または ALTER ステートメントで各節を使用して変更することができます。

details_xml モニター・エレメントおよび **metrics** モニター・エレメントの XML 文書には同じメトリックが入っていますが、重要な相違点が 1 つあります。**details_xml** のメトリックは、一般的には 0 から始まり、次にデータベースが活動化されるときまで累積し続けます。一方、 **metrics** のメトリックは、統計が最後に収集された時からのメトリックの値の変化を示すために計算されます。返される XML 文書のスキーマは、ファイル sqllib/misc/DB2MonCommon.xsd に用意されています。最上位エレメントは、 **system_metrics** です。

metrics モニター・エレメントの XML 文書でシステム・メトリックを確認できる他、EVENT_SCMETRICS 論理データ・グループおよび EVENT_WLMETRICS 論理データ・グループに関連する出力から直接、個々のメトリックを確認することもできます。

重要: バージョン 10.1 フィックスパック 1 以降、**details_xml** の XML 文書は統計イベント・モニターでは推奨されなくなりました。今後のリリースでは除去される可能性があります。**details_xml** に返される XML メトリック・データを使用している場合は、代わりに **metrics** 文書を使用してください。あるいは、イベント・モニターにより収集される情報に EVENT_SCMETRICS 論理データ・グループおよび EVENT_WLMETRICS 論理データ・グループを含めて、メトリック・モニター・エレメントに直接アクセスすることもできます。

統計イベント・モニターが収集するメトリックは、MON_GET_SERVICE_SUBCLASS_DETAILS および MON_GET_WORKLOAD_DETAILS 表関数がレポートするメトリック・セットと同じです これら 2 つの表関数から返される表の DETAILS 列内の XML 文書には、システム・メトリック・モニター・エレメントの他にも多数のモニター・エレメントが含まれています。

要求に関するシステム・メトリック・モニター・エレメントが収集されるのは、親のサービス・スーパークラスで要求モニター・エレメント収集が有効になっているサービス・サブクラス内のエージェントによってその要求が処理される場合、またはシステム・メトリック収集がデータベース全体に対して有効になっている場合のみです。システム・メトリック収集を、サービス・スーパークラスに関してだけでなくデータベース・レベルで無効にすると、 **details_xml** 文書内にレポートされるメトリックの増加はストップします (あるいはデータベース活動化の時点で要求メトリックが無効になっていた場合は 0 のままです)。

統計イベント・モニターによって生成されるデータ

統計イベント・モニターは、データベースの使用中に発生した統計イベントを記録します。統計イベント・モニターの定義では、データベースによりイベントが記録

される場所が指定されます。統計イベント・モニターの出力は、通常の表、ファイル、または名前付きパイプに書き込むか選択できます。

選択した出力形式にかかわらず、統計イベント・モニターによってキャプチャーされた全データは、次の論理データ・グループのいずれかに書き込まれます。また、統計イベント・モニターは、これらの論理データ・グループに関してキャプチャーしたデータを、表、ファイル、または名前付きパイプに書き込むことができます。

- EVENT_SCSTATS
- EVENT_WCSTATS
- EVENT_WLSTATS
- EVENT_QSTATS
- EVENT_HISTOGRAMBIN

統計イベント・モニターのデータを通常の表に書き込むように選択した場合は、以下の 3 つの論理データ・グループが追加で使用可能です。

- EVENT_SCMETRICS
- EVENT_WLMETRICS
- CONTROL 論理データ・グループ。イベント・モニター自体に関するメタデータを生成するために使用します。

統計イベント・モニター用に表に書き込まれる情報:

統計イベント・モニターの出力タイプとして WRITE TO TABLE を選択した場合、いくつかの表が生成されます。各表には、1 つ以上の論理データ・グループのモニター・エレメントが入っています。

次の表に、統計イベント・モニターで使用される論理データ・グループおよび関連表をリストします。各論理データ・グループのデフォルトの表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントによって作成したときにイベント・モニターに指定した名前が連結されて生成されます。ここに示している表名は、 CREATE EVENT MONITOR ステートメントの一部として表名が指定されなかった場合のデフォルトの表名です。

表 53. STATISTICS 表書き込みイベント・モニターによって生成される表

デフォルトの表名	レポートされる論理データ・グループ
QSTATS_*evmon-name*	79 ページの『event_qstats 論理データ・グループ』
SCSTATS_*evmon-name*	89 ページの『event_scstats 論理データ・グループ』
HISTOGRAMBIN_*evmon-name*	78 ページの『event_histogrambin 論理データ・グループ』
WCSTATS_*evmon-name*	97 ページの『event_wcstats 論理データ・グループ』
WLSTATS_*evmon-name*	108 ページの『event_wlstats 論理データ・グループ』
SCMETRICS_*evmon-name*	79 ページの『event_scmetrics 論理データ・グループ』
WLMETRICS_*evmon-name*	98 ページの『event_wlmetrics 論理データ・グループ』
CONTROL_*evmon-name*	CONTROL 論理グループは、 event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 54. 統計イベント・モニターに戻される情報 : デフォルトの表名: QSTATS_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
LAST_WLM_RESET	TIMESTAMP	last_wlm_reset 最後にリセットされた時刻
MON_INTERVAL_ID	BIGINT	mon_interval_id - モニター間隔 ID
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
QUEUE_ASSIGNMENTS_TOTAL	BIGINT	queue_assignments_total キュー割り当ての合計
QUEUE_SIZE_TOP	INTEGER	queue_size_top キュー・サイズの最上位
QUEUE_TIME_TOTAL	BIGINT	queue_time_total キュー時間の合計
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name サービス・サブクラス名
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name サービス・スーパークラス名
STATISTICS_TIMESTAMP	TIMESTAMP	statistics_timestamp 統計タイム・スタンプ
THRESHOLD_DOMAIN	VARCHAR(64)	threshold_domain しきい値ドメイン
THRESHOLD_NAME	VARCHAR(128)	threshold_name しきい値名
THRESHOLD_PREDICATE	VARCHAR(64)	threshold_predicate しきい値述部
THRESHOLDID	INTEGER	thresholdid しきい値 ID
WORK_ACTION_SET_NAME	VARCHAR(128)	work_action_set_name 作業アクション・セット名
WORK_CLASS_NAME	VARCHAR(128)	work_class_name 作業クラス名

表 55. 統計イベント・モニターに戻される情報: 表名: SCSTATS_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACT_CPU_TIME_TOP	BIGINT	act_cpu_time_top - アクティビティーの CPU 時間の最上位
ACT_REMAPPED_IN	BIGINT	act_remapped_in - 再マッピングするアクティビティー
ACT_REMAPPED_OUT	BIGINT	act_remapped_out - 再マッピングの際に除外されるアクティビティー
ACT_ROWS_READ_TOP	BIGINT	act_rows_read_top - アクティビティーの読み取り行数の最上位
ACT_THROUGHPUT	BIGINT	act_throughput - アクティビティー・スループット
AGG_TEMP_TABLESPACE_TOP	BIGINT	agg_temp_tablespace_top 集約 TEMPORARY 表スペースの最上位

表 55. 統計イベント・モニターに戻される情報: 表名: SCSTATS_evmon-name (続き)

列名	データ・タイプ	説明
APP_ACT_ABORTED_TOTAL	BIGINT	app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数
APP_ACT_COMPLETED_TOTAL	BIGINT	app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数
APP_ACT_REJECTED_TOTAL	BIGINT	app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数
CONCURRENT_ACT_TOP	INTEGER	concurrent_act_top 並行アクティビティーの最上位
CONCURRENT_CONNECTION_TOP	INTEGER	concurrent_connection_top 並行接続の最上位
CONCURRENT_WLO_TOP	INTEGER	concurrent_wlo_top 並行ワークロード・オカレンスの最上位
COORD_ACT_ABORTED_TOTAL	BIGINT	coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計
COORD_ACT_COMPLETED_TOTAL	BIGINT	coord_act_completed_total 完了したコーディネーター・アクティビティーの合計
COORD_ACT_EST_COST_AVG	BIGINT	coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト
COORD_ACT_EXEC_TIME_AVG	BIGINT	coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間
COORD_ACT_INTERARRIVAL_TIME_AVG	BIGINT	coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間
COORD_ACT_LIFETIME_AVG	BIGINT	coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均
COORD_ACT_LIFETIME_TOP	BIGINT	coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位
COORD_ACT_QUEUE_TIME_AVG	BIGINT	coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間
COORD_ACT_REJECTED_TOTAL	BIGINT	coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計
COST_ESTIMATE_TOP	BIGINT	cost_estimate_top コスト見積もりの最上位
CPU_UTILIZATION	BIGINT	cpu_utilization - CPU 使用率
DETAILS_XML	BLOB	details_xml - 詳細 XMLスーパークラス SYSDEFAULTSYSTEMCLASS の下のデフォルト・サブクラス SYSDEFAULTSUBCLASS に関して、この文書内に報告されるメトリックの値は 0 です。
LAST_WLM_RESET	TIMESTAMP	last_wlm_reset 最後にリセットされた時刻
MEMBER	SMALLINT	member - データベース・メンバー
METRICS	BLOB	metrics - メトリック
MON_INTERVAL_ID	BIGINT	mon_interval_id - モニター間隔 ID
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
REQUEST_EXEC_TIME_AVG	BIGINT	request_exec_time_avg 要求の平均実行時間
ROWS_RETURNED_TOP	BIGINT	rows_returned_top 実際の戻り行数の最上位
SERVICE_CLASS_ID	INTEGER	service_class_id サービス・クラス ID
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name サービス・サブクラス名

表 55. 統計イベント・モニターに戻される情報: 表名: SCSTATS_evmon-name (続き)

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name サービス・スーパークラス名
STATISTICS_TIMESTAMP	TIMESTAMP	statistics_timestamp 統計タイム・スタンプ
TEMP_TABLESPACE_TOP	BIGINT	temp_tablespace_top TEMPORARY 表スペースの最上位
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間
UOW_COMPLETED_TOTAL	BIGINT	uow_completed_total - 完了済みの合計作業単位
UOW_LIFETIME_AVG	BIGINT	uow_lifetime_avg - 作業単位の平均存続期間
UOW_THROUGHPUT	BIGINT	uow_throughput - 作業単位スループット
UOW_TOTAL_TIME_TOP	BIGINT	uow_total_time_top - UOW 合計時間の最上位

表 56. 統計イベント・モニターに戻される情報: 表名: HISTOGRAMBIN_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
BIN_ID	INTEGER	bin_id ヒストグラム・ビン ID
BOTTOM	BIGINT	bottom ヒストグラム・ビンの最下位
HISTOGRAM_TYPE	VARCHAR(64)	histogram_type ヒストグラム・タイプ
MON_INTERVAL_ID	BIGINT	mon_interval_id - モニター間隔 ID
NUMBER_IN_BIN	BIGINT	number_in_bin ビン内の数
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
SERVICE_CLASS_ID	INTEGER	service_class_id サービス・クラス ID
STATISTICS_TIMESTAMP	TIMESTAMP	statistics_timestamp 統計タイム・スタンプ
TOP	BIGINT	top ヒストグラム・ビンの最上位
WORK_ACTION_SET_ID	INTEGER	work_action_set_id 作業アクション・セット ID
WORK_CLASS_ID	INTEGER	work_class_id 作業クラス ID
WORKLOAD_ID	INTEGER	workload_id ワークロード ID

表 57. 統計イベント・モニターに戻される情報: 表名: WCSTATS_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACT_CPU_TIME_TOP	BIGINT	act_cpu_time_top - アクティビティーの CPU 時間の最上位
ACT_ROWS_READ_TOP	BIGINT	act_rows_read_top - アクティビティーの読み取り行数の最上位
ACT_TOTAL	BIGINT	act_total アクティビティーの合計
COORD_ACT_EST_COST_AVG	BIGINT	coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト

表 57. 統計イベント・モニターに戻される情報: 表名: WCSTATS_evmon-name (続き)

列名	データ・タイプ	説明
COORD_ACT_EXEC_TIME_AVG	BIGINT	coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間
COORD_ACT_INTERARRIVAL_TIME_AVG	BIGINT	coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間
COORD_ACT_LIFETIME_AVG	BIGINT	coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均
COORD_ACT_LIFETIME_TOP	BIGINT	coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位
COORD_ACT_QUEUE_TIME_AVG	BIGINT	coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間
COST_ESTIMATE_TOP	BIGINT	cost_estimate_top コスト見積もりの最上位
LAST_WLM_RESET	TIMESTAMP	last_wlm_reset 最後にリセットされた時刻
MON_INTERVAL_ID	BIGINT	mon_interval_id - モニター間隔 ID
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
ROWS_RETURNED_TOP	BIGINT	rows_returned_top 実際の戻り行数の最上位
STATISTICS_TIMESTAMP	TIMESTAMP	statistics_timestamp 統計タイム・スタンプ
TEMP_TABLESPACE_TOP	BIGINT	temp_tablespace_top TEMPORARY 表スペースの最上位
WORK_ACTION_SET_ID	INTEGER	work_action_set_id 作業アクション・セット ID
WORK_ACTION_SET_NAME	VARCHAR(128)	work_action_set_name 作業アクション・セット名
WORK_CLASS_ID	INTEGER	work_class_id 作業クラス ID
WORK_CLASS_NAME	VARCHAR(128)	work_class_name 作業クラス名

表 58. 統計イベント・モニターに戻される情報: 表名: WLSTATS_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACT_CPU_TIME_TOP	BIGINT	act_cpu_time_top - アクティビティーの CPU 時間の最上位
ACT_ROWS_READ_TOP	BIGINT	act_rows_read_top - アクティビティーの読み取り行数の最上位
ACT_THROUGHPUT	BIGINT	act_throughput - アクティビティー・スループット
APP_ACT_ABORTED_TOTAL	BIGINT	app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数
APP_ACT_COMPLETED_TOTAL	BIGINT	app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数
APP_ACT_REJECTED_TOTAL	BIGINT	app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数
CONCURRENT_WLO_ACT_TOP	INTEGER	concurrent_wlo_act_top 並行 WLO アクティビティーの最上位
CONCURRENT_WLO_TOP	INTEGER	concurrent_wlo_top 並行ワークロード・オカレンスの最上位

表 58. 統計イベント・モニターに戻される情報: 表名: *WLSTATS_evmon-name* (続き)

列名	データ・タイプ	説明
COORD_ACT_ABORTED_TOTAL	BIGINT	coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計
COORD_ACT_COMPLETED_TOTAL	BIGINT	coord_act_completed_total 完了したコーディネーター・アクティビティーの合計
COORD_ACT_EST_COST_AVG	BIGINT	coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト
COORD_ACT_EXEC_TIME_AVG	BIGINT	coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間
COORD_ACT_INTERARRIVAL_TIME_AVG	BIGINT	933 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』
COORD_ACT_LIFETIME_AVG	BIGINT	coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均
COORD_ACT_LIFETIME_TOP	BIGINT	coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位
COORD_ACT_QUEUE_TIME_AVG	BIGINT	coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間
COORD_ACT_REJECTED_TOTAL	BIGINT	coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計
COST_ESTIMATE_TOP	BIGINT	cost_estimate_top コスト見積もりの最上位
CPU_UTILIZATION	BIGINT	cpu_utilization - CPU 使用率
DETAILS_XML	BLOB	details_xml - 詳細 XML
LAST_WLM_RESET	TIMESTAMP	last_wlm_reset 最後にリセットされた時刻
LOCK_WAIT_TIME_GLOBAL_TOP	BIGINT	lock_wait_time_global_top - 最長グローバル・ロック待機時間
LOCK_WAIT_TIME_TOP	BIGINT	lock_wait_time_top - ロック待機時間の最上位
MEMBER	SMALLINT	member - データベース・メンバー
METRICS	BLOB	metrics - メトリック
MON_INTERVAL_ID	BIGINT	mon_interval_id - モニター間隔 ID
ROWS_RETURNED_TOP	BIGINT	rows_returned_top 実際の戻り行数の最上位
STATISTICS_TIMESTAMP	TIMESTAMP	statistics_timestamp 統計タイム・スタンプ
TEMP_TABLESPACE_TOP	BIGINT	temp_tablespace_top TEMPORARY 表スペースの最上位
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間
UOW_COMPLETED_TOTAL	BIGINT	uow_completed_total - 完了済みの合計作業単位
UOW_LIFETIME_AVG	BIGINT	uow_lifetime_avg - 作業単位の平均存続期間
UOW_THROUGHPUT	BIGINT	uow_throughput - 作業単位スループット
UOW_TOTAL_TIME_TOP	BIGINT	uow_total_time_top - UOW 合計時間の最上位
WLO_COMPLETED_TOTAL	BIGINT	wlo_completed_total 完了したワークロード・オカレンスの合計

表 58. 統計イベント・モニターに戻される情報: 表名: *WLSTATS_evmon-name (続き)*

列名	データ・タイプ	説明
WORKLOAD_ID	INTEGER	workload_id ワークロード ID
WORKLOAD_NAME	VARCHAR(128)	workload_name ワークロード名

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: *SCMETRICS_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
SERVICE_CLASS_ID	INTEGER	service_class_id サービス・クラス ID
SERVICE_SUBCLASS_NAME	VARCHAR	service_subclass_name サービス・サブクラス名
SERVICE_SUPERCLASS_NAME	VARCHAR	service_superclass_name サービス・スーパークラス名
STATISTICS_TIMESTAMP	TIMESTAMP	statistics_timestamp 統計タイム・スタンプ
WLM_QUEUE_TIME_TOTAL	BIGINT	1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』
WLM_QUEUE_ASSIGNMENTS_TOTAL	BIGINT	1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』
FCM_TQ_RECV_WAIT_TIME	BIGINT	1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』
FCM_MESSAGE_RECV_WAIT_TIME	BIGINT	1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』
FCM_TQ_SEND_WAIT_TIME	BIGINT	1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』
FCM_MESSAGE_SEND_WAIT_TIME	BIGINT	1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』
AGENT_WAIT_TIME	BIGINT	829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』
AGENT_WAITS_TOTAL	BIGINT	831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』
LOCK_WAIT_TIME	BIGINT	1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』
LOCK_WAITS	BIGINT	1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』
DIRECT_READ_TIME	BIGINT	989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』
DIRECT_READ_REQS	BIGINT	987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』
DIRECT_WRITE_TIME	BIGINT	996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
DIRECT_WRITE_REQS	BIGINT	994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』
LOG_BUFFER_WAIT_TIME	BIGINT	1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』
NUM_LOG_BUFFER_FULL	BIGINT	1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』
LOG_DISK_WAIT_TIME	BIGINT	1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』
LOG_DISK_WAITS_TOTAL	BIGINT	1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』
TCPIP_RECV_WAIT_TIME	BIGINT	1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』
TCPIP_RECVS_TOTAL	BIGINT	1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』
CLIENT_IDLE_WAIT_TIME	BIGINT	897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』
IPC_RECV_WAIT_TIME	BIGINT	1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』
IPC_RECVS_TOTAL	BIGINT	1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』
IPC_SEND_WAIT_TIME	BIGINT	1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』
IPC_SENDS_TOTAL	BIGINT	1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』
TCPIP_SEND_WAIT_TIME	BIGINT	1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』
TCPIP_SENDS_TOTAL	BIGINT	1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』
POOL_WRITE_TIME	BIGINT	1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』
POOL_READ_TIME	BIGINT	1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』
AUDIT_FILE_WRITE_WAIT_TIME	BIGINT	858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』
AUDIT_FILE_WRITES_TOTAL	BIGINT	860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』
AUDIT_SUBSYSTEM_WAIT_TIME	BIGINT	862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』
AUDIT_SUBSYSTEM_WAITS_TOTAL	BIGINT	864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
DIAGLOG_WRITE_WAIT_TIME	BIGINT	984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』
DIAGLOG_WRITES_TOTAL	BIGINT	986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』
FCM_SEND_WAIT_TIME	BIGINT	1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』
FCM_RECV_WAIT_TIME	BIGINT	1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』
TOTAL_WAIT_TIME	BIGINT	1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』
RQSTS_COMPLETED_TOTAL	BIGINT	1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』
TOTAL_RQST_TIME	BIGINT	1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』
APP_RQSTS_COMPLETED_TOTAL	BIGINT	841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』
TOTAL_APP_RQST_TIME	BIGINT	1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』
TOTAL_SECTION_SORT_PROC_TIME	BIGINT	1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』
TOTAL_SECTION_SORTS	BIGINT	1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』
TOTAL_SECTION_SORT_TIME	BIGINT	1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』
ROWS_READ	BIGINT	1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』
ROWS_MODIFIED	BIGINT	1464 ページの『rows_modified 変更行数 : モニター・エレメント』
POOL_DATA_L_READS	BIGINT	1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』
POOL_INDEX_L_READS	BIGINT	1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』
POOL_TEMP_DATA_L_READS	BIGINT	1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』
POOL_TEMP_INDEX_L_READS	BIGINT	1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
POOL_XDA_L_READS	BIGINT	1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』
POOL_TEMP_XDA_L_READS	BIGINT	1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』
TOTAL_CPU_TIME	BIGINT	1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』
ACT_COMPLETED_TOTAL	BIGINT	810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』
POOL_DATA_P_READS	BIGINT	1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』
POOL_TEMP_DATA_P_READS	BIGINT	1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』
POOL_XDA_P_READS	BIGINT	1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』
POOL_TEMP_XDA_P_READS	BIGINT	1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』
POOL_INDEX_P_READS	BIGINT	1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』
POOL_TEMP_INDEX_P_READS	BIGINT	1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』
POOL_DATA_WRITES	BIGINT	1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』
POOL_XDA_WRITES	BIGINT	1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』
POOL_INDEX_WRITES	BIGINT	1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』
DIRECT_READS	BIGINT	991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』
DIRECT_WRITES	BIGINT	998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』
ROWS_RETURNED	BIGINT	1468 ページの『rows_returned 戻り行数 : モニター・エレメント』
DEADLOCKS	BIGINT	978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
LOCK_TIMEOUTS	BIGINT	1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』
LOCK_ESCALS	BIGINT	1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』
FCM_SENDS_TOTAL	BIGINT	1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』
FCM_RECVS_TOTAL	BIGINT	1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』
FCM_SEND_VOLUME	BIGINT	1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』
FCM_RECV_VOLUME	BIGINT	1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』
FCM_MESSAGE_SENDS_TOTAL	BIGINT	1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』
FCM_MESSAGE_RECVS_TOTAL	BIGINT	1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』
FCM_MESSAGE_SEND_VOLUME	BIGINT	1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』
FCM_MESSAGE_RECV_VOLUME	BIGINT	1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』
FCM_TQ_SENDS_TOTAL	BIGINT	1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』
FCM_TQ_RECVS_TOTAL	BIGINT	1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』
FCM_TQ_SEND_VOLUME	BIGINT	1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』
FCM_TQ_RECV_VOLUME	BIGINT	1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』
TQ_TOT_SEND_SPILLS	BIGINT	1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』
TCPIP_SEND_VOLUME	BIGINT	1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』
TCPIP_RECV_VOLUME	BIGINT	1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』
IPC_SEND_VOLUME	BIGINT	1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』
IPC_RECV_VOLUME	BIGINT	1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』
POST_THRESHOLD_SORTS	BIGINT	1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
POST_SHRTHRESHOLD_SORTS	BIGINT	1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』
SORT_OVERFLOWS	BIGINT	1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』
AUDIT_EVENTS_TOTAL	BIGINT	857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』
TOTAL_RQST_MAPPED_IN	BIGINT	1664 ページの『total_rqst_mapped_in - マッピングの際の要求の合計 : モニター・エレメント』
TOTAL_RQST_MAPPED_OUT	BIGINT	1664 ページの『total_rqst_mapped_out - マッピングの際に除外された要求の合計 : モニター・エレメント』
ACT_REJECTED_TOTAL	BIGINT	813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』
ACT_ABORTED_TOTAL	BIGINT	809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』
TOTAL_SORTS	BIGINT	1680 ページの『total_sorts - ソート合計 : モニター・エレメント』
TOTAL_ROUTINE_TIME	BIGINT	1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』
TOTAL_COMPILE_PROC_TIME	BIGINT	1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』
TOTAL_COMPILATIONS	BIGINT	1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』
TOTAL_COMPILE_TIME	BIGINT	1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』
TOTAL_IMPLICIT_COMPILE_PROC_TIME	BIGINT	1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』
TOTAL_IMPLICIT_COMPILATIONS	BIGINT	1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』
TOTAL_IMPLICIT_COMPILE_TIME	BIGINT	1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』
TOTAL_RUNSTATS_PROC_TIME	BIGINT	1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』
TOTAL_RUNSTATS	BIGINT	1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』
TOTAL_RUNSTATS_TIME	BIGINT	1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』
TOTAL_REORG_PROC_TIME	BIGINT	1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』
TOTAL_REORGS	BIGINT	1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』
TOTAL_REORG_TIME	BIGINT	1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_LOAD_PROC_TIME	BIGINT	1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』
TOTAL_LOADS	BIGINT	1642 ページの『total_loads - ロード合計 : モニター・エレメント』
TOTAL_LOAD_TIME	BIGINT	1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』
TOTAL_SECTION_PROC_TIME	BIGINT	1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』
TOTAL_APP_SECTION_EXECUTIONS	BIGINT	1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』
TOTAL_SECTION_TIME	BIGINT	1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』
TOTAL_COMMIT_PROC_TIME	BIGINT	1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』
TOTAL_APP_COMMITS	BIGINT	1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』
TOTAL_COMMIT_TIME	BIGINT	1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』
TOTAL_ROLLBACK_PROC_TIME	BIGINT	1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』
TOTAL_APP_ROLLBACKS	BIGINT	1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』
TOTAL_ROLLBACK_TIME	BIGINT	1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』
TOTAL_ROUTINE_USER_CODE _PROC_TIME	BIGINT	1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』
TOTAL_ROUTINE_USER_CODE_TIME	BIGINT	1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』
THRESH_VIOLATIONS	BIGINT	1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』
NUM_LW_THRESH_EXCEEDED	BIGINT	1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』
TOTAL_ROUTINE_INVOCATIONS	BIGINT	1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』
INT_COMMITS	BIGINT	1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』
INT_ROLLBACKS	BIGINT	1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
CAT_CACHE_INSERTS	BIGINT	880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』
CAT_CACHE_LOOKUPS	BIGINT	882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』
PKG_CACHE_INSERTS	BIGINT	1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』
PKG_CACHE_LOOKUPS	BIGINT	1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』
ACT_RQSTS_TOTAL	BIGINT	815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』
TOTAL_ACT_WAIT_TIME	BIGINT	1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』
TOTAL_ACT_TIME	BIGINT	1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』
LOCK_WAIT_TIME_GLOBAL	BIGINT	1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』
LOCK_WAITS_GLOBAL	BIGINT	1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』
RECLAIM_WAIT_TIME	BIGINT	1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』
SPACEMAPPAGE_RECLAIM_WAIT_TIME	BIGINT	1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』
LOCK_TIMEOUTS_GLOBAL	BIGINT	1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』
LOCK_ESCALS_MAXLOCKS	BIGINT	1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』
LOCK_ESCALS_LOCKLIST	BIGINT	1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』
LOCK_ESCALS_GLOBAL	BIGINT	1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』
CF_WAIT_TIME	BIGINT	887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』
CF_WAITS	BIGINT	886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』
POOL_DATA_GBP_L_READS	BIGINT	1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
POOL_DATA_GBP_P_READS	BIGINT	1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』
POOL_DATA_LBP_PAGES_FOUND	BIGINT	1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』
POOL_DATA_GBP_INVALID_PAGES	BIGINT	1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』
POOL_INDEX_GBP_L_READS	BIGINT	1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』
POOL_INDEX_GBP_P_READS	BIGINT	1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』
POOL_INDEX_LBP_PAGES_FOUND	BIGINT	1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』
POOL_INDEX_GBP_INVALID_PAGES	BIGINT	1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』
POOL_XDA_GBP_L_READS	BIGINT	1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』
POOL_XDA_GBP_P_READS	BIGINT	1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』
POOL_XDA_LBP_PAGES_FOUND	BIGINT	1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』
POOL_XDA_GBP_INVALID_PAGES	BIGINT	1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』
EVMON_WAIT_TIME	BIGINT	1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』
EVMON_WAITS_TOTAL	BIGINT	1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』
TOTAL_EXTENDED_LATCH_WAIT_TIME	BIGINT	1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』
TOTAL_EXTENDED_LATCH_WAITS	BIGINT	1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』
TOTAL_STATS_FABRICATION _PROC_TIME	BIGINT	1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』
TOTAL_STATS_FABRICATIONS	BIGINT	1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_STATS_FABRICATION_TIME	BIGINT	1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』
TOTAL_SYNC_RUNSTATS_PROC_TIME	BIGINT	1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』
TOTAL_SYNC_RUNSTATS	BIGINT	1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』
TOTAL_SYNC_RUNSTATS_TIME	BIGINT	1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』
POOL_QUEUED_ASYNC_DATA_REQS	BIGINT	1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_INDEX_REQS	BIGINT	1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_XDA_REQS	BIGINT	1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP _DATA_REQS	BIGINT	1357 ページの『pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP _INDEX_REQS	BIGINT	1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP _XDA_REQS	BIGINT	1366 ページの『pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_OTHER_REQS	BIGINT	1353 ページの『pool_queued_async_other_reqs - プリフェッチャーが処理したその他の要求モニター・エレメント』
POOL_QUEUED_ASYNC_DATA_PAGES	BIGINT	1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_INDEX_PAGES	BIGINT	1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_XDA_PAGES	BIGINT	1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP _DATA_PAGES	BIGINT	1355 ページの『pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求のモニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
POOL_QUEUED_ASYNC_TEMP _INDEX_PAGES	BIGINT	1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP _XDA_PAGES	BIGINT	1364 ページの『pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_DATA_REQS	BIGINT	1308 ページの『pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_INDEX_REQS	BIGINT	1310 ページの『pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_XDA_REQS	BIGINT	1322 ページの『pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_TEMP _DATA_REQS	BIGINT	1314 ページの『pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_TEMP _INDEX_REQS	BIGINT	1317 ページの『pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_TEMP _XDA_REQS	BIGINT	1319 ページの『pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_OTHER_REQS	BIGINT	1313 ページの『pool_failed_async_other_reqs - 失敗した非プリフェッチの要求のモニター・エレメント』
APP_ACT_COMPLETED_TOTAL	BIGINT	838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』
APP_ACT_ABORTED_TOTAL	BIGINT	837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』
APP_ACT_REJECTED_TOTAL	BIGINT	839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』
TOTAL_PEDS	BIGINT	1647 ページの『total_peds - partial early distinct の合計回数のモニター・エレメント』
DISABLED_PEDS	BIGINT	1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』
POST_THRESHOLD_PEDS	BIGINT	1416 ページの『post_threshold_peds - partial early distinct しきい値のモニター・エレメント』
TOTAL_PEAS	BIGINT	1645 ページの『total_peas - partial early aggregation の合計回数のモニター・エレメント』

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: SCMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
POST_THRESHOLD_PEAS	BIGINT	1413 ページの『post_threshold_peas - partial early aggregation しきい値のモニター・エレメント』
TQ_SORT_HEAP_REQUESTS	BIGINT	1703 ページの『tq_sort_heap_requests - 表キュー・ソート・ヒープ要求のモニター・エレメント』
TQ_SORT_HEAP_REJECTIONS	BIGINT	1700 ページの『tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否のモニター・エレメント』
TOTAL_CONNECT_REQUEST_PROC_TIME	BIGINT	1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』
TOTAL_CONNECT_REQUESTS	BIGINT	1622 ページの『total_connect_requests - 接続要求またはユーザー切り替え要求のモニター・エレメント』
TOTAL_CONNECT_REQUEST_TIME	BIGINT	1623 ページの『total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント』
TOTAL_CONNECT_AUTHENTICATION _PROC_TIME	BIGINT	1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』
TOTAL_CONNECT_AUTHENTICATIONS	BIGINT	1619 ページの『total_connect_authentications - 実行された接続またはユーザー切り替え認証のモニター・エレメント』
TOTAL_CONNECT_AUTHENTICATION _TIME	BIGINT	1620 ページの『total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント』
PREFETCH_WAIT_TIME	BIGINT	1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』
PREFETCH_WAITS	BIGINT	1422 ページの『prefetch_waits - プリフェッチャーの待機カウントのモニター・エレメント』
POOL_DATA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1288 ページの『pool_data_gbp_indep_pages_found _in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
POOL_INDEX_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1326 ページの『pool_index_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
POOL_XDA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1393 ページの『pool_xda_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』
COMM_EXIT_WAIT_TIME	BIGINT	905 ページの『comm_exit_wait_time - 通信バッファー出口待機時間のモニター・エレメント』
COMM_EXIT_WAITS	BIGINT	906 ページの『comm_exit_waits - 通信バッファー出口待機回数のモニター・エレメント』
FCM_TQ_RECV_WAITS_TOTAL	BIGINT	

表 59. 統計イベント・モニターに戻される情報 : デフォルトの表名: *SCMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
FCM_MESSAGE_RECV_WAITS_TOTAL	BIGINT	
FCM_TQ_SEND_WAITS_TOTAL	BIGINT	
FCM_MESSAGE_SEND_WAITS_TOTAL	BIGINT	
FCM_SEND_WAITS_TOTAL	BIGINT	
FCM_RECV_WAITS_TOTAL	BIGINT	
IDA_SEND_WAIT_TIME	BIGINT	1092 ページの『ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント』
IDA_SENDS_TOTAL	BIGINT	1093 ページの『ida_sends_total - データ送信回数 : モニター・エレメント』
IDA_SEND_VOLUME	BIGINT	1090 ページの『ida_send_volume - 送信された合計データ量 : モニター・エレメント』
IDA_RECV_WAIT_TIME	BIGINT	1087 ページの『ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント』
IDA_RECVS_TOTAL	BIGINT	1088 ページの『ida_recvs_total - データ受信回数 : モニター・エレメント』
IDA_RECV_VOLUME	BIGINT	1085 ページの『ida_recv_volume - 受信した合計データ量 : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: *WLMETRICS_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
STATISTICS_TIMESTAMP	TIMESTAMP	statistics_timestamp 統計タイム・スタンプ
WORKLOAD_ID	INTEGER	workload_id ワークロード ID
WORKLOAD_NAME	VARCHAR	workload_name ワークロード名
WLM_QUEUE_TIME_TOTAL	BIGINT	1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』
WLM_QUEUE_ASSIGNMENTS_TOTAL	BIGINT	1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』
FCM_TQ_RECV_WAIT_TIME	BIGINT	1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』
FCM_MESSAGE_RECV_WAIT_TIME	BIGINT	1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』
FCM_TQ_SEND_WAIT_TIME	BIGINT	1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』
FCM_MESSAGE_SEND_WAIT_TIME	BIGINT	1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: WLMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
AGENT_WAIT_TIME	BIGINT	829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』
AGENT_WAITS_TOTAL	BIGINT	831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』
LOCK_WAIT_TIME	BIGINT	1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』
LOCK_WAITS	BIGINT	1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』
DIRECT_READ_TIME	BIGINT	989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』
DIRECT_READ_REQS	BIGINT	987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』
DIRECT_WRITE_TIME	BIGINT	996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』
DIRECT_WRITE_REQS	BIGINT	994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』
LOG_BUFFER_WAIT_TIME	BIGINT	1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』
NUM_LOG_BUFFER_FULL	BIGINT	1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』
LOG_DISK_WAIT_TIME	BIGINT	1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』
LOG_DISK_WAITS_TOTAL	BIGINT	1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』
TCPIP_RECV_WAIT_TIME	BIGINT	1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』
TCPIP_RECVS_TOTAL	BIGINT	1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』
CLIENT_IDLE_WAIT_TIME	BIGINT	897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』
IPC_RECV_WAIT_TIME	BIGINT	1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』
IPC_RECVS_TOTAL	BIGINT	1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』
IPC_SEND_WAIT_TIME	BIGINT	1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』
IPC_SENDS_TOTAL	BIGINT	1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』
TCPIP_SEND_WAIT_TIME	BIGINT	1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』
TCPIP_SENDS_TOTAL	BIGINT	1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: *WLMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
POOL_WRITE_TIME	BIGINT	1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』
POOL_READ_TIME	BIGINT	1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』
AUDIT_FILE_WRITE_WAIT_TIME	BIGINT	858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』
AUDIT_FILE_WRITES_TOTAL	BIGINT	860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』
AUDIT_SUBSYSTEM_WAIT_TIME	BIGINT	862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』
AUDIT_SUBSYSTEM_WAITS_TOTAL	BIGINT	864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』
DIAGLOG_WRITE_WAIT_TIME	BIGINT	984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』
DIAGLOG_WRITES_TOTAL	BIGINT	986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』
FCM_SEND_WAIT_TIME	BIGINT	1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』
FCM_RECV_WAIT_TIME	BIGINT	1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』
TOTAL_WAIT_TIME	BIGINT	1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』
RQSTS_COMPLETED_TOTAL	BIGINT	1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』
TOTAL_RQST_TIME	BIGINT	1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』
APP_RQSTS_COMPLETED_TOTAL	BIGINT	841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』
TOTAL_APP_RQST_TIME	BIGINT	1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』
TOTAL_SECTION_SORT_PROC_TIME	BIGINT	1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』
TOTAL_SECTION_SORTS	BIGINT	1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』
TOTAL_SECTION_SORT_TIME	BIGINT	1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』
ROWS_READ	BIGINT	1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: *WLMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
ROWS_MODIFIED	BIGINT	1464 ページの『rows_modified 変更行数 : モニター・エレメント』
POOL_DATA_L_READS	BIGINT	1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』
POOL_INDEX_L_READS	BIGINT	1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』
POOL_TEMP_DATA_L_READS	BIGINT	1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』
POOL_TEMP_INDEX_L_READS	BIGINT	1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』
POOL_XDA_L_READS	BIGINT	1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』
POOL_TEMP_XDA_L_READS	BIGINT	1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』
TOTAL_CPU_TIME	BIGINT	1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』
ACT_COMPLETED_TOTAL	BIGINT	810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』
POOL_DATA_P_READS	BIGINT	1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』
POOL_TEMP_DATA_P_READS	BIGINT	1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』
POOL_XDA_P_READS	BIGINT	1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』
POOL_TEMP_XDA_P_READS	BIGINT	1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』
POOL_INDEX_P_READS	BIGINT	1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』
POOL_TEMP_INDEX_P_READS	BIGINT	1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』
POOL_DATA_WRITES	BIGINT	1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: WLMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
POOL_XDA_WRITES	BIGINT	1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』
POOL_INDEX_WRITES	BIGINT	1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』
DIRECT_READS	BIGINT	991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』
DIRECT_WRITES	BIGINT	998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』
ROWS_RETURNED	BIGINT	1468 ページの『rows_returned 戻り行数 : モニター・エレメント』
DEADLOCKS	BIGINT	978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』
LOCK_TIMEOUTS	BIGINT	1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』
LOCK_ESCALS	BIGINT	1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』
FCM_SENDS_TOTAL	BIGINT	1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』
FCM_RECVS_TOTAL	BIGINT	1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』
FCM_SEND_VOLUME	BIGINT	1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』
FCM_RECV_VOLUME	BIGINT	1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』
FCM_MESSAGE_SENDS_TOTAL	BIGINT	1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』
FCM_MESSAGE_RECVS_TOTAL	BIGINT	1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』
FCM_MESSAGE_SEND_VOLUME	BIGINT	1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』
FCM_MESSAGE_RECV_VOLUME	BIGINT	1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』
FCM_TQ_SENDS_TOTAL	BIGINT	1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』
FCM_TQ_RECVS_TOTAL	BIGINT	1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』
FCM_TQ_SEND_VOLUME	BIGINT	1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』
FCM_TQ_RECV_VOLUME	BIGINT	1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: WLMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
TQ_TOT_SEND_SPILLS	BIGINT	1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』
TCPIP_SEND_VOLUME	BIGINT	1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』
TCPIP_RECV_VOLUME	BIGINT	1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』
IPC_SEND_VOLUME	BIGINT	1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』
IPC_RECV_VOLUME	BIGINT	1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』
POST_THRESHOLD_SORTS	BIGINT	1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』
POST_SHRTHRESHOLD_SORTS	BIGINT	1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』
SORT_OVERFLOWS	BIGINT	1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』
AUDIT_EVENTS_TOTAL	BIGINT	857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』
TOTAL_RQST_MAPPED_IN	BIGINT	1664 ページの『total_rqst_mapped_in - マッピングの際の要求の合計 : モニター・エレメント』
TOTAL_RQST_MAPPED_OUT	BIGINT	1664 ページの『total_rqst_mapped_out - マッピングの際に除外された要求の合計 : モニター・エレメント』
ACT_REJECTED_TOTAL	BIGINT	813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』
ACT_ABORTED_TOTAL	BIGINT	809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』
TOTAL_SORTS	BIGINT	1680 ページの『total_sorts - ソート合計 : モニター・エレメント』
TOTAL_ROUTINE_TIME	BIGINT	1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』
TOTAL_COMPILE_PROC_TIME	BIGINT	1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』
TOTAL_COMPILATIONS	BIGINT	1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』
TOTAL_COMPILE_TIME	BIGINT	1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』
TOTAL_IMPLICIT_COMPILE_PROC_TIME	BIGINT	1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』
TOTAL_IMPLICIT_COMPILATIONS	BIGINT	1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: WLMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_IMPLICIT_COMPILE_TIME	BIGINT	1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』
TOTAL_RUNSTATS_PROC_TIME	BIGINT	1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』
TOTAL_RUNSTATS	BIGINT	1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』
TOTAL_RUNSTATS_TIME	BIGINT	1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』
TOTAL_REORG_PROC_TIME	BIGINT	1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』
TOTAL_REORGS	BIGINT	1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』
TOTAL_REORG_TIME	BIGINT	1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』
TOTAL_LOAD_PROC_TIME	BIGINT	1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』
TOTAL_LOADS	BIGINT	1642 ページの『total_loads - ロード合計 : モニター・エレメント』
TOTAL_LOAD_TIME	BIGINT	1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』
TOTAL_SECTION_PROC_TIME	BIGINT	1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』
TOTAL_APP_SECTION_EXECUTIONS	BIGINT	1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』
TOTAL_SECTION_TIME	v	1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』
TOTAL_COMMIT_PROC_TIME	BIGINT	1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』
TOTAL_APP_COMMITS	BIGINT	1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』
TOTAL_COMMIT_TIME	BIGINT	1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』
TOTAL_ROLLBACK_PROC_TIME	BIGINT	1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』
TOTAL_APP_ROLLBACKS	BIGINT	1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』
TOTAL_ROLLBACK_TIME	BIGINT	1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』
TOTAL_ROUTINE_USER_CODE _PROC_TIME	BIGINT	1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: WLMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_ROUTINE_USER_CODE_TIME	BIGINT	1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』
THRESH_VIOLATIONS	BIGINT	1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』
NUM_LW_THRESH_EXCEEDED	BIGINT	1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』
TOTAL_ROUTINE_INVOCATIONS	BIGINT	1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』
INT_COMMITS	BIGINT	1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』
INT_ROLLBACKS	BIGINT	1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』
CAT_CACHE_INSERTS	BIGINT	880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』
CAT_CACHE_LOOKUPS	BIGINT	882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』
PKG_CACHE_INSERTS	BIGINT	1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』
PKG_CACHE_LOOKUPS	BIGINT	1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』
ACT_RQSTS_TOTAL	BIGINT	815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』
TOTAL_ACT_WAIT_TIME	BIGINT	1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』
TOTAL_ACT_TIME	BIGINT	1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』
LOCK_WAIT_TIME_GLOBAL	BIGINT	1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』
LOCK_WAITS_GLOBAL	BIGINT	1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』
RECLAIM_WAIT_TIME	BIGINT	1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』
SPACEMAPPAGE_RECLAIM_WAIT_TIME	BIGINT	1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』
LOCK_TIMEOUTS_GLOBAL	BIGINT	1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』
LOCK_ESCALS_MAXLOCKS	BIGINT	1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: *WLMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
LOCK_ESCALS_LOCKLIST	BIGINT	1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』
LOCK_ESCALS_GLOBAL	BIGINT	1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』
CF_WAIT_TIME	BIGINT	887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』
CF_WAITS	BIGINT	886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』
POOL_DATA_GBP_L_READS	BIGINT	1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』
POOL_DATA_GBP_P_READS	BIGINT	1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』
POOL_DATA_LBP_PAGES_FOUND	BIGINT	1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』
POOL_DATA_GBP_INVALID_PAGES	BIGINT	1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』
POOL_INDEX_GBP_L_READS	BIGINT	1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』
POOL_INDEX_GBP_P_READS	BIGINT	1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』
POOL_INDEX_LBP_PAGES_FOUND	BIGINT	1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』
POOL_INDEX_GBP_INVALID_PAGES	BIGINT	1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』
POOL_XDA_GBP_L_READS	BIGINT	1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』
POOL_XDA_GBP_P_READS	BIGINT	1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』
POOL_XDA_LBP_PAGES_FOUND	BIGINT	1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: WLMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
POOL_XDA_GBP_INVALID_PAGES	BIGINT	1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』
EVMON_WAIT_TIME	BIGINT	1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』
EVMON_WAITS_TOTAL	BIGINT	1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』
TOTAL_EXTENDED_LATCH_WAIT_TIME	BIGINT	1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』
TOTAL_EXTENDED_LATCH_WAITS	BIGINT	1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』
TOTAL_STATS_FABRICATION_PROC_TIME	BIGINT	1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』
TOTAL_STATS_FABRICATIONS	BIGINT	1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』
TOTAL_STATS_FABRICATION_TIME	BIGINT	1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』
TOTAL_SYNC_RUNSTATS_PROC_TIME	BIGINT	1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』
TOTAL_SYNC_RUNSTATS	BIGINT	1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』
TOTAL_SYNC_RUNSTATS_TIME	BIGINT	1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』
POOL_QUEUED_ASYNC_DATA_REQS	BIGINT	1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_INDEX_REQS	BIGINT	1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_XDA_REQS	BIGINT	1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP_DATA_REQS	BIGINT	1357 ページの『pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP_INDEX_REQS	BIGINT	1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP_XDA_REQS	BIGINT	1366 ページの『pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求のモニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: *WLMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
POOL_QUEUED_ASYNC_OTHER_REQS	BIGINT	1353 ページの『pool_queued_async_other_reqs - プリフェッチャーが処理したその他の要求モニター・エレメント』
POOL_QUEUED_ASYNC_DATA_PAGES	BIGINT	1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_INDEX_PAGES	BIGINT	1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_XDA_PAGES	BIGINT	1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP _DATA_PAGES	BIGINT	1355 ページの『pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP _INDEX_PAGES	BIGINT	1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』
POOL_QUEUED_ASYNC_TEMP _XDA_PAGES	BIGINT	1364 ページの『pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_DATA_REQS	BIGINT	1308 ページの『pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_INDEX_REQS	BIGINT	1310 ページの『pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_XDA_REQS	BIGINT	1322 ページの『pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_TEMP _DATA_REQS	BIGINT	1314 ページの『pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_TEMP _INDEX_REQS	BIGINT	1317 ページの『pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_TEMP _XDA_REQS	BIGINT	1319 ページの『pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求のモニター・エレメント』
POOL_FAILED_ASYNC_OTHER_REQS	BIGINT	1313 ページの『pool_failed_async_other_reqs - 失敗した非プリフェッチの要求のモニター・エレメント』
APP_ACT_COMPLETED_TOTAL	BIGINT	838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: WLMETRICS_evmon-name (続き)

列名	データ・タイプ	説明
APP_ACT_ABORTED_TOTAL	BIGINT	837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』
APP_ACT_REJECTED_TOTAL	BIGINT	839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』
TOTAL_PEDS	BIGINT	1647 ページの『total_peds - partial early distinct の合計回数のモニター・エレメント』
DISABLED_PEDS	BIGINT	1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』
POST_THRESHOLD_PEDS	BIGINT	1416 ページの『post_threshold_peds - partial early distinct しきい値のモニター・エレメント』
TOTAL_PEAS	BIGINT	1645 ページの『total_peas - partial early aggregation の合計回数のモニター・エレメント』
POST_THRESHOLD_PEAS	BIGINT	1413 ページの『post_threshold_peas - partial early aggregation しきい値のモニター・エレメント』
TQ_SORT_HEAP_REQUESTS	BIGINT	1703 ページの『tq_sort_heap_requests - 表キュー・ソート・ヒープ要求のモニター・エレメント』
TQ_SORT_HEAP_REJECTIONS	BIGINT	1700 ページの『tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否のモニター・エレメント』
TOTAL_CONNECT_REQUEST _PROC_TIME	BIGINT	1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』
TOTAL_CONNECT_REQUESTS	BIGINT	1622 ページの『total_connect_requests - 接続要求またはユーザー切り替え要求のモニター・エレメント』
TOTAL_CONNECT_REQUEST_TIME	BIGINT	1623 ページの『total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント』
TOTAL_CONNECT_AUTHENTICATION _PROC_TIME	BIGINT	1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』
TOTAL_CONNECT_AUTHENTICATIONS	BIGINT	1619 ページの『total_connect_authentications - 実行された接続またはユーザー切り替え認証のモニター・エレメント』
TOTAL_CONNECT_AUTHENTICATION _TIME	BIGINT	1620 ページの『total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント』
PREFETCH_WAIT_TIME	BIGINT	1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』
PREFETCH_WAITS	BIGINT	1422 ページの『prefetch_waits - プリフェッチャーの待機カウントのモニター・エレメント』

表 60. 統計イベント・モニターに戻される情報 : デフォルトの表名: *WLMETRICS_evmon-name (続き)*

列名	データ・タイプ	説明
POOL_DATA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1288 ページの『pool_data_gbp_indep_pages_found _in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
POOL_INDEX_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1326 ページの『pool_index_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
POOL_XDA_GBP_INDEP_PAGES _FOUND_IN_LBP	BIGINT	1393 ページの『pool_xda_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』
COMM_EXIT_WAIT_TIME	BIGINT	905 ページの『comm_exit_wait_time - 通信バッファー出口待機時間のモニター・エレメント』
COMM_EXIT_WAITS	BIGINT	906 ページの『comm_exit_waits - 通信バッファー出口待機回数のモニター・エレメント』
FCM_TQ_RECV_WAITS_TOTAL	BIGINT	
FCM_MESSAGE_RECV_WAITS_TOTAL	BIGINT	
FCM_TQ_SEND_WAITS_TOTAL	BIGINT	
FCM_MESSAGE_SEND_WAITS_TOTAL	BIGINT	
FCM_SEND_WAITS_TOTAL	BIGINT	
FCM_RECV_WAITS_TOTAL	BIGINT	
IDA_SEND_WAIT_TIME	BIGINT	1092 ページの『ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント』
IDA_SENDS_TOTAL	BIGINT	1093 ページの『ida_sends_total - データ送信回数 : モニター・エレメント』
IDA_SEND_VOLUME	BIGINT	1090 ページの『ida_send_volume - 送信された合計データ量 : モニター・エレメント』
IDA_RECV_WAIT_TIME	BIGINT	1087 ページの『ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント』
IDA_RECVS_TOTAL	BIGINT	1088 ページの『ida_recvs_total - データ受信回数 : モニター・エレメント』
IDA_RECV_VOLUME	BIGINT	1085 ページの『ida_recv_volume - 受信した合計データ量 : モニター・エレメント』

表 61. 統計イベント・モニターに戻される情報 : デフォルトの表名: *CONTROL_evmon-name*

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ

表 61. 統計イベント・モニターに戻される情報 : デフォルトの表名: CONTROL_evmon-name (続き)

列名	データ・タイプ	説明
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号

system_metrics および activity_metrics モニター・エレメント用に XML に書き込まれる情報:

activity_metrics モニター・エレメントは、 MON_GET_ACTIVITY_DETAILS 表関数、 MON_GET_PKG_CACHE_STMT_DETAILS 表関数、およびアクティビティー・イベント・モニターで報告されます。**system_metrics** モニター・エレメントは、
MON_GET_CONNECTION_DETAILS、MON_GET_UNIT_OF_WORK_DETAILS、MON_GET_SERVICE_SUBCLASS_DETAILS、MON_GET_WORKLOAD_DETAILS 表関数、および統計イベント・モニターで報告されます。これは、
sqllib/misc/DB2MonCommon.xsd ファイルにも記載されています。

system_metrics

システム・レベル・メトリック。

エレメント・コンテント: (397 ページの『wlm_queue_time_total』、 397 ページの『wlm_queue_assignments_total』、 397 ページの『fcm_tq_recv_wait_time』 、 398 ページの『fcm_message_recv_wait_time』、 398 ページの『fcm_tq_send_wait_time』、 398 ページの『fcm_message_send_wait_time』、 380 ページの『agent_wait_time』、 380 ページの『agent_waits_total』、 399 ページの『lock_wait_time』、 399 ページの『lock_waits』、 399 ページの『direct_read_time』、 399 ページの『direct_read_reqs』、 400 ページの『direct_write_time』、 400 ページの『direct_write_reqs』、 400 ページの『log_buffer_wait_time』、 400 ページの『num_log_buffer_full』、 401 ページの『log_disk_wait_time』、 401 ページの『log_disk_waits_total』、 380 ページの『tcpip_recv_wait_time』、 381 ページの『tcpip_recvs_total』、 381 ページの『client_idle_wait_time』、 381 ページの『ipc_recv_wait_time』、 381 ページの『ipc_recvs_total』、 382 ページの『ipc_send_wait_time』、 382 ページの『ipc_sends_total』、 382 ページの『tcpip_send_wait_time』、 382 ページの『tcpip_sends_total』、 401 ページの『pool_write_time』、 402 ページの『pool_read_time』、 402 ページの『audit_file_write_wait_time』、 402 ページの『audit_file_writes_total』、 402 ページの『audit_subsystem_wait_time』、 403 ページの『audit_subsystem_waits_total』、 403 ページの『diaglog_write_wait_time』、 403 ページの『diaglog_writes_total』、 403 ページの『fcm_send_wait_time』、 404 ページの『fcm_recv_wait_time』、 383 ページの『total_wait_time』、 383 ページの『total_rqst_time』、 383 ページの『rqsts_completed_total』、 383 ページの『total_app_rqst_time』、 384 ページの『app_rqsts_completed_total』、 404 ページの『total_section_sort_proc_time』、 404 ページの『total_section_sort_time』、 405 ページの『total_section_sorts』、 405 ページの『rows_read』、 406 ページの『rows_modified』、 406 ページの『pool_data_l_reads』、 406 ページの『pool_index_l_reads』、 406 ページの『pool_temp_data_l_reads』、 407 ページの『pool_temp_index_l_reads』、 407 ページの

の『pool_xda_l_reads』、 407 ページの『pool_temp_xda_l_reads』、 407 ページの『total_cpu_time』、 384 ページの『act_completed_total』、 408 ページの『pool_data_p_reads』、 408 ページの『pool_temp_data_p_reads』、 408 ページの『pool_xda_p_reads』、 409 ページの『pool_temp_xda_p_reads』 、 409 ページの『pool_index_p_reads』、 409 ページの『pool_temp_index_p_reads』、 409 ページの『pool_data_writes』、 410 ページの『pool_xda_writes』、 410 ページの『pool_index_writes』、 410 ページの『direct_reads』、 410 ページの『direct_writes』、 411 ページの『rows_returned』、 411 ページの『デッドロック』、 411 ページの『lock_timeouts』、 411 ページの『lock_escals』、 412 ページの『fcm_sends_total』、 412 ページの『fcm_recvs_total』、 412 ページの『fcm_send_volume』、 413 ページの『fcm_recv_volume』、 413 ページの『fcm_message_sends_total』、 413 ページの『fcm_message_recvs_total』、 413 ページの『fcm_message_send_volume』、 414 ページの『fcm_message_recv_volume』、 414 ページの『fcm_tq_sends_total』、 414 ページの『fcm_tq_recvs_total』、 414 ページの『fcm_tq_send_volume』、 415 ページの『fcm_tq_recv_volume』、 415 ページの『tq_tot_send_spills』、 384 ページの『tcpip_send_volume』、 384 ページの『tcpip_recv_volume』、 385 ページの『ipc_send_volume』、 385 ページの『ipc_recv_volume』、 415 ページの『post_threshold_sorts』、 415 ページの『post_shrthreshold_sorts』、 416 ページの『sort_overflows』、 416 ページの『audit_events_total』、 385 ページの『total_rqst_mapped_in』 {0 または 1 個 (?)} 、 385 ページの『total_rqst_mapped_out』 { 0 または 1 個 (?)} , 386 ページの『act_rejected_total』, 386 ページの『act_aborted_total』, 416 ページの『total_sorts』, 419 ページの『total_routine_time』, 386 ページの『total_compile_proc_time』, 386 ページの『total_compile_time』, 387 ページの『total_compilations』, 387 ページの『total_implicit_compile_proc_time』, 387 ページの『total_implicit_compile_time』, 387 ページの『total_implicit_compilations』, 388 ページの『total_runstats_proc_time』, 388 ページの『total_runstats_time』, 388 ページの『total_runstats』, 388 ページの『total_reorg_proc_time』, 389 ページの『total_reorg_time』, 389 ページの『total_reorgs』, 389 ページの『total_load_proc_time』, 389 ページの『total_load_time』, 390 ページの『total_loads』, 418 ページの『total_section_proc_time』, 418 ページの『total_section_time』, 418 ページの『total_app_section_executions』, 390 ページの『total_commit_proc_time』, 390 ページの『total_commit_time』, 390 ページの『total_app_commits』, 391 ページの『total_rollback_proc_time』, 391 ページの『total_rollback_time』, 391 ページの『total_app_rollbacks』, 418 ページの『total_routine_user_code_proc_time』, 419 ページの『total_routine_user_code_time』, 419 ページの『thresh_violations』, 419 ページの『num_lw_thresh_exceeded』, 420 ページの『total_routine_invocations』, 391 ページの『int_commits』, 392 ページの『int_rollbacks』, 392 ページの『cat_cache_inserts』, 392 ページの『cat_cache_lookups』, 392 ページの『pkg_cache_inserts』, 393 ページの『pkg_cache_lookups』, 393 ページの『act_rqsts_total』, 404 ページの『total_act_wait_time』, 405 ページの『total_act_time』, 420 ページの『lock_wait_time_global』, 420 ページの『lock_waits_global』, 421 ページの『reclaim_wait_time』, 421 ページの『spacemappage_reclaim_wait_time』, 421 ページの『lock_timeouts_global』, 421 ページの『lock_escals_maxlocks』, 422 ページの『lock_escals_locklist』, 422 ページの『lock_escals_global』, 422 ページの『cf_wait_time』, 422 ページの『cf_waits』, 423 ページの『pool_data_gbp_l_reads』, 423 ページの『pool_data_gbp_p_reads』, 423 ページの

『pool_data_lbp_pages_found』, 424 ページの『pool_data_gbp_invalid_pages』, 424 ページの『pool_index_gbp_l_reads』, 424 ページの『pool_index_gbp_p_reads』, 424 ページの『pool_index_lbp_pages_found』, 425 ページの『pool_index_gbp_invalid_pages』, 425 ページの『pool_xda_gbp_l_reads』, 425 ページの『pool_xda_gbp_p_reads』, 426 ページの『pool_xda_lbp_pages_found』, 426 ページの『pool_xda_gbp_invalid_pages』, 426 ページの『evmon_wait_time』, 426 ページの『evmon_waits_total』, 427 ページの『total_extended_latch_wait_time』, 427 ページの『total_extended_latch_waits』, 393 ページの『total_stats_fabrication_proc_time』, 393 ページの『total_stats_fabrication_time』, 394 ページの『total_stats_fabrications』, 394 ページの『total_sync_runstats_proc_time』, 394 ページの『total_sync_runstats_time』, 394 ページの『total_sync_runstats』, 427 ページの『total_disp_run_queue_time』, 427 ページの『pool_queued_async_data_reqs』, 428 ページの『pool_queued_async_index_reqs』, 428 ページの『pool_queued_async_xda_reqs』, 428 ページの『pool_queued_async_temp_data_reqs』, 429 ページの『pool_queued_async_temp_index_reqs』, 429 ページの『pool_queued_async_temp_xda_reqs』, 429 ページの『pool_queued_async_other_reqs』, 429 ページの『pool_queued_async_data_pages』, 430 ページの『pool_queued_async_index_pages』, 430 ページの『pool_queued_async_xda_pages』, 430 ページの『pool_queued_async_temp_data_pages』, 430 ページの『pool_queued_async_temp_index_pages』, 431 ページの『pool_queued_async_temp_xda_pages』, 431 ページの『pool_failed_async_data_reqs』, 431 ページの『pool_failed_async_index_reqs』, 432 ページの『pool_failed_async_xda_reqs』, 432 ページの『pool_failed_async_temp_data_reqs』, 432 ページの『pool_failed_async_temp_index_reqs』, 432 ページの『pool_failed_async_temp_xda_reqs』, 433 ページの『pool_failed_async_other_reqs』, 395 ページの『app_act_completed_total』, 395 ページの『app_act_aborted_total』, 395 ページの『app_act_rejected_total』, 433 ページの『total_peds』, 433 ページの『disabled_peds』、 433 ページの『post_threshold_peds』、 434 ページの『total_peas』、 434 ページの『post_threshold_peas』、 434 ページの『tq_sort_heap_requests』、 434 ページの『tq_sort_heap_rejections』、 395 ページの『total_connect_request_proc_time』、 396 ページの『total_connect_request_time』、 396 ページの『total_connect_requests』、 396 ページの『total_connect_authentication_proc_time』、 396 ページの『total_connect_authentication_time』、 397 ページの『total_connect_authentications』、 435 ページの『prefetch_wait_time』、 435 ページの『prefetch_waits』、 435 ページの『pool_data_gbp_indep_pages_found_in_lbp』、 435 ページの『pool_index_gbp_indep_pages_found_in_lbp』、 436 ページの『pool_xda_gbp_indep_pages_found_in_lbp』、ANY コンテント (検証しない (skip)) {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
release	xs:long			必須	

QName	タイプ	固定	デフォルト	使用	アノテーション
ANY ネーム・スペースの ANY 属性					

activity_metrics

アクティビティー・レベル・メトリック。

エレメント・コンテンツ: (397 ページの『wlm_queue_time_total』、 397 ページの『wlm_queue_assignments_total』、 397 ページの『fcm_tq_recv_wait_time』 、 398 ページの『fcm_message_recv_wait_time』、 398 ページの『fcm_tq_send_wait_time』、 398 ページの『fcm_message_send_wait_time』、 399 ページの『lock_wait_time』、 399 ページの『lock_waits』、 399 ページの『direct_read_time』、 399 ページの『direct_read_reqs』、 400 ページの『direct_write_time』、 400 ページの『direct_write_reqs』、 400 ページの『log_buffer_wait_time』、 400 ページの『num_log_buffer_full』、 401 ページの『log_disk_wait_time』、 401 ページの『log_disk_waits_total』、 401 ページの『pool_write_time』、 402 ページの『pool_read_time』、 402 ページの『audit_file_write_wait_time』、 402 ページの『audit_file_writes_total』、 402 ページの『audit_subsystem_wait_time』、 403 ページの『audit_subsystem_waits_total』、 403 ページの『diaglog_write_wait_time』、 403 ページの『diaglog_writes_total』、 403 ページの『fcm_send_wait_time』、 404 ページの『fcm_recv_wait_time』、 404 ページの『total_act_wait_time』、 404 ページの『total_section_sort_proc_time』、 404 ページの『total_section_sort_time』、 405 ページの『total_section_sorts』、 405 ページの『total_act_time』、 405 ページの『rows_read』、 406 ページの『rows_modified』、 406 ページの『pool_data_l_reads』、 406 ページの『pool_index_l_reads』、 406 ページの『pool_temp_data_l_reads』、 407 ページの『pool_temp_index_l_reads』、 407 ページの『pool_xda_l_reads』、 407 ページの『pool_temp_xda_l_reads』、 407 ページの『total_cpu_time』、 408 ページの『pool_data_p_reads』、 408 ページの『pool_temp_data_p_reads』、 408 ページの『pool_xda_p_reads』、 409 ページの『pool_temp_xda_p_reads』、 409 ページの『pool_index_p_reads』、 409 ページの『pool_temp_index_p_reads』、 409 ページの『pool_data_writes』、 410 ページの『pool_xda_writes』、 410 ページの『pool_index_writes』、 410 ページの『direct_reads』、 410 ページの『direct_writes』、 411 ページの『rows_returned』、 411 ページの『デッドロック』、 411 ページの『lock_timeouts』、 411 ページの『lock_escals』、 412 ページの『fcm_sends_total』、 412 ページの『fcm_recvs_total』、 412 ページの『fcm_send_volume』、 413 ページの『fcm_recv_volume』、 413 ページの『fcm_message_sends_total』、 413 ページの『fcm_message_recvs_total』、 413 ページの『fcm_message_send_volume』、 414 ページの『fcm_message_recv_volume』、 414 ページの『fcm_tq_sends_total』、 414 ページの『fcm_tq_recvs_total』、 414 ページの『fcm_tq_send_volume』、 415 ページの『fcm_tq_recv_volume』、 415 ページの『tq_tot_send_spills』、 415 ページの『post_threshold_sorts』、 415 ページの『post_shrthreshold_sorts』、 416 ページの『sort_overflows』、 416 ページの『audit_events_total』、 416 ページの『total_sorts』、 417 ページの『stmt_exec_time』、 417 ページの『coord_stmt_exec_time』 {0 または 1 個 (?)} , 417 ページの『total_routine_non_sect_proc_time』, 417 ページの『total_routine_non_sect_time』,

418 ページの『total_section_proc_time』, 418 ページの『total_section_time』, 418 ページの『total_app_section_executions』, 418 ページの『total_routine_user_code_proc_time』, 419 ページの『total_routine_user_code_time』, 419 ページの『total_routine_time』, 419 ページの『thresh_violations』, 419 ページの『num_lw_thresh_exceeded』, 420 ページの『total_routine_invocations』, 420 ページの『lock_wait_time_global』, 420 ページの『lock_waits_global』, 421 ページの『reclaim_wait_time』, 421 ページの『spacemappage_reclaim_wait_time』, 421 ページの『lock_timeouts_global』, 421 ページの『lock_escals_maxlocks』, 422 ページの『lock_escals_locklist』, 422 ページの『lock_escals_global』, 422 ページの『cf_wait_time』, 422 ページの『cf_waits』, 423 ページの『pool_data_gbp_l_reads』, 423 ページの『pool_data_gbp_p_reads』, 423 ページの『pool_data_lbp_pages_found』, 424 ページの『pool_data_gbp_invalid_pages』, 424 ページの『pool_index_gbp_l_reads』, 424 ページの『pool_index_gbp_p_reads』, 424 ページの『pool_index_lbp_pages_found』, 425 ページの『pool_index_gbp_invalid_pages』, 425 ページの『pool_xda_gbp_l_reads』, 425 ページの『pool_xda_gbp_p_reads』, 426 ページの『pool_xda_lbp_pages_found』, 426 ページの『pool_xda_gbp_invalid_pages』, 426 ページの『evmon_wait_time』, 426 ページの『evmon_waits_total』, 427 ページの『total_extended_latch_wait_time』, 427 ページの『total_extended_latch_waits』, 427 ページの『total_disp_run_queue_time』, 427 ページの『pool_queued_async_data_reqs』, 428 ページの『pool_queued_async_index_reqs』, 428 ページの『pool_queued_async_xda_reqs』, 428 ページの『pool_queued_async_temp_data_reqs』, 429 ページの『pool_queued_async_temp_index_reqs』, 429 ページの『pool_queued_async_temp_xda_reqs』, 429 ページの『pool_queued_async_other_reqs』, 429 ページの『pool_queued_async_data_pages』, 430 ページの『pool_queued_async_index_pages』, 430 ページの『pool_queued_async_xda_pages』, 430 ページの『pool_queued_async_temp_data_pages』, 430 ページの『pool_queued_async_temp_index_pages』, 431 ページの『pool_queued_async_temp_xda_pages』, 431 ページの『pool_failed_async_data_reqs』, 431 ページの『pool_failed_async_index_reqs』, 432 ページの『pool_failed_async_xda_reqs』, 432 ページの『pool_failed_async_temp_data_reqs』, 432 ページの『pool_failed_async_temp_index_reqs』, 432 ページの『pool_failed_async_temp_xda_reqs』, 433 ページの『pool_failed_async_other_reqs』, 433 ページの『total_peds』、 433 ページの『disabled_peds』、 433 ページの『post_threshold_peds』、 434 ページの『total_peas』、 434 ページの『post_threshold_peas』、 434 ページの『tq_sort_heap_requests』、 434 ページの『tq_sort_heap_rejections』、 435 ページの『prefetch_wait_time』、 435 ページの『prefetch_waits』、 435 ページの『pool_data_gbp_indep_pages_found_in_lbp』、 435 ページの『pool_index_gbp_indep_pages_found_in_lbp』、 436 ページの『pool_xda_gbp_indep_pages_found_in_lbp』、ANY コンテント (検証しない (skip)) {0 個以上 (*)})

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
release	xs:long			必須	
ANY ネーム・スペースの ANY 属性					

stmt_value_index

このエレメントは、SQL ステートメントで使用される入力パラメーター・マーカーまたはホスト変数の位置を表します。詳しくは、モニター・エレメント 1543 ページの『stmt_value_index 値索引』 を参照してください。

これを含むエレメント:

エレメント・コンテント :

タイプ	ファセット
xs:int	

stmt_value_isnull

これを含むエレメント:

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			必須	

stmt_value_isreopt

これを含むエレメント:

属性:

QName	タイプ	固定	デフォルト	使用	アノテーション
id	xs:int			必須	

stmt_value_type

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

これを含むエレメント:

エレメント・コンテント :

タイプ	ファセット
xs:string	最大長: 16

stmt_value_data

このエレメントは、SQL ステートメントに関連したデータ値のストリング表記です。詳しくは、モニター・エレメント 1543 ページの『stmt_value_data 値データ』を参照してください。

これを含むエレメント:

エレメント・コンテント :

タイプ	ファセット
xs:string	最大長: 32768

agent_wait_time

詳しくは、モニター・エレメント 829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

agent_waits_total

詳しくは、モニター・エレメント 831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tcpip_recv_wait_time

詳しくは、モニター・エレメント 1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tcpip_recvs_total

詳しくは、モニター・エレメント 1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

client_idle_wait_time

詳しくは、モニター・エレメント 897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

ipc_recv_wait_time

詳しくは、モニター・エレメント 1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

ipc_recvs_total

詳しくは、モニター・エレメント 1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

ipc_send_wait_time

詳しくは、モニター・エレメント 1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

ipc_sends_total

詳しくは、モニター・エレメント 1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tcpip_send_wait_time

詳しくは、モニター・エレメント 1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tcpip_sends_total

詳しくは、モニター・エレメント 1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_wait_time

詳しくは、モニター・エレメント 1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_rqst_time

詳しくは、モニター・エレメント 1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

rqsts_completed_total

詳しくは、モニター・エレメント 1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_app_rqst_time

詳しくは、モニター・エレメント 1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

app_rqsts_completed_total

詳しくは、モニター・エレメント 841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

act_completed_total

詳しくは、モニター・エレメント 810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tcpip_send_volume

詳しくは、モニター・エレメント 1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tcpip_recv_volume

詳しくは、モニター・エレメント 1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

ipc_send_volume

詳しくは、モニター・エレメント 1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

ipc_recv_volume

詳しくは、モニター・エレメント 1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_rqst_mapped_in

詳しくは、モニター・エレメント 1664 ページの『total_rqst_mapped_in - マッピングの際の要求の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_rqst_mapped_out

詳しくは、モニター・エレメント 1664 ページの『total_rqst_mapped_out - マッピングの際に除外された要求の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

act_rejected_total

詳しくは、モニター・エレメント 813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

act_aborted_total

詳しくは、モニター・エレメント 809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_compile_proc_time

詳しくは、モニター・エレメント 1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_compile_time

詳しくは、モニター・エレメント 1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_compilations

詳しくは、モニター・エレメント 1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_implicit_compile_proc_time

詳しくは、モニター・エレメント 1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_implicit_compile_time

詳しくは、モニター・エレメント 1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_implicit_compilations

詳しくは、モニター・エレメント 1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_runstats_proc_time

詳しくは、モニター・エレメント 1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_runstats_time

詳しくは、モニター・エレメント 1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_runstats

詳しくは、モニター・エレメント 1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_reorg_proc_time

詳しくは、モニター・エレメント 1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_reorg_time

詳しくは、モニター・エレメント 1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_reorgs

詳しくは、モニター・エレメント 1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_load_proc_time

詳しくは、モニター・エレメント 1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_load_time

詳しくは、モニター・エレメント 1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_loads

詳しくは、モニター・エレメント 1642 ページの『total_loads - ロード合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_commit_proc_time

詳しくは、モニター・エレメント 1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_commit_time

詳しくは、モニター・エレメント 1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_app_commits

詳しくは、モニター・エレメント 1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_rollback_proc_time

詳しくは、モニター・エレメント 1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_rollback_time

詳しくは、モニター・エレメント 1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_app_rollbacks

詳しくは、モニター・エレメント 1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

int_commits

詳しくは、モニター・エレメント 1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

int_rollbacks

詳しくは、モニター・エレメント 1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

cat_cache_inserts

詳しくは、モニター・エレメント 880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

cat_cache_lookups

詳しくは、モニター・エレメント 882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pkg_cache_inserts

詳しくは、モニター・エレメント 1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pkg_cache_lookups

詳しくは、モニター・エレメント 1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

act_rqsts_total

詳しくは、モニター・エレメント 815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_stats_fabrication_proc_time

詳しくは、モニター・エレメント 1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_stats_fabrication_time

詳しくは、モニター・エレメント 1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_stats_fabrications

詳しくは、モニター・エレメント 1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_sync_runstats_proc_time

詳しくは、モニター・エレメント 1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_sync_runstats_time

詳しくは、モニター・エレメント 1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_sync_runstats

詳しくは、モニター・エレメント 1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

app_act_completed_total

詳しくは、モニター・エレメント 838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

app_act_aborted_total

詳しくは、モニター・エレメント 837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

app_act_rejected_total

詳しくは、モニター・エレメント 839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_connect_request_proc_time

詳しくは、モニター・エレメント 1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_connect_request_time

詳しくは、モニター・エレメント 1623 ページの『total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_connect_requests

詳しくは、モニター・エレメント 1622 ページの『total_connect_requests - 接続要求またはユーザー切り替え要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_connect_authentication_proc_time

詳しくは、モニター・エレメント 1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_connect_authentication_time

詳しくは、モニター・エレメント 1620 ページの『total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_connect_authentications

詳しくは、モニター・エレメント 1619 ページの『total_connect_authentications - 実行された接続またはユーザー切り替え認証のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

wlm_queue_time_total

詳しくは、モニター・エレメント 1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

wlm_queue_assignments_total

詳しくは、モニター・エレメント 1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_tq_recv_wait_time

詳しくは、モニター・エレメント 1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_message_recv_wait_time

詳しくは、モニター・エレメント 1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_tq_send_wait_time

詳しくは、モニター・エレメント 1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_message_send_wait_time

詳しくは、モニター・エレメント 1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_wait_time

詳しくは、モニター・エレメント 1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_waits

詳しくは、モニター・エレメント 1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

direct_read_time

詳しくは、モニター・エレメント 989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

direct_read_reqs

詳しくは、モニター・エレメント 987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

direct_write_time

詳しくは、モニター・エレメント 996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

direct_write_reqs

詳しくは、モニター・エレメント 994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

log_buffer_wait_time

詳しくは、モニター・エレメント 1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

num_log_buffer_full

詳しくは、モニター・エレメント 1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

log_disk_wait_time

詳しくは、モニター・エレメント 1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

log_disk_waits_total

詳しくは、モニター・エレメント 1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_write_time

詳しくは、モニター・エレメント 1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_read_time

詳しくは、モニター・エレメント 1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

audit_file_write_wait_time

詳しくは、モニター・エレメント 858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

audit_file_writes_total

詳しくは、モニター・エレメント 860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

audit_subsystem_wait_time

詳しくは、モニター・エレメント 862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

audit_subsystem_waits_total

詳しくは、モニター・エレメント 864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

diaglog_write_wait_time

詳しくは、モニター・エレメント 984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

diaglog_writes_total

詳しくは、モニター・エレメント 986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_send_wait_time

詳しくは、モニター・エレメント 1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

fcm_recv_wait_time

詳しくは、モニター・エレメント 1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

total_act_wait_time

詳しくは、モニター・エレメント 1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

total_section_sort_proc_time

詳しくは、モニター・エレメント 1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

total_section_sort_time

詳しくは、モニター・エレメント 1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_section_sorts

詳しくは、モニター・エレメント 1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_act_time

詳しくは、モニター・エレメント 1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

rows_read

詳しくは、モニター・エレメント 1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

rows_modified

詳しくは、モニター・エレメント 1464 ページの『rows_modified 変更行数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_data_l_reads

詳しくは、モニター・エレメント 1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_index_l_reads

詳しくは、モニター・エレメント 1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_temp_data_l_reads

詳しくは、モニター・エレメント 1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_temp_index_l_reads

詳しくは、モニター・エレメント 1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_xda_l_reads

詳しくは、モニター・エレメント 1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_temp_xda_l_reads

詳しくは、モニター・エレメント 1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_cpu_time

詳しくは、モニター・エレメント 1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_data_p_reads

詳しくは、モニター・エレメント 1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_temp_data_p_reads

詳しくは、モニター・エレメント 1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_xda_p_reads

詳しくは、モニター・エレメント 1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_temp_xda_p_reads

詳しくは、モニター・エレメント 1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_index_p_reads

詳しくは、モニター・エレメント 1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_temp_index_p_reads

詳しくは、モニター・エレメント 1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_data_writes

詳しくは、モニター・エレメント 1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_xda_writes

詳しくは、モニター・エレメント 1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_index_writes

詳しくは、モニター・エレメント 1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

direct_reads

詳しくは、モニター・エレメント 991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

direct_writes

詳しくは、モニター・エレメント 998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

rows_returned

詳しくは、モニター・エレメント 1468 ページの『rows_returned 戻り行数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

デッドロック

詳しくは、モニター・エレメント 978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

lock_timeouts

詳しくは、モニター・エレメント 1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

lock_escals

詳しくは、モニター・エレメント 1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_sends_total

詳しくは、モニター・エレメント 1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_recvs_total

詳しくは、モニター・エレメント 1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_send_volume

詳しくは、モニター・エレメント 1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_recv_volume

詳しくは、モニター・エレメント 1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_message_sends_total

詳しくは、モニター・エレメント 1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_message_recvs_total

詳しくは、モニター・エレメント 1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_message_send_volume

詳しくは、モニター・エレメント 1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_message_recv_volume

詳しくは、モニター・エレメント 1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_tq_sends_total

詳しくは、モニター・エレメント 1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_tq_recvs_total

詳しくは、モニター・エレメント 1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

fcm_tq_send_volume

詳しくは、モニター・エレメント 1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

fcm_tq_recv_volume

詳しくは、モニター・エレメント 1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム：モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

tq_tot_send_spills

詳しくは、モニター・エレメント 1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数：モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

post_threshold_sorts

詳しくは、モニター・エレメント 1418 ページの『post_threshold_sorts - ポストしきい値ソート：モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

post_shrthreshold_sorts

詳しくは、モニター・エレメント 1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート：モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

sort_overflows

詳しくは、モニター・エレメント 1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

audit_events_total

詳しくは、モニター・エレメント 857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_sorts

詳しくは、モニター・エレメント 1680 ページの『total_sorts - ソート合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

stmt_exec_time

詳しくは、モニター・エレメント 1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

coord_stmt_exec_time

詳しくは、モニター・エレメント 941 ページの『coord_stmt_exec_time - コーディネーター・エージェントのステートメントの実行時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_routine_non_sect_proc_time

詳しくは、モニター・エレメント 1657 ページの『total_routine_non_sect_proc_time - 非セクション処理時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_routine_non_sect_time

詳しくは、モニター・エレメント 1658 ページの『total_routine_non_sect_time - 非セクション・ルーチンの実行時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_section_proc_time

詳しくは、モニター・エレメント 1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_section_time

詳しくは、モニター・エレメント 1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_app_section_executions

詳しくは、モニター・エレメント 1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_routine_user_code_proc_time

詳しくは、モニター・エレメント 1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_routine_user_code_time

詳しくは、モニター・エレメント 1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_routine_time

詳しくは、モニター・エレメント 1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

thresh_violations

詳しくは、モニター・エレメント 1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

num_lw_thresh_exceeded

詳しくは、モニター・エレメント 1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_routine_invocations

詳しくは、モニター・エレメント 1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_wait_time_global

詳しくは、モニター・エレメント 1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_waits_global

詳しくは、モニター・エレメント 1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

reclaim_wait_time

詳しくは、モニター・エレメント 1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

spacemappage_reclaim_wait_time

詳しくは、モニター・エレメント 1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_timeouts_global

詳しくは、モニター・エレメント 1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_escals_maxlocks

詳しくは、モニター・エレメント 1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_escals_locklist

詳しくは、モニター・エレメント 1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

lock_escals_global

詳しくは、モニター・エレメント 1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

cf_wait_time

詳しくは、モニター・エレメント 887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

cf_waits

詳しくは、モニター・エレメント 886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_data_gbp_l_reads

詳しくは、モニター・エレメント 1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_data_gbp_p_reads

詳しくは、モニター・エレメント 1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_data_lbp_pages_found

詳しくは、モニター・エレメント 1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_data_gbp_invalid_pages

詳しくは、モニター・エレメント 1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_index_gbp_l_reads

詳しくは、モニター・エレメント 1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_index_gbp_p_reads

詳しくは、モニター・エレメント 1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_index_lbp_pages_found

詳しくは、モニター・エレメント 1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_index_gbp_invalid_pages

詳しくは、モニター・エレメント 1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_xda_gbp_l_reads

詳しくは、モニター・エレメント 1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_xda_gbp_p_reads

詳しくは、モニター・エレメント 1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_xda_lbp_pages_found

詳しくは、モニター・エレメント 1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_xda_gbp_invalid_pages

詳しくは、モニター・エレメント 1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

evmon_wait_time

詳しくは、モニター・エレメント 1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

evmon_waits_total

詳しくは、モニター・エレメント 1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_extended_latch_wait_time

詳しくは、モニター・エレメント 1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_extended_latch_waits

詳しくは、モニター・エレメント 1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_disp_run_queue_time

詳しくは、モニター・エレメント 1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_data_reqs

詳しくは、モニター・エレメント 1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_index_reqs

詳しくは、モニター・エレメント 1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_xda_reqs

詳しくは、モニター・エレメント 1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_temp_data_reqs

詳しくは、モニター・エレメント 1357 ページの『pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_temp_index_reqs

詳しくは、モニター・エレメント 1361 ページの『pool_queued_async_temp_index_reqs - TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_temp_xda_reqs

詳しくは、モニター・エレメント 1366 ページの『pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_other_reqs

詳しくは、モニター・エレメント 1353 ページの『pool_queued_async_other_reqs - プリフェッチャーが処理したその他の要求モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_data_pages

詳しくは、モニター・エレメント 1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_queued_async_index_pages

詳しくは、モニター・エレメント 1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_queued_async_xda_pages

詳しくは、モニター・エレメント 1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_queued_async_temp_data_pages

詳しくは、モニター・エレメント 1355 ページの『pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_queued_async_temp_index_pages

詳しくは、モニター・エレメント 1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_queued_async_temp_xda_pages

詳しくは、モニター・エレメント 1364 ページの『pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_failed_async_data_reqs

詳しくは、モニター・エレメント 1308 ページの『pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_failed_async_index_reqs

詳しくは、モニター・エレメント 1310 ページの『pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_failed_async_xda_reqs

詳しくは、モニター・エレメント 1322 ページの『pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_failed_async_temp_data_reqs

詳しくは、モニター・エレメント 1314 ページの『pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_failed_async_temp_index_reqs

詳しくは、モニター・エレメント 1317 ページの『pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_failed_async_temp_xda_reqs

詳しくは、モニター・エレメント 1319 ページの『pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

pool_failed_async_other_reqs

詳しくは、モニター・エレメント 1313 ページの『pool_failed_async_other_reqs - 失敗した非プリフェッチの要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_peds

詳しくは、モニター・エレメント 1647 ページの『total_peds - partial early distinct の合計回数のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

disabled_peds

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

post_threshold_peds

詳しくは、モニター・エレメント 1416 ページの『post_threshold_peds - partial early distinct しきい値のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

total_peas

詳しくは、モニター・エレメント 1645 ページの『total_peas - partial early aggregation の合計回数のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

post_threshold_peas

詳しくは、モニター・エレメント 1413 ページの『post_threshold_peas - partial early aggregation しきい値のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tq_sort_heap_requests

詳しくは、モニター・エレメント 1703 ページの『tq_sort_heap_requests - 表キュー・ソート・ヒープ要求のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

tq_sort_heap_rejections

詳しくは、モニター・エレメント 1700 ページの『tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否のモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

prefetch_wait_time

詳しくは、モニター・エレメント 1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

prefetch_waits

詳しくは、モニター・エレメント 1422 ページの『prefetch_waits - プリフェッチャーの待機カウントのモニター・エレメント』 を参照してください。

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_data_gbp_indep_pages_found_in_lbp

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_index_gbp_indep_pages_found_in_lbp

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント：

タイプ	ファセット
xs:long	

pool_xda_gbp_indep_pages_found_in_lbp

これを含むエレメント: 374 ページの『system_metrics』、 377 ページの『activity_metrics』

エレメント・コンテント :

タイプ	ファセット
xs:long	

データベース・イベント・モニター

データベース・イベント・モニターによって生成されるデータ

データベース・イベント・モニターは、データベース・レベルのカウンターに関するデータを生成します。データベース・イベント・モニターから通常の表、ファイル、またはパイプに出力するように選択することができます。

選択した出力形式にかかわらず、すべてのデータベース・イベント・データは、次の 2 つの論理グループのいずれかから取得されます。

- 70 ページの『event_db 論理データ・グループ』
- 75 ページの『event_dbmemuse 論理データ・グループ』

また、データベース・イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

データベース・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、データベース・イベント・モニターによって書き込まれる情報。

データベース・イベント・モニターの出力タイプとして WRITE TO TABLE を選択した場合、デフォルトでは、3 つの表が生成されます。各表には、1 つ以上の論理データ・グループのモニター・エレメントが入っています。

表 62. DATABASE 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
DB_*evmon-name*	event_db
DBMEMUSE_*evmon-name*	event_dbmemuse
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

生成される表

表 63. データベース・イベント・モニターに戻される情報 : デフォルトの表名: DB_evmon-name

列名	データ・タイプ	説明
APPL_SECTION_INSERTS	BIGINT	appl_section_inserts セクション挿入数
APPL_SECTION_LOOKUPS	BIGINT	appl_section_lookups - セクション検索
ASYNC_RUNSTATS	BIGINT	async_runstats - 非同期 RUNSTATS 要求の合計数
BINDS_PRECOMPILES	BIGINT	binds_precompiles 試行されたバインド/プリコンパイル
BLOCKS_PENDING_CLEANUP	BIGINT	blocks_pending_cleanup - クリーンアップ保留中のロールアウト済みブロック
CAT_CACHE_HEAP_FULL	BIGINT	cat_cache_heap_full - カタログ・キャッシュ・ヒープ・フル
CAT_CACHE_INSERTS	BIGINT	cat_cache_inserts カタログ・キャッシュ挿入数
CAT_CACHE_LOOKUPS	BIGINT	cat_cache_lookups カタログ・キャッシュ検索
CAT_CACHE_OVERFLOWS	BIGINT	cat_cache_overflows カタログ・キャッシュ・オーバーフロー数
CAT_CACHE_SIZE_TOP	BIGINT	cat_cache_size_top - カタログ・キャッシュの最高水準点
CATALOG_NODE	BIGINT	catalog_node カタログ・ノード番号
CATALOG_NODE_NAME	VARCHAR(32)	catalog_node_name カタログ・ノード・ネットワーク名
COMMIT_SQL_STMTS	BIGINT	commit_sql_stmts 試行されたコミット・ステートメント
CONNECTIONS_TOP	BIGINT	connections_top 同時接続の最大数
DB_HEAP_TOP	BIGINT	db_heap_top 割り振られた最大データベース・ヒープ

表 63. データベース・イベント・モニターに戻される情報 : デフォルトの表名: DB_evmon-name (続き)

列名	データ・タイプ	説明
DDL_SQL_STMTS	BIGINT	ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント
DEADLOCKS	BIGINT	deadlocks デッドロック検出数
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
DISCONN_TIME	TIMESTAMP	disconn_time データベース非アクティブ化タイム・スタンプ
DYNAMIC_SQL_STMTS	BIGINT	dynamic_sql_stmts 試行された動的 SQL ステートメント
ELAPSED_EXEC_TIME	BIGINT	elapsed_exec_time ステートメント実行経過時間
EVMON_ACTIVATES	BIGINT	evmon_activates イベント・モニター活動化回数
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数
FAILED_SQL_STMTS	BIGINT	failed_sql_stmts 失敗したステートメント操作
FILES_CLOSED	BIGINT	files_closed - クローズしたデータベース・ファイル
HASH_JOIN_OVERFLOWS	BIGINT	hash_join_overflows ハッシュ結合のオーバーフロー
HASH_JOIN_SMALL_OVERFLOWS	BIGINT	hash_join_small_overflows ハッシュ結合の短精度オーバーフロー
INT_AUTO_REBINDS	BIGINT	int_auto_rebinds 内部自動再バインド
INT_COMMITS	BIGINT	int_commits 内部コミット数
INT_ROLLBACKS	BIGINT	int_rollbacks 内部ロールバック数
INT_ROWS_DELETED	BIGINT	int_rows_deleted 削除された内部行数

表 63. データベース・イベント・モニターに戻される情報 : デフォルトの表名: DB_evmon-name (続き)

列名	データ・タイプ	説明
INT_ROWS_INSERTED	BIGINT	int_rows_inserted 挿入された内部行数
INT_ROWS_UPDATED	BIGINT	int_rows_updated 更新された内部行数
LOCK_ESCALS	BIGINT	lock_escals ロック・エスカレーション数
LOCK_TIMEOUTS	BIGINT	lock_timeouts ロック・タイムアウト数
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCK_WAITS	BIGINT	lock_waits ロック待機数
LOG_FILE_ARCHIVE	BIGINT	
LOG_FILE_NUM_CURR	BIGINT	
LOG_FILE_NUM_FIRST	BIGINT	
LOG_FILE_NUM_LAST	BIGINT	
LOG_HELD_BY_DIRTY_PAGES	BIGINT	log_held_by_dirty_pages ダーティー・ページによって占有されるログ・スペースの量
LOG_READ_TIME	BIGINT	log_read_time ログ読み取り時間
LOG_READS	BIGINT	log_reads 読み取られたログ・ページの数
LOG_TO_REDO_FOR_RECOVERY	BIGINT	log_to_redo_for_recovery リカバリーの場合に再実行されるログの量
LOG_WRITE_TIME	BIGINT	log_write_time ログ書き込み時間
LOG_WRITES	BIGINT	log_writes 書き込まれたログ・ページの数
NUM_LOG_BUFF_FULL	BIGINT	
NUM_LOG_DATA_IN_BUFF	BIGINT	
NUM_LOG_PART_PAGE_IO	BIGINT	num_log_part_page_io 部分ログ・ページ書き込み数
NUM_LOG_READ_IO	BIGINT	num_log_read_io ログ読み取り数
NUM_LOG_WRITE_IO	BIGINT	num_log_write_io ログ書き込み数
NUM_THRESHOLD_VIOLATIONS	INTEGER	num_threshold_violations しきい値違反の回数
OLAP_FUNC_OVERFLOWS	BIGINT	olap_func_overflows OLAP 関数のオーバーフロー
PARTIAL_RECORD	SMALLINT	partial_record 部分レコード

表 63. データベース・イベント・モニターに戻される情報 : デフォルトの表名: DB_evmon-name (続き)

列名	データ・タイプ	説明
PKG_CACHE_INSERTS	BIGINT	pkg_cache_inserts パッケージ・キャッシュ挿入
PKG_CACHE_LOOKUPS	BIGINT	pkg_cache_lookups パッケージ・キャッシュ検索
PKG_CACHE_NUM_OVERFLOWS	BIGINT	pkg_cache_num_overflows パッケージ・キャッシュ・オーバーフロー
PKG_CACHE_SIZE_TOP	BIGINT	pkg_cache_size_top パッケージ・キャッシュの最高水準点
POOL_ASYNC_DATA_READ_REQS	BIGINT	pool_async_data_read_reqs バッファー・プール非同期読み取り要求
POOL_ASYNC_DATA_READS	BIGINT	pool_async_data_reads バッファー・プール非同期データ読み取り
POOL_ASYNC_DATA_WRITES	BIGINT	pool_async_data_writes バッファー・プール非同期データ書き込み
POOL_ASYNC_INDEX_READ_REQS	BIGINT	pool_async_index_read_reqs バッファー・プール非同期索引読み取り要求
POOL_ASYNC_INDEX_READS	BIGINT	pool_async_index_reads バッファー・プール非同期索引読み取り
POOL_ASYNC_INDEX_WRITES	BIGINT	pool_async_index_writes バッファー・プール非同期索引書き込み
POOL_ASYNC_READ_TIME	BIGINT	pool_async_read_time バッファー・プール非同期読み取り時間
POOL_ASYNC_WRITE_TIME	BIGINT	pool_async_write_time バッファー・プール非同期書き込み時間
POOL_ASYNC_XDA_READ_REQS	BIGINT	pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求
POOL_ASYNC_XDA_READS	BIGINT	pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り
POOL_ASYNC_XDA_WRITES	BIGINT	pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み

表 63. データベース・イベント・モニターに戻される情報 : デフォルトの表名: DB_evmon-name (続き)

列名	データ・タイプ	説明
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み
POOL_DRTY_PG_STEAL_CLNS	BIGINT	pool_drty_pg_steal_clns 起動されたバッファー・プール・ビクティム・ページ・クリーナー
POOL_DRTY_PG_THRSH_CLNS	BIGINT	pool_drty_pg_thrsh_clns 起動されたバッファー・プールしきい値クリーナー
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
POOL_LSN_GAP_CLNS	BIGINT	pool_lsn_gap_clns 起動されたバッファー・プール・ログ・スペース・クリーナー
POOL_NO_VICTIM_BUFFER	BIGINT	pool_no_victim_buffer バッファー・プールの非ビクティム・バッファー数
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り

表 63. データベース・イベント・モニターに戻される情報 : デフォルトの表名: DB_evmon-name (続き)

列名	データ・タイプ	説明
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
POST_SHRTHRESHOLD_HASH_JOINS	BIGINT	post_shrthreshold_hash_joins ポストしきい値ハッシュ結合
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts ポスト共有しきい値ソート
PREFETCH_WAIT_TIME	BIGINT	prefetch_wait_time プリフェッチ待ち時間
ROLLBACK_SQL_STMTS	BIGINT	rollback_sql_stmts - 試行されたロールバック・ステートメント
ROWS_DELETED	BIGINT	rows_deleted 削除行数
ROWS_INSERTED	BIGINT	rows_inserted 挿入行数
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_SELECTED	BIGINT	rows_selected 選択行数
ROWS_UPDATED	BIGINT	rows_updated 更新行数
SEC_LOG_USED_TOP	BIGINT	sec_log_used_top 使用された最大 2 次ログ・スペース
SELECT_SQL_STMTS	BIGINT	select_sql_stmts 実行された選択 SQL ステートメント
SERVER_PLATFORM	INTEGER	server_platform サーバーのオペレーティング・システム
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
SORT_SHRHEAP_TOP	BIGINT	sort_shrheap_top ソート共有ヒープの最高水準点

表 63. データベース・イベント・モニターに戻される情報 : デフォルトの表名: DB_evmon-name (続き)

列名	データ・タイプ	説明
STATIC_SQL_STMTS	BIGINT	static_sql_stmts 試行された静的 SQL ステートメント
STATS_CACHE_SIZE	BIGINT	stats_cache_size - 統計キャッシュのサイズ
STATS_FABRICATE_TIME	BIGINT	stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間
STATS_FABRICATIONS	BIGINT	stats_fabrications - 統計作成の合計数
SYNC_RUNSTATS	BIGINT	sync_runstats - 同期 RUNSTATS アクティビティーの合計数
SYNC_RUNSTATS_TIME	BIGINT	1552 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント』
TOT_LOG_USED_TOP	BIGINT	tot_log_used_top 使用された最大合計ログ・スペース
TOTAL_CONS	BIGINT	total_cons データベース活動化以降の接続
TOTAL_HASH_JOINS	BIGINT	total_hash_joins ハッシュ結合の合計
TOTAL_HASH_LOOPS	BIGINT	total_hash_loops ハッシュ・ループの合計
TOTAL_OLAP_FUNCS	BIGINT	total_olap_funcs OLAP 関数の合計数
TOTAL_SORT_TIME	BIGINT	total_sort_time ソート時間合計
TOTAL_SORTS	BIGINT	total_sorts ソート合計
UID_SQL_STMTS	BIGINT	uid_sql_stmts 実行された UPDATE/INSERT/DELETE SQL ステートメント
UNREAD_PREFETCH_PAGES	BIGINT	unread_prefetch_pages 読み取り不能プリフェッチ・ページ
X_LOCK_ESCALS	BIGINT	x_lock_escals 排他ロック・エスカレーション数
XQUERY_STMTS	BIGINT	xquery_stmts - 試行された XQuery ステートメント

表 64. データベース・イベント・モニターに戻される情報 : デフォルトの表名: DBMEMUSE_evmon-name

列名	データ・タイプ	説明
EVMON_ACTIVATES	BIGINT	evmon_activates イベント・モニター活動化回数
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数
POOL_CUR_SIZE	BIGINT	pool_cur_size メモリー・プールの現行サイズ
POOL_ID	BIGINT	pool_id メモリー・プール ID
POOL_MAX_SIZE	BIGINT	
POOL_SECONDARY_ID	CHARACTER(32)	pool_secondary_id メモリー・プール 2 次 ID
POOL_WATERMARK	BIGINT	pool_watermark メモリー・プール水準点

表 65. データベース・イベント・モニターに戻される情報 : デフォルトの表名: CONTROL_evmon-name

列名	データ・タイプ	説明
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ

しきい値違反イベント・モニター

しきい値違反イベント・モニターによって生成されるデータ

しきい値違反イベント・モニターは、しきい値違反に関するデータを生成します。データベース・イベント・モニターから通常の表、ファイル、またはパイプに出力するように選択することができます。

選択した出力形式にかかわらず、すべてのしきい値違反イベント・データは、論理データ・グループ event_thresholdviolations から取得されます。また、イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

しきい値違反イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、しきい値違反イベント・モニターによって書き込まれる情報。

以降のセクションでは、 WRITE TO TABLE オプションを CREATE EVENT MONITOR ステートメントに使用した場合のしきい値違反イベント・モニターの出

力について説明します。イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

表 66. THRESHOLD 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
THRESHOLDVIOLATIONS_*evmon-name*	event_thresholdviolations
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 67. しきい値違反イベント・モニターに戻される情報 : デフォルトの表名: THRESHOLDVIOLATIONS_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
ACTIVATE_TIMESTAMP	TIMESTAMP	activate_timestamp タイム・スタンプの活動化
ACTIVITY_COLLECTED	CHARACTER(1)	activity_collected 収集されたアクティビティー
ACTIVITY_ID	BIGINT	activity_id アクティビティー ID
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID)
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
COORD_PARTITION_NUM	INTEGER	coord_partition_num コーディネーター・パーティション番号
DESTINATION_SERVICE_CLASS_ID	INTEGER	982 ページの『destination_service_class_id - 宛先サービス・クラス ID : モニター・エレメント』
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号
SOURCE_SERVICE_CLASS_ID	INTEGER	source_service_class_id ソース・サービス・クラス ID
THRESHOLD_ACTION	VARCHAR(16)	threshold_action しきい値アクション
THRESHOLD_MAXVALUE	BIGINT	threshold_maxvalue しきい値最大値
THRESHOLD_PREDICATE	VARCHAR(64)	threshold_predicate しきい値述部
THRESHOLD_QUEUESIZE	BIGINT	threshold_queuesize しきい値キュー・サイズ
THRESHOLDID	INTEGER	thresholdid しきい値 ID
TIME_OF_VIOLATION	TIMESTAMP	time_of_violation 違反時刻
UOW_ID	INTEGER	uow_id 作業単位 ID

表 68. しきい値違反イベント・モニターに戻される情報 : 表名: CONTROL_evmon-name

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	1259 ページの『partition_key - パーティション・キーのモニター・エレメント』
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ
PARTITION_NUMBER	SMALLINT	partition_number パーティション番号

ステートメント・イベント・モニター

ステートメント・イベント・モニターによって生成されるデータ

ステートメント・イベント・モニターは、システムで実行されるステートメントに関するデータを生成します。データベース・イベント・モニターから通常の表、ファイル、またはパイプに出力するように選択することができます。

選択した出力形式にかかわらず、すべてのステートメント・イベント・データは、次の 3 つの論理グループのいずれかから取得されます。

- 91 ページの『event_stmt 論理データ・グループ』
- 68 ページの『event_connheader 論理データ・グループ』
- 93 ページの『event_subsection 論理データ・グループ』

また、ステートメント・イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

ステートメント・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、ステートメント・イベント・モニターによって書き込まれる情報。

以降のセクションでは、 WRITE TO TABLE オプションを CREATE EVENT MONITOR ステートメントに使用した場合のステートメント・イベント・モニターの出力について説明します。イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

表 69. STATEMENT 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
STMT_*evmon-name*	91 ページの『event_stmt 論理データ・グループ』

表 69. STATEMENT 表書き込みイベント・モニターによって生成される表 (続き): 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
CONNHEADER_*evmon-name*	68 ページの『event_connheader 論理データ・グループ』
CONNHEADER_*evmon-name*	68 ページの『event_connheader 論理データ・グループ』
SUBSECTION_*evmon-name*	93 ページの『event_subsection 論理データ・グループ』 (パーティション・データベース環境の場合にのみ生成されます)
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 70. ステートメント・イベント・モニターに戻される情報 : デフォルトの表名: STMT_evmon-name

列名	データ・タイプ	説明
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID)
AGENTS_TOP	BIGINT	agents_top 作成されたエージェントの数
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
BLOCKING_CURSOR	SMALLINT	blocking_cursor ブロック・カーソル
CONSISTENCY_TOKEN	CHARACTER	consistency_token パッケージ整合性トークン
CREATOR	VARCHAR(128)	creator アプリケーション作成者
CURSOR_NAME	VARCHAR(18)	cursor_name カーソル名
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数
FETCH_COUNT	BIGINT	fetch_count 成功したフェッチの数
INT_ROWS_DELETED	BIGINT	int_rows_deleted 削除された内部行数

表 70. ステートメント・イベント・モニターに戻される情報 : デフォルトの表名: STMT_evmon-name (続き)

列名	データ・タイプ	説明
INT_ROWS_INSERTED	BIGINT	int_rows_inserted 挿入された内部行数
INT_ROWS_UPDATED	BIGINT	int_rows_updated 更新された内部行数
PACKAGE_NAME	VARCHAR(128)	package_name パッケージ名
PACKAGE_VERSION_ID	VARCHAR(64)	package_version_id パッケージ・バージョン
PARTIAL_RECORD	SMALLINT	partial_record 部分レコード
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り

表 70. ステートメント・イベント・モニターに戻される情報 : デフォルトの表名: STMT_evmon-name (続き)

列名	データ・タイプ	説明
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_WRITTEN	BIGINT	rows_written 書き込み行数
SECTION_NUMBER	BIGINT	section_number セクション番号
SEQUENCE_NO	CHARACTER	sequence_no シーケンス番号
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
SQL_REQ_ID	BIGINT	sql_req_id SQL ステートメントの要求 ID
SQLCABC	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLCAID	CHARACTER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLCODE	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD1	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD2	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD3	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD4	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD5	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRD6	INTEGER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRM	VARCHAR(72)	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLERRP	CHARACTER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。

表 70. ステートメント・イベント・モニターに戻される情報 : デフォルトの表名: STMT_evmon-name (続き)

列名	データ・タイプ	説明
SQLSTATE	CHARACTER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
SQLWARN	CHARACTER	「*SQL リファレンス 第 1 巻*」の『SQLCA (SQL 連絡域)』を参照。
START_TIME	TIMESTAMP	start_time イベント開始時刻
STATS_FABRICATE_TIME	BIGINT	stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間
STMT_OPERATION	BIGINT	
STMT_TYPE	BIGINT	stmt_type ステートメント・タイプ
STOP_TIME	TIMESTAMP	stop_time イベント停止時刻
SYNC_RUNSTATS_TIME	BIGINT	1552 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント』
SYSTEM_CPU_TIME	BIGINT	system_cpu_time システム CPU 時間
TOTAL_SORT_TIME	BIGINT	total_sort_time ソート時間合計
TOTAL_SORTS	BIGINT	total_sorts ソート合計
USER_CPU_TIME	BIGINT	user_cpu_time ユーザー CPU 時間
STMT_TEXT	CLOB(2097152)	stmt_text SQL ステートメント・テキスト

表 71. ステートメント・イベント・モニターに戻される情報 : デフォルトの表名: CONNHEADER_evmon-name

列名	データ・タイプ	説明
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID)
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
APPL_NAME	VARCHAR(255)	appl_name アプリケーション名
AUTH_ID	VARCHAR(128)	auth_id 許可 ID
CLIENT_DB_ALIAS	CHARACTER	client_db_alias アプリケーションで使用するデータベース別名

表 71. ステートメント・イベント・モニターに戻される情報 : デフォルトの表名: CONNHEADER_evmon-name (続き)

列名	データ・タイプ	説明
CLIENT_NNAME	VARCHAR(20)	898 ページの『client_nname - クライアント名モニター・エレメント』
CLIENT_PID	BIGINT	client_pid クライアント・プロセス ID
CLIENT_PLATFORM	INTEGER	client_platform クライアント・オペレーティング・プラットフォーム
CLIENT_PRDID	VARCHAR(20)	client_prdid クライアント製品およびバージョン ID
CLIENT_PROTOCOL	INTEGER	client_protocol クライアント通信プロトコル
CODEPAGE_ID	INTEGER	codepage_id アプリケーションで使用するコード・ページ ID
CONN_TIME	TIMESTAMP	conn_time データベース接続時刻
CORR_TOKEN	VARCHAR(64)	corr_token - DRDA® 相関トークン
EXECUTION_ID	VARCHAR(128)	execution_id ユーザー・ログイン ID
SEQUENCE_NO	CHARACTER	sequence_no シーケンス番号
TERRITORY_CODE	INTEGER	territory_code データベース・テリトリー・コード

表 72. ステートメント・イベント・モニターに戻される情報 : デフォルトの表名: CONTROL_evmon-name

列名	データ・タイプ	説明
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ

表イベント・モニター

表イベント・モニターによって生成されるデータ

表イベント・モニターは、データベースの表に関する集約メトリックを生成します。データベース・イベント・モニターから通常の表、ファイル、またはパイプに出力するように選択することができます。

注: 特定の表オブジェクトに関するより詳細な使用情報が必要な場合は、そのオブジェクトに使用リストを作成することを検討してください。

選択した出力形式にかかわらず、すべての表イベント・データは、論理データ・グループ event_table から取得されます。また、ステートメント・イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

表イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、表イベント・モニターによって書き込まれる情報。

以降のセクションでは、 WRITE TO TABLE オプションを CREATE EVENT MONITOR ステートメントに使用した場合の表イベント・モニターの出力について説明します。イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

表 73. TABLE 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
TABLE_*evmon-name*	event_table
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 74. 表イベント・モニターに戻される情報 : デフォルトの表名: TABLE_evmon-name

列名	データ・タイプ	説明
DATA_OBJECT_PAGES	BIGINT	data_object_pages データ・オブジェクト・ページ数
DATA_PARTITION_ID	INTEGER	data_partition_id - データ・パーティション ID
EVENT_TIME	TIMESTAMP	event_time イベント時刻
EVMON_ACTIVATES	BIGINT	evmon_activates イベント・モニター活動化回数
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数

表 74. 表イベント・モニターに戻される情報 : デフォルトの表名: TABLE_evmon-name (続き)

列名	データ・タイプ	説明
INDEX_OBJECT_PAGES	BIGINT	index_object_pages 索引オブジェクト・ページ数
LOB_OBJECT_PAGES	BIGINT	lob_object_pages LOB オブジェクト・ページ数
LONG_OBJECT_PAGES	BIGINT	long_object_pages 長いオブジェクト・ページ数
OVERFLOW_ACCESSES	BIGINT	overflow_accesses オーバーフロー・レコードへのアクセス
PAGE_REORGS	BIGINT	page_reorgs ページ再編成
PARTIAL_RECORD	SMALLINT	partial_record 部分レコード
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_WRITTEN	BIGINT	rows_written 書き込み行数
TABLE_NAME	VARCHAR(128)	table_name - デフォルトの表名
TABLE_SCHEMA	VARCHAR(128)	table_schema 表スキーマ名
TABLE_TYPE	BIGINT	table_type 表タイプ
TABLESPACE_ID	BIGINT	tablespace_id 表スペース ID
XDA_OBJECT_PAGES	BIGINT	xda_object_pages XDA オブジェクト・ページ数

表 75. 表イベント・モニターに戻される情報 : デフォルトの表名: CONTROL_evmon-name

列名	データ・タイプ	説明
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ

バッファー・プール・イベント・モニター

バッファー・プール・イベント・モニターによって生成されるデータ

バッファー・プール・イベント・モニターは、バッファー・プール・アクティビティーに関する集約メトリックを生成します。データベース・イベント・モニターから通常の表、ファイル、またはパイプに出力するように選択することができます。

選択した出力形式にかかわらず、すべてのバッファー・プール・イベント・データは、論理データ・グループ event_bufferpool から取得されます。また、ステートメント・イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

バッファー・プール・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、バッファー・プール・イベント・モニターによって書き込まれる情報。

以降のセクションでは、 WRITE TO TABLE オプションを CREATE EVENT MONITOR ステートメントに使用した場合のバッファー・プール・イベント・モニターの出力について説明します。イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

表 76. BUFFERPOOL 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
BUFFERPOOL_*evmon-name*	event_bufferpool
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 77. バッファー・プール・イベント・モニターに戻される情報 : デフォルトの表名: BUFFERPOOL_evmon-name

列名	データ・タイプ	説明
BLOCK_IOS	BIGINT	block_ios ブロック入出力要求数
BP_NAME	VARCHAR(20)	bp_name バッファー・プール名
DB_NAME	CHARACTER(8)	db_name データベース名
DB_PATH	VARCHAR(215)	db_path データベース・パス
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
EVENT_TIME	TIMESTAMP	event_time イベント時刻
EVMON_ACTIVATES	BIGINT	evmon_activates イベント・モニター活動化回数
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数
FILES_CLOSED	BIGINT	files_closed - クローズしたデータベース・ファイル

表 77. バッファー・プール・イベント・モニターに戻される情報 : デフォルトの表名: BUFFERPOOL_evmon-name (続き)

列名	データ・タイプ	説明
PAGES_FROM_BLOCK_IOS	BIGINT	pages_from_block_ios ブロック入出力によって読み取られたページ数の合計
PAGES_FROM_VECTORED_IOS	BIGINT	pages_from_vectored_ios ベクトル化入出力によって読み取られたページ数の合計
PARTIAL_RECORD	SMALLINT	partial_record 部分レコード
POOL_ASYNC_DATA_READ_REQS	BIGINT	pool_async_data_read_reqs バッファー・プール非同期読み取り要求
POOL_ASYNC_DATA_READS	BIGINT	pool_async_data_reads バッファー・プール非同期データ読み取り
POOL_ASYNC_DATA_WRITES	BIGINT	pool_async_data_writes バッファー・プール非同期データ書き込み
POOL_ASYNC_INDEX_READ_REQS	BIGINT	pool_async_index_read_reqs バッファー・プール非同期索引読み取り要求
POOL_ASYNC_INDEX_READS	BIGINT	pool_async_index_reads バッファー・プール非同期索引読み取り
POOL_ASYNC_INDEX_WRITES	BIGINT	pool_async_index_writes バッファー・プール非同期索引書き込み
POOL_ASYNC_READ_TIME	BIGINT	pool_async_read_time バッファー・プール非同期読み取り時間
POOL_ASYNC_WRITE_TIME	BIGINT	pool_async_write_time バッファー・プール非同期書き込み時間
POOL_ASYNC_XDA_READ_REQS	BIGINT	pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求
POOL_ASYNC_XDA_READS	BIGINT	pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り
POOL_ASYNC_XDA_WRITES	BIGINT	pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
POOL_NO_VICTIM_BUFFER	BIGINT	pool_no_victim_buffer バッファー・プールの非ビクティム・バッファー数
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計

表 77. バッファー・プール・イベント・モニターに戻される情報 : デフォルトの表名: *BUFFERPOOL_evmon-name (続き)*

列名	データ・タイプ	説明
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
UNREAD_PREFETCH_PAGES	BIGINT	unread_prefetch_pages 読み取り不能プリフェッチ・ページ
VECTORED_IOS	BIGINT	vectored_ios ベクトル化入出力要求数

表 78. バッファー・プール・イベント・モニターに戻される情報 : デフォルトの表名: *CONTROL_evmon-name*

列名	データ・タイプ	説明
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ

表スペース・イベント・モニター

表スペース・イベント・モニターによって生成されるデータ

表スペース・イベント・モニターは、データベースの表スペースに関する集約メトリックを生成します。データベース・イベント・モニターから通常の表、ファイル、またはパイプに出力するように選択することができます。

選択した出力形式にかかわらず、すべての表スペース・イベント・データは、論理データ・グループ event_tablespace から取得されます。また、ステートメント・イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

表スペース・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、表スペース・イベント・モニターによって書き込まれる情報。

以降のセクションでは、 WRITE TO TABLE オプションを CREATE EVENT MONITOR ステートメントに使用した場合の表スペース・イベント・モニターの出力について説明します。イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

表 79. TABLESPACE 表書き込みイベント・モニターによって生成される表： 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
TABLESPACE_*evmon-name*	event_tablespace
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

生成される表

表 80. 表スペース・イベント・モニターに戻される情報 : デフォルトの表名: TABLESPACE_evmon-name

列名	データ・タイプ	説明
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
EVENT_TIME	TIMESTAMP	event_time イベント時刻
EVMON_ACTIVATES	BIGINT	evmon_activates イベント・モニター活動化回数
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数
FILES_CLOSED	BIGINT	files_closed - クローズしたデータベース・ファイル
PARTIAL_RECORD	SMALLINT	partial_record 部分レコード
POOL_ASYNC_DATA_READ_REQS	BIGINT	pool_async_data_read_reqs バッファー・プール非同期読み取り要求
POOL_ASYNC_DATA_READS	BIGINT	pool_async_data_reads バッファー・プール非同期データ読み取り
POOL_ASYNC_DATA_WRITES	BIGINT	pool_async_data_writes バッファー・プール非同期データ書き込み
POOL_ASYNC_INDEX_READ_REQS	BIGINT	pool_async_index_read_reqs バッファー・プール非同期索引読み取り要求
POOL_ASYNC_INDEX_READS	BIGINT	pool_async_index_reads バッファー・プール非同期索引読み取り
POOL_ASYNC_INDEX_WRITES	BIGINT	pool_async_index_writes バッファー・プール非同期索引書き込み

表 80. 表スペース・イベント・モニターに戻される情報 : デフォルトの表名: *TABLESPACE_evmon-name (続き)*

列名	データ・タイプ	説明
POOL_ASYNC_READ_TIME	BIGINT	pool_async_read_time バッファー・プール非同期読み取り時間
POOL_ASYNC_WRITE_TIME	BIGINT	pool_async_write_time バッファー・プール非同期書き込み時間
POOL_ASYNC_XDA_READ_REQS	BIGINT	pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求
POOL_ASYNC_XDA_READS	BIGINT	pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り
POOL_ASYNC_XDA_WRITES	BIGINT	pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
POOL_NO_VICTIM_BUFFER	BIGINT	pool_no_victim_buffer バッファー・プールの非ビクティム・バッファー数
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
TABLESPACE_FS_CACHING	SMALLINT	fs_caching ファイル・システム・キャッシング
TABLESPACE_NAME	VARCHAR(18)	tablespace_name 表スペース名
UNREAD_PREFETCH_PAGES	BIGINT	unread_prefetch_pages 読み取り不能プリフェッチ・ページ

表 81. 表スペース・イベント・モニターに戻される情報 : デフォルトの表名: *CONTROL_evmon-name*

列名	データ・タイプ	説明
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名

表 81. 表スペース・イベント・モニターに戻される情報 : デフォルトの表名: CONTROL_evmon-name (続き)

列名	データ・タイプ	説明
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ

接続イベント・モニター

接続イベント・モニターによって生成されるデータ

接続イベント・モニターは、アプリケーションによるデータベースへの接続ごとに、メトリックおよびその他のモニター・エレメントをキャプチャーします。出力の書き込み先は、ファイル、名前付きパイプ、または通常の表から選択することができます。

選択した出力形式にかかわらず、すべての接続イベント・データは、次の 3 つの論理グループのいずれかから取得されます。

- event_connheader
- event_conn
- event_connmemuse

また、接続イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

接続イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、接続イベント・モニターによって書き込まれる情報。

接続イベント・モニターの出力タイプとして WRITE TO TABLE を選択した場合、デフォルトでは、4 つの表が生成されます。各表には、1 つ以上の論理データ・グループのモニター・エレメントが入っています。

表 82. CONNECTIONS 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
CONNHEADER_*evmon-name*	event_connheader
CONN_*evmon-name*	event_conn
CONMEMUSE_*evmon-name*	event_connmemuse
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

特定の表にイベント・モニターから出力されるように制限するには、CREATE EVENT MONITOR または ALTER EVENT MONITOR ステートメントで表を作成するための論理グループ名を指定します。詳しくは、これらのステートメントの参照トピックをご覧ください。

イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

生成される表

表 83. 接続イベント・モニターに戻される情報 : デフォルトの表名: CONNHEADER_evmon-name

列名	データ・タイプ	説明
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID)
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
APPL_NAME	VARCHAR(255)	appl_name アプリケーション名
AUTH_ID	VARCHAR(128)	auth_id 許可 ID
CLIENT_DB_ALIAS	CHAR(8)	client_db_alias アプリケーションで使用するデータベース別名
CLIENT_NNAME	VARCHAR(20)	898 ページの『client_nname - クライアント名モニター・エレメント』
CLIENT_PID	BIGINT	client_pid クライアント・プロセス ID
CLIENT_PLATFORM	INTEGER	client_platform クライアント・オペレーティング・プラットフォーム
CLIENT_PRDID	VARCHAR(20)	client_prdid クライアント製品およびバージョン ID
CLIENT_PROTOCOL	INTEGER	client_protocol クライアント通信プロトコル
CODEPAGE_ID	INTEGER	codepage_id アプリケーションで使用するコード・ページ ID
CONN_TIME	TIMESTAMP	conn_time データベース接続時刻
CORR_TOKEN	VARCHAR(64)	corr_token - DRDA 相関トークン
EXECUTION_ID	VARCHAR(128)	execution_id ユーザー・ログイン ID
SEQUENCE_NO	CHAR(5)	sequence_no シーケンス番号
TERRITORY_CODE	INTEGER	territory_code データベース・テリトリー・コード

表 84. 接続イベント・モニターに戻される情報: 表名 : CONN_evmon-name

列名	データ・タイプ	説明
ACC_CURS_BLK	BIGINT	acc_curs_blk 受け入れられたブロック・カーソル要求
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID)
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID

表 84. 接続イベント・モニターに戻される情報: 表名 : CONN_evmon-name (続き)

列名	データ・タイプ	説明
APPL_PRIORITY	BIGINT	appl_priority アプリケーション・エージェント優先順位
APPL_PRIORITY_TYPE	BIGINT	appl_priority_type アプリケーション優先順位タイプ
APPL_SECTION_INSERTS	BIGINT	appl_section_inserts セクション挿入数
APPL_SECTION_LOOKUPS	BIGINT	appl_section_lookups - セクション検索
APPL_STATUS	BIGINT	appl_status アプリケーション状況
AUTHORITY_BITMAP	CHARACTER(22)	authority_bitmap ユーザー許可レベル
AUTHORITY_LVL	BIGINT	authority_lvl ユーザー許可レベル
BINDS_PRECOMPILES	BIGINT	binds_precompiles 試行されたバインド/プリコンパイル
CAT_CACHE_HEAP_FULL	BIGINT	cat_cache_heap_full - カタログ・キャッシュ・ヒープ・フル
CAT_CACHE_INSERTS	BIGINT	cat_cache_inserts カタログ・キャッシュ挿入数
CAT_CACHE_LOOKUPS	BIGINT	cat_cache_lookups カタログ・キャッシュ検索
CAT_CACHE_OVERFLOWS	BIGINT	cat_cache_overflows カタログ・キャッシュ・オーバーフロー数
CAT_CACHE_SIZE_TOP	BIGINT	cat_cache_size_top - カタログ・キャッシュの最高水準点
COMMIT_SQL_STMTS	BIGINT	commit_sql_stmts 試行されたコミット・ステートメント
COORD_NODE	BIGINT	coord_node コーディネーター・ノード
DDL_SQL_STMTS	BIGINT	ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント
DEADLOCKS	BIGINT	deadlocks デッドロック検出数
DIRECT_READ_REQS	BIGINT	direct_read_reqs 直接読み取り要求
DIRECT_READ_TIME	BIGINT	direct_read_time 直接読み取り時間
DIRECT_READS	BIGINT	direct_reads データベースからの直接読み取り
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs 直接書き込み要求
DIRECT_WRITE_TIME	BIGINT	direct_write_time 直接書き込み時間
DIRECT_WRITES	BIGINT	direct_writes データベースへの直接書き込み
DISCONN_TIME	TIMESTAMP	disconn_time データベース非アクティブ化タイム・スタンプ
DYNAMIC_SQL_STMTS	BIGINT	dynamic_sql_stmts 試行された動的 SQL ステートメント
ELAPSED_EXEC_TIME	BIGINT	elapsed_exec_time ステートメント実行経過時間
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数

表 84. 接続イベント・モニターに戻される情報: 表名 : CONN_evmon-name (続き)

列名	データ・タイプ	説明
FAILED_SQL_STMTS	BIGINT	failed_sql_stmts 失敗したステートメント操作
HASH_JOIN_OVERFLOWS	BIGINT	hash_join_overflows ハッシュ結合のオーバーフロー
HASH_JOIN_SMALL_OVERFLOWS	BIGINT	hash_join_small_overflows ハッシュ結合の短精度オーバーフロー
INT_AUTO_REBINDS	BIGINT	int_auto_rebinds 内部自動再バインド
INT_COMMITS	BIGINT	int_commits 内部コミット数
INT_DEADLOCK_ROLLBACKS	BIGINT	int_deadlock_rollbacks デッドロックによる内部ロールバック
INT_ROLLBACKS	BIGINT	int_rollbacks 内部ロールバック数
INT_ROWS_DELETED	BIGINT	int_rows_deleted 削除された内部行数
INT_ROWS_INSERTED	BIGINT	int_rows_inserted 挿入された内部行数
INT_ROWS_UPDATED	BIGINT	int_rows_updated 更新された内部行数
LOCK_ESCALS	BIGINT	lock_escals ロック・エスカレーション数
LOCK_TIMEOUTS	BIGINT	lock_timeouts ロック・タイムアウト数
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCK_WAITS	BIGINT	lock_waits ロック待機数
OLAP_FUNC_OVERFLOWS	BIGINT	olap_func_overflows OLAP 関数のオーバーフロー
PARTIAL_RECORD	SMALLINT	partial_record 部分レコード
PKG_CACHE_INSERTS	BIGINT	pkg_cache_inserts パッケージ・キャッシュ挿入
PKG_CACHE_LOOKUPS	BIGINT	pkg_cache_lookups パッケージ・キャッシュ検索
POOL_DATA_L_READS	BIGINT	pool_data_l_reads バッファー・プール・データの論理読み取り
POOL_DATA_P_READS	BIGINT	pool_data_p_reads バッファー・プール・データの物理読み取り
POOL_DATA_WRITES	BIGINT	pool_data_writes バッファー・プールへのデータの書き込み
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads バッファー・プール索引の論理読み取り
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads バッファー・プール索引の物理読み取り
POOL_INDEX_WRITES	BIGINT	pool_index_writes バッファー・プール索引の書き込み
POOL_READ_TIME	BIGINT	pool_read_time バッファー・プール物理読み取り時間の合計
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads バッファー・プール一時データの論理読み取り

表 84. 接続イベント・モニターに戻される情報: 表名 : CONN_evmon-name (続き)

列名	データ・タイプ	説明
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads バッファー・プール一時データの物理読み取り
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads バッファー・プール一時索引の論理読み取り
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads バッファー・プール一時索引の物理読み取り
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_WRITE_TIME	BIGINT	pool_write_time バッファー・プール物理書き込み時間の合計
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_XDA_WRITES	BIGINT	pool_xda_writes - バッファー・プール XDA データの書き込み
PREFETCH_WAIT_TIME	BIGINT	prefetch_wait_time プリフェッチ待ち時間
REJ_CURS_BLK	BIGINT	rej_curs_blk リジェクトされたブロック・カーソル要求
ROLLBACK_SQL_STMTS	BIGINT	rollback_sql_stmts - 試行されたロールバック・ステートメント
ROWS_DELETED	BIGINT	rows_deleted 削除行数
ROWS_INSERTED	BIGINT	rows_inserted 挿入行数
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_SELECTED	BIGINT	rows_selected 選択行数
ROWS_UPDATED	BIGINT	rows_updated 更新行数
ROWS_WRITTEN	BIGINT	rows_written 書き込み行数
SELECT_SQL_STMTS	BIGINT	select_sql_stmts 実行された選択 SQL ステートメント
SEQUENCE_NO	CHARACTER(5)	sequence_no シーケンス番号
SORT_OVERFLOWS	BIGINT	sort_overflows ソート・オーバーフロー
STATIC_SQL_STMTS	BIGINT	static_sql_stmts 試行された静的 SQL ステートメント
SYSTEM_CPU_TIME	BIGINT	system_cpu_time システム CPU 時間
TOTAL_HASH_JOINS	BIGINT	total_hash_joins ハッシュ結合の合計
TOTAL_HASH_LOOPS	BIGINT	total_hash_loops ハッシュ・ループの合計
TOTAL_OLAP_FUNCS	BIGINT	total_olap_funcs OLAP 関数の合計数

表 84. 接続イベント・モニターに戻される情報: 表名 : CONN_evmon-name (続き)

列名	データ・タイプ	説明
TOTAL_SORT_TIME	BIGINT	total_sort_time ソート時間合計
TOTAL_SORTS	BIGINT	total_sorts ソート合計
UID_SQL_STMTS	BIGINT	uid_sql_stmts 実行された UPDATE/INSERT/DELETE SQL ステートメント
UNREAD_PREFETCH_PAGES	BIGINT	unread_prefetch_pages 読み取り不能プリフェッチ・ページ
USER_CPU_TIME	BIGINT	user_cpu_time ユーザー CPU 時間
X_LOCK_ESCALS	BIGINT	x_lock_escals 排他ロック・エスカレーション数
XQUERY_STMTS	BIGINT	xquery_stmts - 試行された XQuery ステートメント

表 85. 接続イベント・モニターに戻される情報: 表名 : CONMEMUSE_evmon-name

列名	データ・タイプ	説明
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数
POOL_CUR_SIZE	BIGINT	pool_cur_size メモリー・プールの現行サイズ
POOL_ID	BIGINT	pool_id メモリー・プール ID
POOL_LIST_ID	BIGINT	
POOL_MAX_SIZE	BIGINT	
POOL_WATERMARK	BIGINT	pool_watermark メモリー・プール水準点

表 86. 接続イベント・モニターに戻される情報 : デフォルトの表名: CONTROL_evmon-name

列名	データ・タイプ	説明
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数
POOL_CUR_SIZE	BIGINT	pool_cur_size メモリー・プールの現行サイズ
POOL_ID	BIGINT	pool_id メモリー・プール ID
POOL_LIST_ID	BIGINT	
POOL_MAX_SIZE	BIGINT	
POOL_WATERMARK	BIGINT	pool_watermark メモリー・プール水準点

トランザクション・イベント・モニター

トランザクション・イベント・モニターによって生成されるデータ

トランザクション・イベント・モニターは、データベース・トランザクションに関する情報を記録します。

注: このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースでは除去される予定です。作業単位をモニターするには、代わりに作業単位イベント・モニターを使用してください。

選択した出力形式にかかわらず、すべてのトランザクション・イベント・データは、次の 2 つの論理グループから取得されます。

- 110 ページの『event_xact 論理データ・グループ』
- 68 ページの『event_connheader 論理データ・グループ』

また、トランザクション・イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

トランザクション・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、トランザクション・イベント・モニターによって書き込まれる情報。

以降のセクションでは、 WRITE TO TABLE オプションを CREATE EVENT MONITOR ステートメントに使用した場合のトランザクション・イベント・モニターの出力について説明します。イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

表 87. TRANSACTION 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
XACT_*evmon-name*	110 ページの『event_xact 論理データ・グループ』
CONNHEADER_*evmon-name*	68 ページの『event_connheader 論理データ・グループ』
CONTROL_*evmon-name*	CONTROL 論理グループは、event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

生成される表

表 88. トランザクション・イベント・モニターに戻される情報 : デフォルトの表名: XACT_evmon-name

列名	データ・タイプ	説明
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID)
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
EVMON_FLUSHES	BIGINT	evmon_flushes イベント・モニター・フラッシュ回数

表 88. トランザクション・イベント・モニターに戻される情報 : デフォルトの表名: *XACT_evmon-name (続き)*

列名	データ・タイプ	説明
LOCK_ESCALS	BIGINT	lock_escals ロック・エスカレーション数
LOCK_WAIT_TIME	BIGINT	lock_wait_time ロック待機中の時間
LOCKS_HELD_TOP	BIGINT	locks_held_top ロック保持最大数
PARTIAL_RECORD	SMALLINT	partial_record 部分レコード
PREV_UOW_STOP_TIME	TIMESTAMP	prev_uow_stop_time 直前の作業単位完了タイム・スタンプ
ROWS_READ	BIGINT	rows_read 読み取り行数
ROWS_WRITTEN	BIGINT	rows_written 書き込み行数
SEQUENCE_NO	CHARACTER	sequence_no シーケンス番号
STOP_TIME	TIMESTAMP	stop_time イベント停止時刻
SYSTEM_CPU_TIME	BIGINT	system_cpu_time システム CPU 時間
TPMON_ACC_STR	VARCHAR(200)	tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング
TPMON_CLIENT_APP	VARCHAR(255)	tpmon_client_app TP モニター・クライアント・アプリケーション名
TPMON_CLIENT_USERID	VARCHAR(255)	tpmon_client_userid TP モニター・クライアント・ユーザー ID
TPMON_CLIENT_WKSTN	VARCHAR(255)	tpmon_client_wkstn TP モニター・クライアント・ワークステーション名
UOW_LOG_SPACE_USED	BIGINT	uow_log_space_used 使用されている作業単位ログ・スペース
UOW_START_TIME	TIMESTAMP	uow_start_time 作業単位開始タイム・スタンプ
UOW_STATUS	BIGINT	uow_status 作業単位の状況
USER_CPU_TIME	BIGINT	user_cpu_time ユーザー CPU 時間
X_LOCK_ESCALS	BIGINT	x_lock_escals 排他ロック・エスカレーション数

表 89. トランザクション・イベント・モニターに戻される情報 : デフォルトの表名: *CONNHEADER_evmon-name*

列名	データ・タイプ	説明
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID)
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID
APPL_NAME	VARCHAR(255)	appl_name アプリケーション名
AUTH_ID	VARCHAR(128)	auth_id 許可 ID
CLIENT_DB_ALIAS	CHARACTER(8)	client_db_alias アプリケーションで使用するデータベース別名
CLIENT_NNAME	VARCHAR(20)	898 ページの『client_nname - クライアント名モニター・エレメント』
CLIENT_PID	BIGINT	client_pid クライアント・プロセス ID

表 89. トランザクション・イベント・モニターに戻される情報 : デフォルトの表名: *CONNHEADER_evmon-name (続き)*

列名	データ・タイプ	説明
CLIENT_PLATFORM	INTEGER	client_platform クライアント・オペレーティング・プラットフォーム
CLIENT_PRDID	VARCHAR(20)	client_prdid クライアント製品およびバージョン ID
CLIENT_PROTOCOL	INTEGER	client_protocol クライアント通信プロトコル
CODEPAGE_ID	INTEGER	codepage_id アプリケーションで使用するコード・ページ ID
CONN_TIME	TIMESTAMP	conn_time データベース接続時刻
CORR_TOKEN	VARCHAR(64)	corr_token - DRDA 相関トークン
EXECUTION_ID	VARCHAR(128)	execution_id ユーザー・ログイン ID
SEQUENCE_NO	CHARACTER(5)	sequence_no シーケンス番号
TERRITORY_CODE	INTEGER	territory_code データベース・テリトリー・コード

表 90. トランザクション・イベント・モニターに戻される情報 : デフォルトの表名: *CONTROL_evmon-name*

列名	データ・タイプ	説明
EVENT_MONITOR_NAME	VARCHAR(128)	event_monitor_name イベント・モニター名
MESSAGE	VARCHAR(128)	message コントロール表メッセージ
MESSAGE_TIME	TIMESTAMP	message_time タイム・スタンプ・コントロール表メッセージ

デッドロック・イベント・モニター

デッドロック・イベント・モニターによって生成されるデータ

デッドロック・イベント・モニターは、デッドロック状態に関する情報を記録します。

注: このイベント・モニターは推奨されなくなりました。これはもはや推奨されていません。将来のリリースでは除去される予定です。デッドロックをモニターするには、代わりにロック・イベント・モニターを使用してください。

選択した出力形式にかかわらず、すべてのデッドロック・イベント・データは、次の 3 つの論理グループから取得されます。

- 68 ページの『event_connheader 論理データ・グループ』
- 75 ページの『event_deadlock 論理データ・グループ』
- 77 ページの『event_dlconn 論理データ・グループ』

また、トランザクション・イベント・データを表に書き込むように選択した場合は、イベント・モニター自体に関するメタデータを生成するために、付加的なグループ (CONTROL) のデータも使用されます。

デッドロック・イベント・モニター用に表に書き込まれる情報:

WRITE TO TABLE オプションを指定した場合に、デッドロック・イベント・モニターによって書き込まれる情報。

以降のセクションでは、 WRITE TO TABLE オプションを CREATE EVENT MONITOR ステートメントに使用した場合のデッドロック・イベント・モニターの出力について説明します。イベント・モニターがファイルまたは名前付きパイプに書き込む場合に戻される出力については、 158 ページの『イベント・モニター自己記述型データ・ストリーム』を参照してください。

表 91. DEADLOCK 表書き込みイベント・モニターによって生成される表 : 表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前 (次の表に示す表名では *evmon-name* と表しています) が連結されて生成されます。

デフォルトの表名	レポートされる論理データ・グループ
CONNHEADER_*evmon-name*	68 ページの『event_connheader 論理データ・グループ』
DEADLOCK_*evmon-name*	75 ページの『event_deadlock 論理データ・グループ』
DLCONN_*evmon-name*	77 ページの『event_dlconn 論理データ・グループ』
CONTROL_*evmon-name*	CONTROL 論理グループは、 event_dbheader、event_start、および event_overflow 論理データ・グループの 1 つ以上から選択されたエレメントで構成されています。

生成される表

表 92. デッドロック・イベント・モニターに戻される情報 : デフォルトの表名: CONNHEADER_evmon-name

列名	データ・タイプ	説明
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID : モニター・エレメント
APPL_NAME	VARCHAR(255)	appl_name アプリケーション名 : モニター・エレメント
AUTH_ID	VARCHAR(128)	auth_id 許可 ID : モニター・エレメント
CLIENT_DB_ALIAS	CHARACTER(8)	client_db_alias アプリケーションで使用するデータベース別名 : モニター・エレメント
CLIENT_NNAME	VARCHAR(20)	898 ページの『client_nname - クライアント名モニター・エレメント』
CLIENT_PID	BIGINT	client_pid - クライアント・プロセス ID : モニター・エレメント
CLIENT_PLATFORM	INTEGER	client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント
CLIENT_PRDID	VARCHAR(20)	client_prdid クライアント製品およびバージョン ID : モニター・エレメント

表 92. デッドロック・イベント・モニターに戻される情報 : デフォルトの表名: *CONNHEADER_evmon-name (続き)*

列名	データ・タイプ	説明
CLIENT_PROTOCOL	INTEGER	client_protocol - クライアント通信プロトコル : モニター・エレメント
CODEPAGE_ID	INTEGER	codepage_id アプリケーションで使用するコード・ページ ID : モニター・エレメント
CONN_TIME	TIMESTAMP	conn_time データベース接続時刻 : モニター・エレメント
CORR_TOKEN	VARCHAR(64)	corr_token - DRDA 相関トークン : モニター・エレメント
EXECUTION_ID	VARCHAR(128)	execution_id ユーザー・ログイン ID : モニター・エレメント
SEQUENCE_NO	CHARACTER(5)	sequence_no シーケンス番号 : モニター・エレメント
TERRITORY_CODE	INTEGER	territory_code データベース・テリトリー・コード : モニター・エレメント

表 93. デッドロック・イベント・モニターに戻される情報 : デフォルトの表名: *DEADLOCK_evmon-name*

列名	データ・タイプ	説明
DEADLOCK_ID	BIGINT	deadlock_id デッドロック・イベント ID : モニター・エレメント
DL_CONNS	BIGINT	dl_conns - デッドロックに関係している接続 : モニター・エレメント
EVMON_ACTIVATES	BIGINT	evmon_activates イベント・モニター活動化回数 : モニター・エレメント
ROLLED_BACK_AGENT_ID	BIGINT	rolled_back_agent_id ロールバックされたエージェント : モニター・エレメント
ROLLED_BACK_APPL_ID	VARCHAR(64)	rolled_back_appl_id - ロールバック・アプリケーション : モニター・エレメント
ROLLED_BACK_PARTICIPANT_NO	SMALLINT	rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント
ROLLED_BACK_SEQUENCE_NO	CHARACTER(5)	rolled_back_sequence_no ロールバックされたシーケンス番号 : モニター・エレメント
START_TIME	TIMESTAMP	start_time イベント開始時刻 : モニター・エレメント

表 94. デッドロック・イベント・モニターに戻される情報 : デフォルトの表名: *DLCONN_evmon-name*

列名	データ・タイプ	説明
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID : モニター・エレメント
APPL_ID_HOLDING_LK	VARCHAR(64)	appl_id_holding_lk - ロックを保持しているアプリケーション ID : モニター・エレメント

表 94. デッドロック・イベント・モニターに戻される情報 : デフォルトの表名: *DLCONN_evmon-name (続き)*

列名	データ・タイプ	説明
DATA_PARTITION_ID	INTEGER	data_partition_id - データ・パーティション ID : モニター・エレメント
DEADLOCK_ID	BIGINT	deadlock_id デッドロック・イベント ID : モニター・エレメント
EVMON_ACTIVATES	BIGINT	evmon_activates イベント・モニター活動化回数 : モニター・エレメント
LOCK_ATTRIBUTES	BIGINT	lock_attributes ロック属性 : モニター・エレメント
LOCK_COUNT	BIGINT	lock_count ロック・カウント : モニター・エレメント
LOCK_CURRENT_MODE	BIGINT	lock_current_mode - 変換前の元のロック・モード : モニター・エレメント
LOCK_ESCALATION	SMALLINT	lock_escalation ロック・エスカレーション : モニター・エレメント
LOCK_HOLD_COUNT	BIGINT	lock_hold_count ロック保留カウント : モニター・エレメント
LOCK_MODE	BIGINT	lock_mode ロック・モード : モニター・エレメント
LOCK_MODE_REQUESTED	BIGINT	lock_mode_requested 要求されているロック・モード : モニター・エレメント
LOCK_NAME	CHARACTER(13)	lock_name ロック名 : モニター・エレメント
LOCK_NODE	BIGINT	lock_node ロック・ノード : モニター・エレメント
LOCK_OBJECT_NAME	BIGINT	lock_object_name - ロック対象名 : モニター・エレメント
LOCK_OBJECT_TYPE	BIGINT	lock_object_type - 待機中のロック対象タイプ : モニター・エレメント
LOCK_RELEASE_FLAGS	BIGINT	lock_release_flags ロック保留解除フラグ : モニター・エレメント
LOCK_WAIT_START_TIME	TIMESTAMP	lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント
PARTICIPANT_NO	SMALLINT	participant_no デッドロック内の参加者 : モニター・エレメント
PARTICIPANT_NO_HOLDING_LK	SMALLINT	participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者 : モニター・エレメント
SEQUENCE_NO	CHARACTER(5)	sequence_no シーケンス番号 : モニター・エレメント
SEQUENCE_NO_HOLDING_LK	CHARACTER(5)	sequence_no_holding_lk - ロックを保持しているシーケンス番号 : モニター・エレメント
START_TIME	TIMESTAMP	start_time イベント開始時刻 : モニター・エレメント
TABLE_NAME	VARCHAR(128)	table_name - 表名 : モニター・エレメント
TABLE_SCHEMA	VARCHAR(128)	table_schema - 表スキーマ名 : モニター・エレメント

表 94. デッドロック・イベント・モニターに戻される情報 : デフォルトの表名: *DLCONN_evmon-name (続き)*

列名	データ・タイプ	説明
TABLESPACE_NAME	VARCHAR(18)	tablespace_name - 表スペース名 : モニター・エレメント

表 95. デッドロック・イベント・モニターに戻される情報 : デフォルトの表名: *CONTROL_evmon-name*

列名	データ・タイプ	説明
AGENT_ID	BIGINT	agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント
APPL_ID	VARCHAR(64)	appl_id - アプリケーション ID : モニター・エレメント
APPL_ID_HOLDING_LK	VARCHAR(64)	appl_id_holding_lk - ロックを保持しているアプリケーション ID : モニター・エレメント
DATA_PARTITION_ID	INTEGER	data_partition_id - データ・パーティション ID : モニター・エレメント
DEADLOCK_ID	BIGINT	deadlock_id デッドロック・イベント ID : モニター・エレメント
EVMON_ACTIVATES	BIGINT	evmon_activates イベント・モニター活動化回数 : モニター・エレメント
LOCK_ATTRIBUTES	BIGINT	lock_attributes ロック属性 : モニター・エレメント
LOCK_COUNT	BIGINT	lock_count ロック・カウント : モニター・エレメント
LOCK_CURRENT_MODE	BIGINT	lock_current_mode - 変換前の元のロック・モード : モニター・エレメント
LOCK_ESCALATION	SMALLINT	lock_escalation ロック・エスカレーション : モニター・エレメント
LOCK_HOLD_COUNT	BIGINT	lock_hold_count ロック保留カウント : モニター・エレメント
LOCK_MODE	BIGINT	lock_mode ロック・モード : モニター・エレメント
LOCK_MODE_REQUESTED	BIGINT	lock_mode_requested 要求されているロック・モード : モニター・エレメント
LOCK_NAME	CHARACTER(13)	lock_name ロック名 : モニター・エレメント
LOCK_NODE	BIGINT	lock_node ロック・ノード : モニター・エレメント
LOCK_OBJECT_NAME	BIGINT	lock_object_name - ロック対象名 : モニター・エレメント
LOCK_OBJECT_TYPE	BIGINT	lock_object_type - 待機中のロック対象タイプ : モニター・エレメント
LOCK_RELEASE_FLAGS	BIGINT	lock_release_flags ロック保留解除フラグ : モニター・エレメント
LOCK_WAIT_START_TIME	TIMESTAMP	lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント
PARTICIPANT_NO	SMALLINT	participant_no デッドロック内の参加者 : モニター・エレメント

表 95. デッドロック・イベント・モニターに戻される情報 : デフォルトの表名: CONTROL_evmon-name (続き)

列名	データ・タイプ	説明
PARTICIPANT_NO_HOLDING_LK	SMALLINT	participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者 : モニター・エレメント
SEQUENCE_NO	CHARACTER(5)	sequence_no シーケンス番号 : モニター・エレメント
SEQUENCE_NO_HOLDING_LK	CHARACTER(5)	sequence_no_holding_lk - ロックを保持しているシーケンス番号 : モニター・エレメント
START_TIME	TIMESTAMP	start_time イベント開始時刻 : モニター・エレメント
TABLE_NAME	VARCHAR(128)	table_name - 表名 : モニター・エレメント
TABLE_SCHEMA	VARCHAR(128)	table_schema - 表スキーマ名 : モニター・エレメント
TABLESPACE_NAME	VARCHAR(18)	tablespace_name - 表スペース名 : モニター・エレメント

変更履歴イベント・モニター

変更履歴イベント・モニターは、標準的なデータベース・ワークロードの実行に影響を与える可能性がある、データベース・サーバー上のイベントに関する情報をキャプチャーします。このイベント・モニターでキャプチャーしたデータを使用すると、データベースおよびデータベース管理システムの動作、パフォーマンス、または安定性の変化について調べることができます。

標準的なワークロードにパフォーマンスの低下が発生したり、予期しない動作が見られたりした場合は、そのワークロードの動作の変化と、変更履歴イベント・モニターでキャプチャーされたイベントとの相関関係を調べることができます。次の変更は、データベース・システムに悪影響を及ぼす場合があります。

- 予期しない索引の作成またはドロップ
- 定期保守の実行の失敗
- データベース構成パラメーターまたは DB2 レジストリー変数の変更

変更は、ユーザーによって明示的に実行された可能性があります。例えば、管理者が、索引をドロップする DDL ステートメントを実行した可能性があります。また、ユーザーとの対話なしに暗黙的または自動的に変更が発生した可能性もあります。例えば、セルフチューニング・メモリー・マネージャー (STMM) によって構成パラメーターが変更されたり、表の自動再編成によって表が再編成されたりした可能性があります。

データベース・サーバーに対する変更を手動で追跡することは、困難な作業となる場合があります。これまで、さまざまなインターフェースを介して、さまざまなタイプの変更に関する情報がキャプチャーされてきました。例えば、構成の更新は診断ログ・ファイル (db2diag ログ・ファイルなど) に書き込まれますが、ユーティリティーの進行はデータベース履歴ファイルにキャプチャーされます。変更履歴イベント・モニターは、データベース・システムの動作およびパフォーマンスの特性を

変化させるイベントをキャプチャーする、単一のインターフェースを提供します。イベント・モニター表を使用して、関心のある変更イベントについて調べることができます。

変更履歴イベント・モニターは、多数のアクションおよび操作に関して変更関連イベントをキャプチャーできます。以下に例を示します。

- データベースおよびデータベース・マネージャーの構成パラメーターの変更
- レジストリー変数の変更
- DDL ステートメントの実行
- 変更履歴イベント・モニターの開始
- 次の DB2 ユーティリティーおよびコマンドの実行
  - LOAD
  - ADMIN_MOVE_TABLE プロシージャーの呼び出し
  - BACKUP DATABASE (ONLINE オプションのみ)
  - RESTORE DATABASE (ONLINE オプションのみ)
  - ROLLFORWARD DATABASE (ONLINE オプションのみ)
  - REDISTRIBUTE DATABASE PARTITION GROUP
  - REORG
  - RUNSTATS

一般的に、変更履歴イベント・モニターが非アクティブである間、またはデータベースがオフラインの間に発生したイベントに関連する情報は、キャプチャーされません。ただし、イベント・モニターの活動化の際に有効なレジストリー変数値をキャプチャーするように、変更履歴イベント・モニターを構成することができます。同様に、データベースおよびデータベース・マネージャーの構成パラメーター値も、変更履歴イベント・モニターの活動化の際にキャプチャーすることができます。構成パラメーター値をキャプチャーするときに、イベント・モニターは、イベント・モニターが非アクティブであった間に構成パラメーターが変更されたかどうかを検出できます。そのため、イベント・モニターは、変更が行われていた場合にのみ、構成パラメーター値をキャプチャーします。

変更履歴イベント・モニターによって生成されるデータ

変更履歴イベント・モニターは、データベースとデータベース管理システムのパフォーマンス、動作、および安定性に影響を与える可能性があるアクティビティーに関する情報をキャプチャーします。変更履歴イベント・モニターの出力は、論理データ・グループに書き込まれます。論理データ・グループには、それぞれ、関連付けられたイベント・モニター表があります。

1 つの変更関連アクションで、変更履歴イベント・モニターに 1 つ以上のイベントが生成される可能性があります。例えば、データベース構成の更新では単一のイベントが生成されますが、REORG ユーティリティーの実行では、 REORG 操作の開始と終了を示す 2 つのイベントが生成されます。イベントと論理データ・グループは、1 対多でマッピングされます。 1 つのイベントで、複数の論理データ・グループに情報が書き込まれる場合があります。また、 1 つのイベントで、特定の論理デ

ータ・グループに関連付けられている表に、複数のエントリー (行) が書き込まれる場合があります。各変更関連イベントは、次の 3 つのキー・フィールドによって一意的に識別されます。

イベント・タイム・スタンプ
イベントが発生した時刻。

イベント ID
イベント・タイム・スタンプが共通している場合に、固有性を確保するための数値トークン。

メンバー
イベントが発生したデータベース・マネージャー・プロセス。

すべての論理グループには、これらの 3 つのフィールドが含まれており、同じイベントに対応するレコードまたは行はすべて、これらのフィールドに同じ値が入っています。これらの共通する値によって、異なる論理データ・グループ間で情報を結合することが容易になります。別のメンバーでのユーティリティー操作および構成パラメーター更新は、別のイベントとしてキャプチャーされるため、これらのキー・フィールドは異なる値になります。

変更履歴イベント・モニターは、イベント・モニターの論理データ・グループの TABLE ターゲットのみをサポートします。変更履歴イベント・モニターは、UNFORMATTED EVENT TABLE、FILE、および PIPE ターゲットをサポートしません。

次の表に、変更履歴イベント・モニターで使用される論理データ・グループおよび関連表をリストします。各論理データ・グループのデフォルトの表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントによって作成したときにイベント・モニターに指定した名前が連結されて生成されます。ここに示している表名は、 CREATE EVENT MONITOR ステートメントの一部として表名が指定されなかった場合のデフォルトの表名です。

表 96. 変更履歴イベント・モニターの論理データ・グループ

論理データ・グループ	デフォルトの表名	内容
CHANGESUMMARY	CHANGESUMMARY_evmon-name (478 ページの『CHANGESUMMARY 論理データ・グループ』を参照)	変更履歴イベント・モニターによってキャプチャーされたすべてのイベントの要約
DBDBMCFG	DBDBMCFG_evmon-name (482 ページの『DBDBMCFG 論理データ・グループ』を参照)	構成パラメーターの変更
REGVAR	REGVAR_evmon-name (484 ページの『REGVAR 論理データ・グループ』を参照)	レジストリー変数の変更
DDLSTMTEXEC	DDLSTMTEXEC_evmon-name (485 ページの『DDLSTMTEXEC 論理データ・グループ』を参照)	DDL 実行
TXNCOMPLETION	TXNCOMPLETION_evmon-name (488 ページの『TXNCOMPLETION 論理データ・グループ』を参照)	コミット、ロールバック、またはセーブポイントまでのロールバックの発生

表 96. 変更履歴イベント・モニターの論理データ・グループ (続き)

論理データ・グループ	デフォルトの表名	内容
EVMONSTART	EVMONSTART_evmon-name (489 ページの『EVMONSTART 論理データ・グループ』を参照)	イベント・モニターの開始情報
UTILSTART	UTILSTART_evmon-name (490 ページの『UTILSTART 論理データ・グループ』を参照)	ユーティリティーの開始情報
UTILLOCATION	UTILLOCATION_evmon-name (495 ページの『UTILLOCATION 論理データ・グループ』を参照)	ユーティリティーのパスまたはファイル情報
UTILSTOP	UTILSTOP_evmon-name (497 ページの『UTILSTOP 論理データ・グループ』を参照)	ユーティリティーの停止情報
UTILPHASE	UTILPHASE_evmon-name (499 ページの『UTILPHASE 論理データ・グループ』を参照)	ユーティリティーのフェーズ情報

変更履歴イベント・モニターは、さまざまなイベントをキャプチャーできます。すべてのイベントが、すべてのユーザーにとって重要なイベントというわけではありません。イベント・モニターの作成時に CREATE EVENT MONITOR ステートメントに WHERE EVENT IN 節を使用すると、変更履歴イベント・モニターでキャプチャーするイベント・タイプを制御することができます。

次の表に、変更履歴イベント・モニターでキャプチャー可能なイベントが生成される、データベース・サーバー上のアクションを示します。また、それらのイベントをキャプチャーするために WHERE EVENT IN 節に指定する制御オプション、イベントが生成されたときにデータが設定される論理データ・グループも示します。

表 97. アクションによって生成されるイベント

アクション	イベント・タイプ	WHERE EVENT IN 節	論理データ・グループ	詳細
データベース構成パラメーターの変更	DBCFG	ALL CFGALL DBCFG	CHANGESUMMARY DBDBMCFG	CHANGESUMMARY に 1 つのレコードが書き込まれます 変更されたパラメーターごとに DBDBMCFG に 1 つのレコードが書き込まれます。
イベント・モニターが非アクティブであった間に、データベース構成パラメーターが変更されていた場合の、イベント・モニター開始時の全データベース構成パラメーター値のキャプチャー	DBCFGVALUES	ALL CFGALL DBCFGVALUES	CHANGESUMMARY DBDBMCFG	CHANGESUMMARY に 1 つのレコードが書き込まれます パラメーターごとに DBDBMCFG に 1 つのレコードが書き込まれます。

表 97. アクションによって生成されるイベント (続き)

アクション	イベント・タイプ	WHERE EVENT IN 節	論理データ・グループ	詳細
データベース・マネージャー構成パラメーターの変更	DBMCFG	ALL CFGALL DBMCFG	CHANGESUMMARY DBDBMCFG	CHANGESUMMARY に 1 つのレコードが書き込まれます 変更されたパラメーターごとに DBDBMCFG に 1 つのレコードが書き込まれます。
イベント・モニターが非アクティブであった間に、データベース・マネージャー構成パラメーターが変更されていた場合の、イベント・モニター開始時の全データベース・マネージャー構成パラメーター値のキャプチャー	DBMCFGVALUES	ALL CFGALL DBMCFGVALUES	CHANGESUMMARY DBDBMCFG	CHANGESUMMARY に 1 つのレコードが書き込まれます パラメーターごとに DBDBMCFG に 1 つのレコードが書き込まれます。
レジストリー変数の変更。即時更新 (**db2set** コマンドに -immediate フラグを使用したレジストリー変数変更) の場合のみ、イベントが生成されます。	REGVAR	ALL CFGALL REGVAR	CHANGESUMMARY REGVAR	CHANGESUMMARY に 1 つのレコードが書き込まれます 変更された変数ごとに REGVAR に 1 つのレコードが書き込まれます。明示的に設定されたレジストリー変数のみ、キャプチャーされます。集約レジストリー変数によって暗黙的に設定された変数の場合は、レコードは書き込まれません。
イベント・モニター開始時のレジストリー変数値のキャプチャー	REGVARVALUES	ALL CFGALL REGVARVALUES	CHANGESUMMARY REGVAR	CHANGESUMMARY に 1 つのレコードが書き込まれます 明示的に設定された変数ごとに REGVAR に 1 つのレコードが書き込まれます。

表 97. アクションによって生成されるイベント (続き)

アクション	イベント・タイプ	WHERE EVENT IN 節	論理データ・グループ	詳細
DDL ステートメントの正常実行	DDLSTMTEXEC	ALL DDLALL DDLDATA DDLFEDERATED DDLMONITOR DDLSECURITY DDLSQL DDLSTORAGE DDLWLM DDLXML	CHANGESUMMARY DDLSTMTEXEC	CHANGESUMMARY に 1 つのレコードが書き込まれます DDLSTMTEXEC に 1 つのレコードが書き込まれます
正常実行した DDL ステートメントを含むトランザクションのコミットまたはロールバック、あるいは正常実行した DDL ステートメントを含むセーブポイントまでのロールバック	TXNCOMPLETION	ALL DDLALL DDLDATA DDLFEDERATED DDLMONITOR DDLSECURITY DDLSQL DDLSTORAGE DDLWLM DDLXML	CHANGESUMMARY TXNCOMPLETION	CHANGESUMMARY に 1 つのレコードが書き込まれます TXNCOMPLETION に 1 つのレコードが書き込まれます
イベント・モニターの開始または停止	EVMONSTART	該当なし	CHANGESUMMARY EVMONSTART	CHANGESUMMARY に 1 つのレコードが書き込まれます EVMONSTART に 1 つのレコードが書き込まれます
ユーティリティー実行の開始、または一時停止後の実行の再開。このイベントは、コーディネーター・メンバー上でのみ生成されます。	UTILSTART	ALL BACKUP LOAD MOVETABLE REDISTRIBUTE REORG RESTORE ROLLFORWARD RUNSTATS UTILALL	CHANGESUMMARY UTILSTART UTILLOCATION	CHANGESUMMARY に 1 つのレコードが書き込まれます UTILSTART に 1 つのレコードが書き込まれます 0 個以上のレコード (ユーティリティーの開始に関連するファイルごとに 1 レコード) が UTILLOCATION に書き込まれます。

表 97. アクションによって生成されるイベント (続き)

アクション	イベント・タイプ	WHERE EVENT IN 節	論理データ・グループ	詳細
ユーティリティー実行の完了または実行の一時停止。このイベントは、コーディネーター・メンバー上でのみ生成されます。	UTILSTOP	ALL BACKUP LOAD MOVETABLE REDISTRIBUTE REORG RESTORE ROLLFORWARD RUNSTATS UTILALL	CHANGESUMMARY UTILSTOP	CHANGESUMMARY に 1 つのレコードが書き込まれます UTILSTOP に 1 つのレコードが書き込まれます
メンバー上でのユーティリティー処理の開始。このイベントは、複数メンバー環境でのみ生成されます。	UTILSTARTPROC	ALL BACKUP LOAD MOVETABLE REDISTRIBUTE REORG RESTORE ROLLFORWARD RUNSTATS UTILALL	CHANGESUMMARY UTILSTART	CHANGESUMMARY に 1 つのレコードが書き込まれます UTILSTART に 1 つのレコードが書き込まれます
メンバー上でのユーティリティー処理の停止。このイベントは、複数メンバー環境でのみ生成されます。	UTILSTOPPROC	ALL BACKUP LOAD MOVETABLE REDISTRIBUTE REORG RESTORE ROLLFORWARD RUNSTATS UTILALL	CHANGESUMMARY UTILSTOP	CHANGESUMMARY に 1 つのレコードが書き込まれます UTILSTOP に 1 つのレコードが書き込まれます
メンバー上でのユーティリティーの特定の処理フェーズの実行の開始	UTILPHASESTART	ALL BACKUP UTILALL	CHANGESUMMARY UTILPHASE	CHANGESUMMARY に 1 つのレコードが書き込まれます UTILPHASE に 1 つのレコードが書き込まれます
メンバー上でのユーティリティーの特定の処理フェーズの実行の停止	UTILPHASESTOP	ALL BACKUP UTILALL	CHANGESUMMARY UTILPHASE	CHANGESUMMARY に 1 つのレコードが書き込まれます UTILPHASE に 1 つのレコードが書き込まれます

CHANGESUMMARY 論理データ・グループ:

CHANGESUMMARY 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行は、発生した固有の変更履歴イベントを表します。

この表を使用すると、変更履歴イベント・モニターによって変更がキャプチャーされたかどうか、およびそれらの変更の詳細はどの論理データ・グループに入っているのかを、簡単に調べることができます。

次の表に、変更履歴イベント・モニターによって収集される変更イベント情報の要約を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 98. 変更履歴イベント・モニターによって戻される CHANGESUMMARY 論理データ・グループの情報。デフォルトの表名は CHANGESUMMARY_evmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは以下のいずれかです。 • DBCFG • DBCFGVALUES • DBMCFG • DBMCFGVALUES • REGVAR • REGVARVALUES • DDLSTMTEXEC • TXNCOMPLETION • EVMONSTART • UTILSTART • UTILSTOP • UTILSTARTPROC • UTILSTOPPROC • UTILPHASESTART • UTILPHASESTOP
COORD_MEMBER	SMALLINT	該当する作業単位またはワークロードのコーディネーター・メンバー。

表 98. 変更履歴イベント・モニターによって戻される CHANGESUMMARY 論理データ・グループの情報。デフォルトの表名は CHANGESUMMARY_evmon-name です。 (続き)

列名	データ・タイプ	説明
UTILITY_INVOCATION_ID	VARCHAR(32) FOR BIT DATA	次の EVENT_TYPE の場合にキャプチャーされるユーティリティー呼び出しに対応するユニーク ID。 • UTILSTART • UTILSTOP • UTILSTARTPROC • UTILSTOPPROC • UTILPHASESTART • UTILPHASESTOP 他の EVENT_TYPE の場合は、空のストリングとなります。
UTILITY_TYPE	VARCHAR(16)	UTILITY_INVOCATION_ID が空ではない場合、タイプは次のいずれかです。 • BACKUP • LOAD • MOVETABLE • REDISTRIBUTE • REORG • RESTORE • ROLLFORWARD • RUNSTATS そうでない場合、空ストリングが入っています。
APPL_ID	VARCHAR(64)	アプリケーションのデータベース接続時に生成された ID。
APPL_NAME	VARCHAR(255)	クライアントで実行中のアプリケーションの名前。データベースが識別できる名前です。
APPLICATION_HANDLE	BIGINT	システム全体でのアプリケーションのユニーク ID。
SYSTEM_AUTHID	VARCHAR(128)	接続のシステム許可 ID。これは system_auth_id モニター・エレメントのシノニムです。
SESSION_AUTHID	VARCHAR(128)	アプリケーションによって使用されている現在のセッション許可 ID。これは session_auth_id モニター・エレメントのシノニムです。

表 98. 変更履歴イベント・モニターによって戻される CHANGESUMMARY 論理データ・グループの情報。デフォルトの表名は CHANGESUMMARY_evmon-name です。 (続き)

列名	データ・タイプ	説明
CLIENT_PLATFORM	VARCHAR(12)	クライアント・アプリケーションが稼働中のオペレーティング・システム。
CLIENT_PROTOCOL	VARCHAR(10)	クライアント・アプリケーションがサーバーとの通信に使用している通信プロトコル。
CLIENT_PORT_NUMBER	INTEGER	データベース・サーバーと通信するためにアプリケーションが使用しているクライアント・マシン側のポート番号。
CLIENT_PID	BIGINT	データベースへの接続を行ったクライアント・アプリケーションのプロセス ID。
CLIENT_HOSTNAME	VARCHAR(255)	クライアント・アプリケーションの接続元のマシンのホスト名。
CLIENT_WRKSTNNAME	VARCHAR(255)	クライアント・システムまたはワークステーションを識別する名前。
CLIENT_ACCTNG	VARCHAR(200)	ロギングおよび診断の目的でターゲット・データベースに渡されたデータ。
CLIENT_USERID	VARCHAR(255)	サーバーに示されたクライアント・ユーザー ID。
CLIENT_APPLNAME	VARCHAR(255)	トランザクションを実行するサーバー・トランザクション・プログラムを識別する名前。

表 98. 変更履歴イベント・モニターによって戻される CHANGESUMMARY 論理データ・グループの情報。デフォルトの表名は CHANGESUMMARY_evmon-name です。 (続き)

列名	データ・タイプ	説明
BACKUP_TIMESTAMP	VARCHAR(14)	UTILITY_TYPE が BACKUP で、EVENT_TYPE が UTILSTART である場合、BACKUP_TIMESTAMP 値はバックアップ・イメージのタイム・スタンプです。 UTILITY_TYPE が RESTORE で、EVENT_TYPE が UTILSTOP である場合、BACKUP_TIMESTAMP 値はバックアップ・イメージのタイム・スタンプです。 それ以外の場合、BACKUP_TIMESTAMP は空ストリングです。 BACKUP_TIMESTAMP は、SYSIBMADM.DB_HISTORY 管理ビューを使用して、データベース履歴ファイルに保管されている情報と照合する (シーケンス情報を参照するなど) ことができます。

DBDBMCFG 論理データ・グループ:

DBDBMCFG 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行は、DBCFG または DBMCFG イベントの一環として更新された構成パラメーター、または、イベント・モニターの開始時に DBCFGVALUES または DBMCFGVALUES イベントの一環としてキャプチャーされた構成パラメーターを表します。

CFG_COLLECTION_TYPE モニター・エレメントは、レコードの内容が、構成パラメーターの更新に関するものか、あるいはイベント・モニターの開始時に記録された初期値に関するものかを示します。

次の表に、変更履歴イベント・モニターによって収集される構成パラメーター変更を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 99. 変更履歴イベント・モニターによって戻される DBDBMCFG 論理データ・グループの情報。デフォルトの表名は DBDBMCFG_evmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。

表 99. 変更履歴イベント・モニターによって戻される DBDBMCFG 論理データ・グループの情報。デフォルトの表名は DBDBMCFG_evmon-name です。 (続き)

列名	データ・タイプ	説明
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは以下のいずれかです。 • DBCFG • DBCFGVALUES • DBMCFG • DBMCFGVALUES
CFG_NAME	VARCHAR(128)	構成パラメーターの名前。
CFG_VALUE	VARCHAR(255)	EVENT_TYPE が DBCFG または DBMCFG の場合、これは、構成パラメーターの新しい値です。 EVENT_TYPE が DBCFGVALUES または DBMCFGVALUES の場合、これは、ディスク上の構成パラメーター値です。ディスク上の構成パラメーター値は、最新の値であり、まだ反映されていない可能性があります。
CFG_VALUE_FLAGS	VARCHAR(32)	このフラグは、新しい構成パラメーターの値がどのように決定されたかを示します。 • AUTOMATIC • COMPUTED • NONE EVENT_TYPE が DBCFGVALUES または DBMCFGVALUES の場合、このフラグは、現行のディスク上の構成パラメーター値を表します。
CFG_OLD_VALUE	VARCHAR(255)	EVENT_TYPE が DBCFG または DBMCFG の場合、これは構成パラメーターの古い値です。 EVENT_TYPE が DBCFGVALUES または DBMCFGVALUES の場合、これは、現行のメモリー内の構成パラメーター値です。これは、現在使用中の構成パラメーター値です。

表 99. 変更履歴イベント・モニターによって戻される DBDBMCFG 論理データ・グループの情報。デフォルトの表名は DBDBMCFG_evmon-name です。 (続き)

列名	データ・タイプ	説明
CFG_OLD_VALUE_FLAGS	VARCHAR(32)	このフラグは、古い構成パラメーターの値がどのように決定されたかを示します。 • AUTOMATIC • COMPUTED • NONE EVENT_TYPE が DBCFGVALUES または DBMCFGVALUES の場合、このフラグは、現行のメモリー内の構成パラメーター値を表します。
CFG_COLLECTION_TYPE	CHAR(1)	構成パラメーター値がいつ収集されたかを示します。 **I** イベント・モニターが活動化されたときにキャプチャーされた初期値。 **U** 更新された値
DEFERRED	CHAR(1)	構成パラメーター値の変更が据え置かれたかどうかを示します。 **Y** 変更は次のデータベース活動化まで据え置かれます。 **N** 変更は即時に有効になります

REGVAR 論理データ・グループ:

REGVAR 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行は、REGVAR イベントの一環として更新されたレジストリー変数、または、イベント・モニターの開始時に RERVARVALUES イベントの一環としてキャプチャーされたレジストリー変数を表します。

REGVAR_COLLECTION_TYPE モニター・エレメントは、レコードの内容が、即時レジストリー変数更新 (*U*) に関するものか、あるいはイベント・モニターの開始時に記録された初期値 (*I*) に関するものかを示します。

次の表に、変更履歴イベント・モニターによって収集されるレジストリー変数の変更を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 100. 変更履歴イベント・モニターによって戻される REGVAR 論理データ・グループの情報。デフォルトの表名は REGVAR_evmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。

表 100. 変更履歴イベント・モニターによって戻される REGVAR 論理データ・グループの情報。デフォルトの表名は REGVAR_evmon-name です。 (続き)

列名	データ・タイプ	説明
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは以下のいずれかです。 • REGVAR • REGVARVALUES
REGVAR_NAME	VARCHAR(256)	レジストリー変数の名前。
REGVAR_VALUE	CLOB(2k)	これはレジストリー変数の値です。設定されていない場合、値は空ストリング です。
REGVAR_OLD_VALUE	CLOB(2k)	これはレジストリー変数の古い値です。値が設定されなかった場合、この値は空ストリングです。
REGVAR_LEVEL	CHAR(1)	レジストリー変数のレベルを示します。 **E** 環境 **G** グローバル **I** インスタンス・レベル **P** データベース・パーティション
REGVAR_COLLECTION_TYPE	CHAR(1)	レジストリー変数の値がいつ収集されたかを示します。 **I** イベント・モニターが活動化されたときにキャプチャーされた初期値。 **U** 更新された値

DDLSTMTEXEC 論理データ・グループ:

DDLSTMTEXEC 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行は、1 つの実行済み DDL ステートメント・イベントを表します。パーティション・データベース環境では、行は、DDL 実行のコーディネーター・パーティション上でキャプチャーされます。

DDLSTMTEXEC_evmon-name 表に行が書き込まれるたびに、DDL ステートメントに続く、同じ作業単位で実行されたそれぞれの関連トランザクションの状態変更 (コミット、ロールバック、またはセーブポイントへのロールバック) に対して、TXNCOMPLETION_evmon-name 表に行が書き込まれます。その後、DDLSTMTEXEC_evmon-name 表の GLOBAL_TRANSACTION_ID 列、LOCAL_TRANSACTION_ID 列、および SAVEPOINT_ID 列を使用して、対応する状態変更操作を TXNCOMPLETION_evmon-name 表内で見つけ、DDL がコミットされたかどうかを判別することができます。

次の表に、変更履歴イベント・モニターが収集する DDL ステートメントの実行情報を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 101. 変更履歴イベント・モニターによって戻される DDLSTMTEXEC 論理データ・グループの情報。デフォルトの表名は DDLSTMTEXEC_evmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは DDLSTMTEXEC です。
GLOBAL_TRANSACTION_ID	VARCHAR(40)	イベントが発生した時刻で使用していたグローバル・トランザクション ID。 これは、トランザクション・ログの一部となる SQLP_GXID 構造体のデータ・フィールドです。
LOCAL_TRANSACTION_ID	VARCHAR(16)	イベントが発生した時刻で使用していたローカル・トランザクション ID。 これは、トランザクション・ログの一部となる SQLU_TID 構造体です。
SAVEPOINT_ID	BIGINT	作業単位内で設定されたセーブポイントの名前。
UOW_ID	INTEGER	アプリケーション・ハンドル内の作業単位の固有 ID。

表 101. 変更履歴イベント・モニターによって戻される DDLSTMTEXEC 論理データ・グループの情報。デフォルトの表名は DDLSTMTEXEC_evmon-name です。 (続き)

列名	データ・タイプ	説明
DDL_CLASSIFICATION	VARCHAR(30)	収集された DDL の種別。 **DDLSTORAGE** データベース、バッファー・プール、パーティション・グループ、ストレージ・グループ、および表スペースの変更 DDL の実行。 **DDLWLM** ヒストグラム、サービス・クラス、しきい値、作業アクション・セット、作業クラス・セット、およびワークロードの DDL の実行。 **DDLMONITOR** イベント・モニターおよび使用リストの DDL の実行。 **DDLSECURITY** 監査ポリシー、権限付与、マスク、権限ロール、権限取り消し、セキュリティー・ラベル、セキュリティー・ラベル・コンポーネント、セキュリティー・ポリシー、およびトラステッド・コンテキストの DDL 実行。 **DDLSQL** 別名、関数、メソッド、モジュール、パッケージ、プロシージャー、スキーマ、シノニム、トランスフォーム、トリガー、タイプ、変数、およびビューの DDL の実行。 **DDLDATA** 索引、シーケンス、表、および一時表の DDL の実行。

表 101. 変更履歴イベント・モニターによって戻される DDLSTMTEXEC 論理データ・グループの情報。デフォルトの表名は DDLSTMTEXEC_evmon-name です。 (続き)

列名	データ・タイプ	説明
DDL_CLASSIFICATION (続き)	VARCHAR(30)	**DDLXML** XSROBJECT の DDL の実行。 **DDLFEDERATED** ニックネーム/サーバー、タイプ/ユーザー・マッピング、およびラッパーの DDL の実行。
STMT_TEXT	CLOB(2MB)	SQL ステートメントのテキスト。

TXNCOMPLETION 論理データ・グループ:

TXNCOMPLETION 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行は、1 つの完了済みトランザクション・イベントを表します。対応する DDLSTMTEXEC_evmon-name 表イベントと同じ作業単位内でコミット、ロールバック、またはセーブポイントへのロールバックが発生するたびに、行が書き込まれます。

TXNCOMPLETION_evmon-name 表内の情報を使用して、トランザクション内の DDL ステートメントがコミットされたかどうかを判別します。その後、TXNCOMPLETION_evmon-name 表の GLOBAL_TRANSACTION_ID 列、LOCAL_TRANSACTION_ID 列、および SAVEPOINT_ID 列を使用して、トランザクション完了イベントによって影響を受けた DDLSTMTEXEC_evmon-name 表内の DDL ステートメントを見つけることができます。

次の表に、変更履歴イベント・モニターが収集するトランザクション完了情報を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 102. 変更履歴イベント・モニターによって戻される TXNCOMPLETION 論理データ・グループの情報。デフォルトの表名は TXNCOMPLETIONevmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは TXNCOMPLETION です。

表 102. 変更履歴イベント・モニターによって戻される TXNCOMPLETION 論理データ・グループの情報。デフォルトの表名は TXNCOMPLETIONevmon-name です。 (続き)

列名	データ・タイプ	説明
GLOBAL_TRANSACTION_ID	VARCHAR(40)	イベントが発生した時刻で使用していたグローバル・トランザクション ID。 これは、トランザクション・ログの一部となる SQLP_GXID 構造体のデータ・フィールドです。
LOCAL_TRANSACTION_ID	VARCHAR(16)	イベントが発生した時刻で使用していたローカル・トランザクション ID。 これは、トランザクション・ログの一部となる SQLU_TID 構造体です。
SAVEPOINT_ID	BIGINT	作業単位内で設定されたセーブポイントの ID。
UOW_ID	INTEGER	アプリケーション・ハンドル内の作業単位の固有 ID。
TXN_COMPLETION_STATUS	CHAR(1)	トランザクションの状況を示します。 **C** Commit **R** ロールバック **S** セーブポイントへのロールバック

EVMONSTART 論理データ・グループ:

EVMONSTART 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表では、各行が変更履歴イベント・モニターの開始を表します。イベント・モニターの始動がシステム・パフォーマンスに直接関連するわけではありませんが、この情報によって、モニターがキャプチャーする他の情報のコンテキストが提供されます。

イベント・モニター始動イベントは、変更がいつから有効になったかを理解するのに役立ちます。例えば、活動化タイム・スタンプは、据え置きデータベースまたはデータベース・マネージャーの構成パラメーターの更新がいつ有効になったかを追跡するのに役立ちます。イベント・モニターがいつ活動化されたかがわかっていると、イベント・モニター表の中でキャプチャーされた情報の完全性を理解するのにも役立ちます。イベント・モニターが非アクティブ化されている間に発生したイベント (明示的でも暗黙的でも) があっても、それらはキャプチャーされません。データベースが活動化されていない場合、イベント・モニターは暗黙的に非アクティブになっています。

次の表に、変更履歴イベント・モニターが収集するイベント・モニター始動情報を示します。表の名前は、表にデータを設定するために使用された論理データ・グル

ープの名前と、CREATE EVENT MONITOR ステートメントでイベント・モニターに指定された名前とを連結することによって導出されます。

表 103. 変更履歴イベント・モニターによって戻される EVMONSTART 論理データ・グループの情報。デフォルトの表名は EVMONSTART_evmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは EVMONSTART です。
DB2START_TIME	TIMESTAMP	据え置きデータベース・マネージャー構成パラメーターの更新がいつ有効になったかを追跡するために使用できるデータベース・メンバー活動化タイム・スタンプ。
DB_CONN_TIME	TIMESTAMP	据え置きデータベース構成パラメーターの更新がいつ有効になったかを追跡するために使用できるデータベース活動化タイム・スタンプ。

UTILSTART 論理データ・グループ:

UTILSTART 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行は、開始されたユーティリティーを表します。

次の表に、変更履歴イベント・モニターによって収集されるユーティリティーの詳細を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 104. 変更履歴イベント・モニターによって戻される UTILSTART 論理データ・グループの情報。デフォルトの表名は UTILSTARTevmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。

表 104. 変更履歴イベント・モニターによって戻される UTILSTART 論理データ・グループの情報。デフォルトの表名は UTILSTARTevmon-name です。 (続き)

列名	データ・タイプ	説明
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは以下のいずれかです。 • UTILSTART • UTILSTARTPROC
UTILITY_INVOCATION_ID	VARCHAR(32) FOR BIT DATA	ユーティリティー呼び出しに対応するユニーク ID。
UTILITY_TYPE	VARCHAR(16)	ユーティリティーのタイプは、以下のいずれかです。 • BACKUP • LOAD • MOVETABLE • REDISTRIBUTE • REORG • RESTORE • ROLLFORWARD • RUNSTATS

表 104. 変更履歴イベント・モニターによって戻される UTILSTART 論理データ・グループの情報。デフォルトの表名は UTILSTARTevmon-name です。 (続き)

列名	データ・タイプ	説明
UTILITY_OPERATION_TYPE	CHAR(1)	UTILITY_TYPE が BACKUP の場合は、以下のいずれか。 **D** 差分 **I** 増分 **F** 完全 UTILITY_TYPE が LOAD の場合は、以下のいずれか。 **I** 挿入 **R** 置換 **S** 再起動 **T** 終了 UTILITY_TYPE が MOVETABLE の場合は、以下のいずれか。 **A** 取り消し **C** コピー **I** 初期化 **L** クリーンアップ **M** 移動 **R** 再生 **S** スワップ **V** 検証 UTILITY_TYPE が REDISTRIBUTE の場合は、以下のいずれか。 **A** 打ち切り **C** 続行 **D** デフォルト **T** ターゲット・マップ UTILITY_TYPE が REORG の場合は、以下のいずれか。 **A** すべての表索引の再編成 **I** 索引の再編成 **N** インプレース表再編成 **R** 表再利用エクステントの再編成 **T** 従来の表再編成

表 104. 変更履歴イベント・モニターによって戻される UTILSTART 論理データ・グループの情報。デフォルトの表名は UTILSTARTevmon-name です。 (続き)

列名	データ・タイプ	説明
UTILITY_OPERATION_TYPE (続き)	CHAR(1)	UTILITY_TYPE が RESTORE の場合は、以下のいずれか。 **A** 自動増分 **B** 増分打ち切り **F** 完全 **M** 手動増分 UTILITY_TYPE が ROLLFORWARD の場合は、以下のいずれか。 **E** ログの最後 **P** 特定の時点 UTILITY_TYPE が RUNSTATS の場合は、以下のいずれか。 **A** 表にあるすべての索引 **I** 索引 **T** 表
UTILITY_INVOKER_TYPE	VARCHAR(4)	ユーティリティーが起動された方法を示します。以下のいずれか • AUTO • USER
UTILITY_PRIORITY	INTEGER	スロットルされたピアに関連したスロットル・ユーティリティーの相対的な重要度を指定します。優先度の範囲は 0 から 100 までで、0 は、ユーティリティーがスロットルされずに実行されることを意味します。
UTILITY_START_TYPE	VARCHAR(8)	ユーティリティーが開始された方法を示します。以下のいずれか • RESUME • START

表 104. 変更履歴イベント・モニターによって戻される UTILSTART 論理データ・グループの情報。デフォルトの表名は UTILSTARTevmon-name です。 (続き)

列名	データ・タイプ	説明
OBJECT_TYPE	VARCHAR(16)	ユーティリティーの操作対象オブジェクトのタイプ。以下のいずれか • DATABASE • INDEX • PARTITIONGROUP • TABLE • TABLESPACE これは objtype モニター・エレメントのシノニムです。
OBJECT_SCHEMA	VARCHAR(128)	OBJECT_TYPE が INDEX または TABLE の場合には索引または表のスキーマで、それ以外の場合は空のストリング。
OBJECT_NAME	VARCHAR(128)	OBJECT_TYPE が INDEX、PARTIONGROUP、または TABLE の場合には、索引、パーティション・グループ、または表の名前。
NUM_TBSPS	INTEGER	OBJECT_TYPE が DATABASE または TABLESPACE の場合には、表スペースの数。
TBSP_NAMES	CLOB(5M)	OBJECT_TYPE が DATABASE または TABLESPACE の場合には、ユーティリティーの実行対象となる表スペースの名前の、コンマで区切られたリスト。

表 104. 変更履歴イベント・モニターによって戻される UTILSTART 論理データ・グループの情報。デフォルトの表名は UTILSTARTevmon-name です。 (続き)

列名	データ・タイプ	説明
UTILITY_DETAIL	CLOB(2M)	ユーティリティーが実行している作業の要旨。ユーティリティーに指定されている一部のオプションを含みます。例えば、REORG を起動するためのレコードには、部分的に再構成されたコマンド・ストリングが含まれます。これには、アクセス・モードなど、ユーティリティーが使用するさまざまなオプションの一部が含まれます。このフィールドのフォーマットはユーティリティーのタイプに依存し、リリースごとに変更される可能性があります。

UTILLOCATION 論理データ・グループ:

UTILLOCATION 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行は、ユーティリティーの開始に関連するそれぞれのファイルまたはパスを表します。

次の表に、変更履歴イベント・モニターによって収集されるユーティリティーの詳細を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 105. 変更履歴イベント・モニターによって戻される UTILLOCATION 論理データ・グループの情報。デフォルトの表名は UTILLOCATION_evmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは UTILSTART です。
UTILITY_INVOCATION_ID	VARCHAR(32) FOR BIT DATA	ユーティリティー呼び出しに対応するユニーク ID。

表 105. 変更履歴イベント・モニターによって戻される UTILLOCATION 論理データ・グループの情報。デフォルトの表名は UTILLOCATION_evmon-name です。 (続き)

列名	データ・タイプ	説明
UTILITY_TYPE	VARCHAR(16)	ユーティリティーのタイプは、以下のいずれかです。 • BACKUP • LOAD • RESTORE • ROLLFORWARD
DEVICE_TYPE	CHAR(1)	UTILSTART イベントに関連するデバイス・タイプの ID。このフィールドにより、LOCATION フィールドの解釈が決まります。以下のいずれか **A** TSM **C** クライアント **D** ディスク **F** スナップショット・バックアップ **L** ローカル **N** DB2 によって内部で生成される **O** その他のベンダー・デバイス・サポート **P** パイプ **Q** カーソル **R** フェッチ・データの削除 **S** サーバー **T** テープ **U** ユーザー出口 **X** X/Open XBSA インターフェース

表 105. 変更履歴イベント・モニターによって戻される UTILLOCATION 論理データ・グループの情報。デフォルトの表名は UTILLOCATION_evmon-name です。 (続き)

列名	データ・タイプ	説明
LOCATION_TYPE	CHAR(1)	ロケーションの使用目的の説明。 UTILITY_TYPE が LOAD の場合は、以下のいずれか。 **C** コピー・ターゲット **D** 入力データ **L** LOB パス **X** XML パス UTILITY_TYPE が BACKUP の場合は、以下のいずれか。 **B** バックアップのターゲット・ロケーション UTILITY_TYPE が RESTORE の場合は、以下のいずれか。 **S** リストアのソース・ロケーション UTILITY_TYPE が ROLLFORWARD の場合は、以下のいずれか。 **O** ROLLFORWARD DATABASE コマンドの一部としてキャプチャーされた代替オーバーフロー・ログ・パス。なお、デフォルトのオーバーフロー・ログ・パスを使用する場合、ロケーション・レコードはキャプチャーされません。 上記以外の場合には、ブランク文字になります。
LOCATION	VARCHAR(1024)	イベントに関連したロケーション。ロケーションは、UTILITY_TYPE によって異なります。例えば、ロード入力ファイル、またはバックアップのターゲット・パス名など。

UTILSTOP 論理データ・グループ:

UTILSTOP 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行は、停止済みのユーティリティーを表します。

次の表に、変更履歴イベント・モニターによって収集されるユーティリティーの詳細を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 106. 変更履歴イベント・モニターによって戻される UTILSTOP 論理データ・グループの情報。デフォルトの表名は UTILSTOP_evmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは以下のいずれかです。 • UTILSTOP • UTILSTOPPROC
UTILITY_INVOCATION_ID	VARCHAR(32) FOR BIT DATA	ユーティリティー呼び出しに対応するユニーク ID。
UTILITY_TYPE	VARCHAR(16)	ユーティリティーのタイプは、以下のいずれかです。 • BACKUP • LOAD • MOVETABLE • REDISTRIBUTE • REORG • RESTORE • ROLLFORWARD • RUNSTATS
UTIL_STOP_TYPE	VARCHAR(8)	ユーティリティーが停止された方法を示します。以下のいずれか • PAUSE • STOP
START_EVENT_ID	BIGINT	対応する UTILSTART イベントまたは UTILSTARTPROC イベントの固有 ID。START_EVENT_TIMESTAMP およびメンバー・エレメントと併用して、停止レコードを、対応する開始レコードと関連付けます。

表 106. 変更履歴イベント・モニターによって戻される UTILSTOP 論理データ・グループの情報。デフォルトの表名は UTILSTOP_evmon-name です。 (続き)

列名	データ・タイプ	説明
START_EVENT_TIMESTAMP	TIMESTAMP	対応する UTILSTART イベントまたは UTILSTARTPROC イベントの時刻。 START_EVENT_ID およびメンバー・エレメントと併用して、停止レコードを、対応する開始レコードと関連付けます。
SQLCA (SQL 連絡域) (*SQL リファレンス 第 1 巻* を参照)	VARCHAR(8)	SQL 連絡域 (SQLCA) の始まりを識別するストリング。
SQLABC	INTEGER	SQL 連絡域 (SQLCA) の長さ。
SQLCODE	INTEGER	SQLCA 構造内の、最後に実行された SQL ステートメントの SQL 戻りコード。
SQLERRM	VARCHAR(72)	SQLCA 構造内の 1 つ以上のトークン。各トークンは X'FF' で区切られ、エラー状態に関する具体的な情報を提供するエラー・メッセージ内の変数に代入されます。
SQLERRP	VARCHAR(8)	SQLCA 構造内の、製品を示す 3 文字の ID。その後に、製品のバージョン、リリース、および修正レベルを示す 5 つの英数字が続きます。
SQLERRD1	INTEGER	SQLCA (SQL 連絡域) を参照。
SQLERRD2	INTEGER	SQLCA (SQL 連絡域) を参照。
SQLERRD3	INTEGER	SQLCA (SQL 連絡域) を参照。
SQLERRD4	INTEGER	SQLCA (SQL 連絡域) を参照。
SQLERRD5	INTEGER	SQLCA (SQL 連絡域) を参照。
SQLERRD6	INTEGER	SQLCA (SQL 連絡域) を参照。
SQLWARN	VARCHAR(11)	SQLCA 構造内の一連の警告標識。それぞれの標識はブランクか W です。
SQLSTATE	VARCHAR(5)	SQLCA 構造内の、最後に実行された SQL ステートメントの結果を示す戻りコード。

UTILPHASE 論理データ・グループ:

UTILPHASE 論理データ・グループの表は、変更履歴イベント・モニターによって生成されます。この表の各行には、開始または停止されるユーティリティーに関する情報が含まれています。

ユーティリティーの実行は、いくつかのフェーズつまり処理段階に分かれています。現在のところ、変更履歴イベント・モニターは、表スペース・バックアップの開始フェーズおよび停止フェーズのみキャプチャーします。

次の表に、変更履歴イベント・モニターによって収集されるユーティリティーのフェーズの詳細を示します。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントでイベント・モニターに指定した名前が連結されて生成されます。

表 107. 変更履歴イベント・モニターによって戻される UTILPHASE 論理データ・グループの情報。デフォルトの表名は UTILPHASE_evmon-name です。

列名	データ・タイプ	説明
PARTITION_KEY	INTEGER	イベント・モニター表のパーティション・キー。
EVENT_ID	BIGINT	イベントに関連したユニークなトークン。
EVENT_TIMESTAMP	TIMESTAMP	イベントが生成された時刻。
MEMBER	SMALLINT	イベントが発生したメンバー。
EVENT_TYPE	VARCHAR(32)	発生したイベントのタイプ。この論理データ・グループの場合、タイプは以下のいずれかです。 • UTILPHASESTART • UTILPHASESTOP
UTILITY_INVOCATION_ID	VARCHAR(32) FOR BIT DATA	ユーティリティー呼び出しに対応するユニーク ID。
UTILITY_TYPE	VARCHAR(16)	開始または停止されるユーティリティーのタイプ。この論理データ・グループの場合、タイプは BACKUP です。
UTILITY_PHASE_TYPE	VARCHAR(16)	UTILITY_TYPE が BACKUP の場合、フェーズ・タイプは以下のとおりです。 **BACKUPTS** バックアップ表スペース
PHASE_START_EVENT_ID	BIGINT	EVENT_TYPE が UTILPHASESTOP の場合、これは対応する UTILPHASESTART の EVENT_ID です。それ以外の場合は -1 です。PHASE_START_EVENT _TIMESTAMP およびメンバー・エレメントと併用して、フェーズ停止レコードを対応するフェーズ開始レコードと関連付けます。
PHASE_START_EVENT_TIMESTAMP	TIMESTAMP	EVENT_TYPE が UTILPHASESTOP の場合、これは対応する UTILPHASESTART の時刻です。それ以外の場合は空です。PHASE_START_EVENT_ID およびメンバー・エレメントと併用して、フェーズ停止レコードをフェーズ開始レコードに関連付けます。

表 107. 変更履歴イベント・モニターによって戻される UTILPHASE 論理データ・グループの情報。デフォルトの表名は UTILPHASE_evmon-name です。 (続き)

列名	データ・タイプ	説明
OBJECT_TYPE	VARCHAR(16)	ユーティリティーの操作対象オブジェクトのタイプ。このタイプは TABLESPACE です。 これは objtype モニター・エレメントのシノニムです。
OBJECT_SCHEMA	VARCHAR(128)	将来の利用のために予約済み
OBJECT_NAME	VARCHAR(128)	OBJECT_TYPE が TABLESPACE の場合は、表の名前。
UTILITY_PHASE_DETAIL	CLOB(2M)	将来の利用のために予約済み

変更履歴イベント・モニターを使用したユーティリティー履歴のモニター

変更履歴イベント・モニターは、ユーティリティーの実行に関するいくつかのイベントをキャプチャーできます。それらのイベントを使用すると、データベース・サーバーのユーティリティー実行履歴をモニターできます。

このイベント履歴は、いくつかの論理データ・グループに書き込まれます。各論理データ・グループには、関連付けられたイベント・モニター表が 1 つ存在します。

1 つのユーティリティーの実行で、変更履歴イベント・モニターに 1 つ以上のイベントが生成される場合があります。例えば、REORG ユーティリティーの実行では、 REORG 操作の開始と終了を示す 2 つのイベントが生成されます。イベントと論理データ・グループは、1 対多でマッピングされます。 1 つのイベントで、複数の論理データ・グループに情報が書き込まれる場合があります。また、 1 つのイベントで、特定の論理データ・グループに関連付けられている表に、複数のエントリー (行) が書き込まれる場合があります。特定のユーティリティー呼び出しに対応する各イベントは、 *utility_invocation_id* エレメントで識別されます。*utility_invocation_id* は、特定のユーティリティー呼び出しを一意的に識別するバイナリー・トークンです。*utility_invocation_id* は、ユーティリティーを実行している全メンバーで同じです。*utility_invocation_id* の固有性は、データベースの非アクティブ化、再活動化、およびメンバーのシャットダウンが行われても維持されます。このため、特定のユーティリティー呼び出しに対応するすべてのイベント・モニター・レコードを迅速に識別することができます。他のフィールドと結合する必要もなければ、重複 ID について懸念する必要もありません。

utility_invocation_id を使用すると、特定のユーティリティー呼び出しに関するすべてのイベントを識別できます。例えば、 REORG コマンドを表に対して実行した場合、ユーティリティーが実行を開始したときに UTILSTART イベントが生成され、ユーティリティーが実行を完了したときに UTILSTOP イベントが生成されます。この UTILSTART および UTILSTOP イベントの両方は、同じ REORG コマンド呼び出しに関するものであるため、同じ *utility_invocation_id* を持つことになります。これらのイベントを *utility_invocation_id* で結合すると、ユーティリティーの経過時間を計算できます。

変更履歴イベント・モニターは、以下のユーティリティー・タイプの実行をモニターできます。

- BACKUP
- LOAD
- MOVETABLE
- REDISTRIBUTE
- REORG
- RESTORE
- ROLLFORWARD
- RUNSTATS

変更履歴イベント・モニターは、オフラインのバックアップ、リストア、およびロールフォワードの実行はキャプチャーしません。ユーティリティー・イベントがキャプチャーされるのは、そのユーティリティーの実行時に変更履歴イベント・モニターがアクティブである場合に限られることに注意してください。ユーティリティーの実行前にイベント・モニターが非アクティブ化された場合、そのユーティリティーの実行に関するイベントはキャプチャーされません。例えば、イベント・モニターのターゲット表が存在する表スペースに対して、ユーティリティーが排他的アクセスを必要とする場合などがあります。

以下の表に、ユーティリティー実行イベントに関連する、変更履歴イベント・モニターの論理データ・グループおよび関連表をリストします。表名は、表にデータを設定するために使用される論理データ・グループの名前と、 CREATE EVENT MONITOR ステートメントによって作成したときにイベント・モニターに指定した名前が連結されて生成されます。ここに示している表名は、 CREATE EVENT MONITOR ステートメントの一部として表名が指定されなかった場合のデフォルトの表名です。

表 108. ユーティリティーの実行時にデータが設定される論理データ・グループ

論理データ・グループ	デフォルトの表名	内容
CHANGESUMMARY	CHANGESUMMARY_evmon-name (478 ページの『CHANGESUMMARY 論理データ・グループ』を参照)	変更履歴イベント・モニターによってキャプチャーされたすべてのイベントの要約
UTILSTART	UTILSTART_evmon-name (490 ページの『UTILSTART 論理データ・グループ』を参照)	ユーティリティーの開始情報
UTILLOCATION	UTILLOCATION_evmon-name (495 ページの『UTILLOCATION 論理データ・グループ』を参照)	ユーティリティーのパスまたはファイル情報
UTILSTOP	UTILSTOP_evmon-name (497 ページの『UTILSTOP 論理データ・グループ』を参照)	ユーティリティーの停止情報
UTILPHASE	UTILPHASE_evmon-name (499 ページの『UTILPHASE 論理データ・グループ』を参照)	ユーティリティーのフェーズ情報

CREATE EVENT MONITOR (変更履歴) ステートメントの WHERE EVENT IN 節は、変更履歴イベント・モニターでモニターするユーティリティーを制御します。以下のリストに、各制御によって、どのユーティリティーのキャプチャーが有効になるかを示します。

UTILALL
ロード、表移動、オンライン・バックアップ、オンライン・リストア、オンライン・ロールフォワード、再配分、REORG、および RUNSTATS ユーティリティーの実行をキャプチャーします。

BACKUP
オンライン・バックアップ・ユーティリティーの実行をキャプチャーします。

LOAD ロード・ユーティリティーの実行をキャプチャーします。

MOVETABLE
表移動ユーティリティー (ADMIN_MOVE_TABLE ストアード・プロシージャーの呼び出し) の実行をキャプチャーします。

REDISTRIBUTE
パーティション・グループ再配分ユーティリティーの実行をキャプチャーします。

REORG
REORG ユーティリティーの実行をキャプチャーします。

RESTORE
オンライン・リストア・ユーティリティーの実行をキャプチャーします。

ROLLFORWARD
オンライン・ロールフォワード・ユーティリティーの実行をキャプチャーします。

RUNSTATS
RUNSTATS ユーティリティーの実行をキャプチャーします。

変更履歴イベント・データの収集

変更履歴イベント・モニターを使用して、データベースとデータベース管理システムのパフォーマンス、動作、および安定性に影響を与える可能性があるアクティビティーに関する情報を収集することができます。

始める前に

変更履歴イベント・モニターを作成し、変更履歴イベント・モニター・データを収集するには、 DBADM または SQLADM 権限が必要です。

このタスクについて

変更履歴イベント・モニターは、標準的なデータベース・ワークロードの実行に影響を与える可能性がある変更をキャプチャーします。通常のワークロードにパフォーマンスの低下が発生したり、予期しない動作が見られた場合は、問題を引き起こすような変更が行われていないか調べる必要があります。各変更関連イベントは、次の 3 つのキー・フィールドによって一意的に識別されます。

イベント・タイム・スタンプ
イベントが発生した時刻。

イベント ID
イベント・タイム・スタンプが共通している場合に、固有性を確保するための数値トークン。

メンバー
イベントが発生したデータベース・マネージャー・プロセス。イベントのタイム・スタンプおよびイベント ID は、メンバー内でのみユニークであるため、メンバーによってグローバルな固有性が確保されます。

すべての論理グループには、これらの 3 つのフィールドが含まれており、同じイベントに対応するレコードまたは行はすべて、これらのフィールドに同じ値が入っています。これらの共通する値によって、異なる論理データ・グループ間で情報を結合することが容易になります。別のメンバーでのユーティリティー操作および構成パラメーター更新は、別のイベントとしてキャプチャーされるため、これらのキー・フィールドは異なる値になります。

制約事項

変更履歴イベント・モニター・データは、論理データ・グループに関連付けられている表にのみ書き込むことができます。変更履歴イベント・モニターは、未フォーマット・イベント表、ファイル、および名前付きパイプには書き込めません。

手順

データベースのパフォーマンス、動作、または安定性に影響を与えている可能性があるアクティビティーに関する詳細情報を収集するには、以下の手順を実行します。

1. 対象とする変更履歴イベントを決定します。変更履歴イベント・モニターは、以下に関するイベントをキャプチャーすることができます。
   - 構成パラメーターの変更
   - レジストリー変数の変更
   - DDL 実行
   - コミット、ロールバック、またはセーブポイントまでのロールバックの発生
   - イベント・モニターの開始情報
   - ユーティリティーの開始情報
   - ユーティリティーのパスまたはファイル情報
   - ユーティリティーの停止情報
   - ユーティリティーのフェーズ情報
2. CREATE EVENT MONITOR FOR CHANGE HISTORY ステートメントを使用して、 `whats_changed` という変更履歴イベント・モニターを作成します。WHERE EVENT IN 節を使用して、キャプチャーする変更履歴イベントを指定します。以下の例では、すべてのイベント・タイプをキャプチャーする変更履歴イベント・モニターを作成する方法を示しています。

```
CREATE EVENT MONITOR whats_changed
   FOR CHANGE HISTORY WHERE EVENT IN (ALL)
   WRITE TO TABLE
```

3. 以下のステートメントを実行して、whats_changed という変更履歴イベント・モニターを活動化します。

```
SET EVENT MONITOR whats_changed STATE 1
```

タスクの結果

変更履歴イベント・モニターがアクティブである (データベース構成の更新など) 間に変更履歴イベントが発生した場合は、必ず、そのイベントに関する情報が、変更履歴イベント・モニターの表にキャプチャーされます。イベント・モニターの WHERE EVENT IN 節に指定したイベントのみが、変更履歴イベント・モニターでキャプチャーされます。

例

例: 変更履歴イベント・モニターを使用したロック・エスカレーションの増加の調査:

変更履歴イベント・モニターを使用して、データベースのパフォーマンスの低下を引き起こした可能性がある変更を突きとめることができます。

シナリオ

この例では、データベースのパフォーマンスが低下したとユーザーが報告しています。データベース管理者 (DBA) は、この 24 時間に発生したロック・エスカレーションの数が異常に多いことに気が付きました。また、DBA は、この同じ期間内のアプリケーション・ロック待機時間にも、類似した増加が見られることに気が付きました。

DBA は、変更履歴イベント・モニターを使用して、構成変更、索引変更、および LOAD 操作のモニターを行っています。このイベント・モニターは、以下のステートメントを使用して作成しました。

```
CREATE EVENT MONITOR CFGHIST
FOR CHANGE HISTORY WHERE EVENT IN (DBCFG, DBMCFG, DBCFGVALUES,
     DBMCFGVALUES,REGVAR,REGVARVALUES, DDLDATA, LOAD)
WRITE TO TABLE
```

このイベント・モニターを活動化するために使用したステートメントは以下のとおりです。

```
SET EVENT MONITOR CFGHIST STATE=1
```

以下の表に、 CFGHIST 変更履歴イベント・モニターが表 CHANGESUMMARY_CFGHIST に書き込む可能性があるイベント・モニター・データの例を示します。すべての変更履歴イベント・モニターは、CHANGESUMMARY 論理データ・グループに書き込みます。 CHANGESUMMARY 論理データ・グループで説明しているように、CHANGESUMMARY 論理データ・グループは、キャプチャーされたイベントを要約したいくつかのイベント・モニター・エレメントを戻します。以下の出力は、それらのエレメントのサブセットのみを示しています。表の名前は、表 (CHANGESUMMARY) にデータを設定するために使用された論理データ・グループの名前と、CREATE EVENT MONITOR ステートメント (CFGHIST) でイベント・モニターに指定された名前とを連結することによって導出されます。

```
APPL_ID                       APPL_NAME .... EVENT_ID EVENT_TIMESTAMP
----------------------------- --------- .... -------- ------------------
*LOCAL.tripathy.111028110756 db2bp     .... 1        28/10/2011 07:12:02

EVENT_TYPE MEMBER ....
---------- ------ ....
EVMONSTART 0      ....
```

これまでパフォーマンスに問題はなかったため、 DBA は、最近行われた何らかの変更によって問題が生じているのではないかと考え、以下の手順を実行しました。

1. この 24 時間に行われた変更がないか、CHANGESUMMARY 論理データ・グループを確認しました。この例では、現在時刻は 2011/10/31 06:00:00 だとします。

   ```
   SELECT EVENT_TYPE FROM CHANGESUMMARY_CFGHIST
      WHERE EVENT_TIMESTAMP > CURRENT TIMESTAMP - 24 HOURS
   ```

   この照会は、以下の結果を返します。

   ```
   EVENT_TYPE
   ----------
   DBCFG
   DBCFG
   ```

   出力に、この 24 時間にデータベース構成の更新が 2 回あったことが示されます。

2. DBDBMCFG 論理データ・グループを照会して、それらの構成変更の詳細情報を取得しました。

3. ```
   SELECT EVENT_TIMESTAMP, CFG_NAME, CFG_VALUE, CFG_OLD_VALUE, DB_DEFERRED
      FROM DBDBMCFG_CHGHIST
   ```

   この照会は、以下の結果を返します。

   ```
   EVENT_TIMESTAMP       CFG_NAME      CFG_VALUE   CFG_OLD_VALUE   DB_DEFERRED
   -------------------   -----------   ---------   -------------   -----------
   30/10/2011 08:41:39   LOCKLIST           1024            2048   N
   30/10/2011 08:42:35   LOCKTIMEOUT           0              -1   Y
   ```

   出力に、パフォーマンスが低下した期間内にロッキングの変更が行われたことが示されています。

DBA は、LOCKTIMEOUT 変更が据え置かれたことに気が付き、この構成変更の後にデータベースが活動化されたかどうかを確認するための照会を実行しました。この確認により、構成変更がデータベースに反映されたかどうかが分かります。変更が反映されていなかったのであれば、その変更がパフォーマンス問題の原因である可能性はなくなります。データベースの活動化時刻は、EVMONSTART 論理データ・グループに記録されています。すべての変更履歴イベント・モニターは、デフォルトでは EVMONSTART 論理データ・グループに書き込みます。

```
SELECT COUNT (*)as POST_CFG_ACTIVATIONS FROM EVMONSTART_CHGHIST
   WHERE DB_CONN_TIME > TIMESTAMP(2011-10-30-08:42:35)
```

照会を実行すると、ゼロ以外の値が戻されました。

```
POST_CFG_ACTIVATIONS
--------------------
                   1
```
```

このゼロ以外の値により、 LOCKTIMEOUT 構成パラメーターの変更後にデータベースが活動化されたことが確認できました。つまり、新しい値は有効になっています。これで、システム上で何が変更されたかが明らかになり、DBA は、ロック関連の構成パラメーターを元の値に戻して、問題が解決するかどうかを試すことができます。

注: 構成パラメーターの変更時に変更履歴イベント・モニターが非アクティブであった場合、変更履歴イベント・モニターはその DBCFG イベントをキャプチャーしません。その代わり、変更履歴イベント・モニターは、開始されたときに DBCFGVALUES イベントをキャプチャーします。 DBDBMCFG 論理データ・グループの各行は、 DBCFG または DBMCFG イベントの一環として更新された構成パラメーター、または、 DBCFGVALUES または DBMCFGVALUES イベントの一環としてイベント・モニターの開始時にキャプチャーされた構成パラメーターを表します。 CFG_COLLECTION_TYPE モニター・エレメントは、レコードの内容が、構成パラメーターの更新に関するものか、あるいはイベント・モニターの開始時に記録された初期値に関するものかを示します。DBA は、問題を引き起こした可能性がある変更値を探すときには、現行の変更履歴イベント・モニターの開始時にキャプチャーされた値と、その前にキャプチャーされた値とを比較する必要があります。診断ログを調べることも有用です。

例: 変更履歴イベント・モニターを使用した構成変更およびユーティリティー実行の特定:

変更履歴イベント・モニターを使用して、構成変更またはユーティリティー実行が最近行われたかどうかを調べることができます。

シナリオ

この例では、データベース管理者 (DBA) が、この 24 時間にデータベースのパフォーマンスが変化したことに気が付きました。 DBA は、HIST という変更履歴イベント・モニターを既に作成しており、データベースおよびデータベース管理システムの動作、パフォーマンス、または安定性の変化について調査するときに使用してきました。

DBA は、この 24 時間に行われた変更イベントまたはユーティリティー実行を要約するために、CHANGESUMMARY 論理データ・グループに対して以下の照会を実行しました。

```
SELECT EVENT_TIMESTAMP,
       EVENT_TYPE,
       UTILITY_TYPE,
       COORD_MEMBER,
       MEMBER
  FROM CHANGESUMMARY_HIST
  WHERE EVENT_TIMESTAMP > CURRENT TIMESTAMP - 24 HOURS
  ORDER BY BY EVENT_TIMESTAMP ASC
```

この照会によって、以下のような出力が戻されたとします。

```
EVENT_TIMESTAMP            EVENT_TYPE     UTILITY_TYPE COORD_MEMBER MEMBER
-------------------------- -------------- ------------ ------------ ------
2010-10-31-17.29.04.545210 DBCFG                                   0      0
2010-10-31-18.29.04.545210 UTILSTART      LOAD                     0      0
2010-10-31-18.40.04.545210 UTILSTARTPROC  LOAD                     0      0
2010-10-31-18.50.04.545210 UTILSTOPPROC   LOAD                     0      0
2010-10-31-18.40.04.545210 UTILSTARTPROC  LOAD                     0      1
```

```
2010-10-31-18.50.04.545210 UTILSTOPPROC   LOAD                    0      1
2010-10-31-19.29.04.545210 UTILSTOP       LOAD                    0      0
2010-10-31-19.56.04.545210 UTILSTART      BACKUP                  0      0
2010-10-31-20.09.04.545210 UTILPHASESTART BACKUP                  0      0
2010-10-31-20.29.04.545210 UTILPHASESTOP  BACKUP                  0      0
2010-10-31-21.29.04.545210 UTILSTOP       BACKUP                  0      0

9 record(s) selected.
```

この出力から、DBA は、この 24 時間に構成変更およびユーティリティー実行がいくつか行われたことを突き止めます。次に、変更履歴イベント・モニターの他の論理データ・グループを照会して、この CHANGESUMMARY 論理データ・グループで戻されたイベントに関する詳細情報を取得することができます。例えば、UTILSTART イベントに関する詳細を取得する場合は、UTILSTART 論理データ・グループを照会すると、ユーティリティーの操作対象となったオブジェクト、およびユーティリティーの開始時に使用されたオプションを調べることができます。

例: 変更履歴イベント・モニターを使用した LOAD 操作のリスト:

変更履歴イベント・モニターを使用して、データベースで実行されたすべての LOAD 操作を追跡することができます。

シナリオ

この例では、データベース管理者 (DBA) は、データベース上のすべてのロード・ユーティリティー実行の履歴をキャプチャーしてリストすることが必要です。 LOAD ユーティリティー・イベントを追跡するには、以下のようにします。

1. LOAD イベントを追跡する変更履歴イベント・モニターを作成します。例えば、以下のようにします。

```
CREATE EVENT MONITOR MON_LOAD
    FOR CHANGE HISTORY WHERE EVENT IN (LOAD)
    WRITE TO TABLE
       CHANGESUMMARY (TABLE UTIL_COMMON),
       UTILSTART (TABLE LOAD_START),
       UTILSTOP (TABLE LOAD_STOP)
       UTILLOCATION (TABLE LOAD_INPUT_FILES)
       UTILPHASE (TABLE LOAD_PHASES);
```

2. イベント・モニターを活動化します。

```
 SET EVENT MONITOR MON_LOAD STATE=1
```

3. データベースで実行された LOAD 操作に関する情報を、論理データ・グループで照会します。例えば、以下の照会を行うと、実行されたすべてのロード・ユーティリティーの開始時刻および終了時刻がリストされます。この照会は、コーディネーターの開始時刻および終了時刻のみを表示します。ユーティリティーの実行にかかった合計経過時間を示すために、停止レコードおよび再開レコードは無視します。

```
 SELECT A.APPL_ID,
        A.COORD_MEMBER,
        A.EVENT_TIMESTAMP AS START_TIME,
        B.EVENT_TIMESTAMP AS STOP_TIME,
        A.TABLE_SCHEMA,
        A.TABLE_NAME,
        SQLCODE,
        VARCHAR(A.UTILITY_DETAIL, 200) AS DETAIL
    FROM LOAD_START AS A
         LOAD_STOP AS B
```

```
     UTIL_COMMON AS C
WHERE A.UTILITY_INVOCATION_ID = B.UTILITY_INVOCATION_ID AND
      A.UTILITY_START_TYPE = 'START' AND
      B.UTILITY_STOP_TYPE = 'STOP' AND
      A.MEMBER = B.MEMBER AND
      A.MEMBER = A.COORD_MEMBER
ORDER BY A.EVENT_TIMESTAMP ASC
```

照会結果には、 2 つのロード・ユーティリティーが実行されたことが示されています。また、それらの開始時刻および終了時刻の詳細、ロードのターゲット (表名)、および実行されたロードの詳細も示しています。

```
APPL_ID                  START_TIME                 STOP_TIME
------------------------ -------------------------- --------------------------
*LOCAL.test.110131213809 2010-10-31-17.29.04.545210 2010-10-31-17.29.04.545210
*LOCAL.test.110131213809 2010-10-31-17.29.04.545210 2010-10-31-17.29.04.545210

TABLES_SCHEMA TABLE_NAME SQLCODE DETAIL
------------- ---------- ------- --------.....
TEST          T1               0 LOAD CURSOR..
TEST          T3               0 LOAD DEL...

2 record(s) selected.
```

例: 変更履歴イベント・モニターを使用したユーティリティー実行履歴の報告:

変更履歴イベント・モニターを使用して、データベースで実行されたユーティリティー操作を追跡することができます。

シナリオ

この例では、データベース管理者 (DBA) は、データベース上のユーティリティー・イベント実行をキャプチャーしてリストすることが必要です。ユーティリティー・イベントを報告するには、以下のようにします。

1. ユーティリティー・イベントを追跡する変更履歴イベント・モニターを作成します。例えば、以下のようにします。

   ```
   CREATE EVENT MONITOR MON_UTIL
       FOR CHANGE HISTORY WHERE EVENT IN (UTILALL)
       WRITE TO TABLE
          CHANGESUMMARY (TABLE UTIL_COMMON),
          UTILSTART (TABLE UTIL_START),
          UTILSTOP (TABLE UTIL_STOP)
          UTILLOCATION (TABLE UTIL_LOCATION)
          UTILPHASE (TABLE UTIL_PHASES) AUTOSTART;
   ```

2. イベント・モニターを使用可能にします。

   ```
   SET EVENT MONITOR MON_UTIL STATE=1
   ```

3. データベースで実行されたユーティリティー操作に関する情報を、論理データ・グループで照会します。例えば、以下の照会を行うと、メンバー単位の各ユーティリティーの呼び出しの履歴がリストされます。

   ```
   WITH UTIL_HIST(TIMESTAMP, UTIL_TYPE, ACTION, PHASE_TYPE, UTILITY_INVOCATION_ID,
      MEMBER, SQLCODE) AS
    (SELECT A.EVENT_TIMESTAMP,
           A.UTILITY_TYPE,
           CAST('START' AS VARCHAR(32)),
           CAST(NULL AS VARCHAR(16)),
           A.UTILITY_INVOCATION_ID,
           A.MEMBER,
           CAST(NULL as INTEGER)
     FROM UTIL_START AS A
   ```

```
 UNION ALL
  SELECT A.EVENT_TIMESTAMP,
         A.UTILITY_TYPE,
         CASE WHEN EVENT_TYPE IN ('UTILPHASESTART') THEN
            CAST('PHASE START' AS VARCHAR(32))
         ELSE
            CAST('PHASE STOP' AS VARCHAR(32))
         END CASE,
         CAST(UTILITY_PHASE_TYPE AS VARCHAR(16)),
         A.UTILITY_INVOCATION_ID,
         A.MEMBER,
         CAST(NULL as INTEGER)
  FROM UTIL_PHASE AS B
 UNION ALL
  SELECT A.EVENT_TIMESTAMP,
         A.UTILITY_TYPE,
         CAST('STOP' AS VARCHAR(32))
         CAST(NULL AS VARCHAR(16)),
         A.UTILITY_INVOCATION_ID,
         A.MEMBER,
         A.SQLCODE
  FROM UTIL_STOP AS C)
SELECT * FROM UTIL_HIST
ORDER BY UTILITY_INVOCATION_ID, MEMBER, TIMESTAMP ASC
```

結果のユーティリティー・イベント・レポートは、以下の目的に使用できます。

- オーバーラップしているユーティリティーを特定する。例えば、あるパーティションでユーティリティーが停止する前に、同じパーティションで別のユーティリティーが開始された場合などです。
- ユーティリティーのどこで時間がかかっているのか調べる。例えば、各フェーズの消費時間などです。注: バージョン 10.1 では、オンライン・バックアップの表スペース・バックアップ・フェーズに関してのみ調べることができます。

```
TIMESTAMP                  UTIL_TYPE ACTION      PHASE_TYPE    UTILITY_INVOCATION_ID MEMBER SQLCODE
-------------------------- --------- ----------  ------------- --------------------- ------ -------
2010-10-31-17.29.04.545210 LOAD      START       -             x'18A901F...621'      0      -
2010-10-31-17.50.04.344230 LOAD      STOP        -             x'18A901F...621'      0      0
2010-10-31-17.29.04.545211 LOAD      START       -             x'18A901F...633'      1      -
2010-10-31-17.50.04.344229 LOAD      STOP        -             x'18A901F...633'      1      0
2010-10-31-17.29.04.344210 BACKUP    START       -             x'18A901F...645'      0      -
2010-10-31-17.50.04.344211 BACKUP    PHASE START BACKUPTS      x'18A901F...645'      0      0
2010-10-31-17.51.04.545214 BACKUP    PHASE STOP  BACKUPTS      x'18A901F...645'      0      -
2010-10-31-17.52.04.344218 BACKUP    STOP        -             x'18A901F...645'      0      0
8 record(s) selected.
```

例: 変更履歴イベント・モニターを使用したコミット済みの全 DDL ステートメントのリスト:

変更履歴イベント・モニターを使用すると、実行されたコミット済みの全 DDL ステートメントをすぐにリストして、ワークロードに影響を与えるような変更が行われたかどうかを調べることができます。

シナリオ

この例では、データベース管理者 (DBA) が、この 24 時間でいくつかの照会のパフォーマンスが低下していることに気が付きます。 DBA は、ワークロードに重大な影響を与えるような変更 (索引がドロップされたなど) が行われたかどうかを調べるために、変更履歴イベント・モニターを使用して、この期間に実行された DDL をただちに確認します。 DBA は、CHGHIST という変更履歴イベント・モニターを既に作成しており、これを使用して DDL ステートメントを追跡しています。 DBA は、以下のステートメントを実行して、この 24 時間に変更履歴イベント・モニタ

ーでキャプチャーされたコミット済みの全 DDL ステートメントをリストします。実行するこのステートメントは、ROLLBACK ステートメントおよび ROLLBACK TO SAVEPOINT ステートメントのいずれかを使用してロールバックされたステートメントを除外しています。

変更履歴イベント・モニターは、 DDL が実行されたときに DDL イベントを記録することに注意してください。 DDL がデータベースの変更を招くかどうかは、その DDL がコミットされるかどうかによって決まります。

```
WITH savepoint_rollbacks (global_tran_id, local_tran_id, savepoint_id) AS
   (SELECT DISTINCT T.global_transaction_id, T.local_transaction_id, T.savepoint_id
      FROM DDLSTMTEXEC_CHGHIST as D, TXNCOMPLETION_CHGHIST as T
      WHERE T.txn_completion_status='S' AND
            D.savepoint_id >= T.savepoint_id AND
            D.event_timestamp <= T.event_timestamp)
   SELECT VARCHAR(D.STMT_TEXT, 70) AS STMT_TEXT FROM DDLSTMTEXEC_CHGHIST as D,
            TXNCOMPLETION_CHGHIST as T
      WHERE D.global_transaction_id = T.global_transaction_id AND
            D.local_transaction_id = T.local_transaction_id AND
            T.txn_completion_status = 'C' AND
            (D.global_transaction_id, D.local_transaction_id, D.savepoint_id)
   NOT IN (SELECT * FROM savepoint_rollbacks) AND
            D.EVENT_TIMESTAMP > CURRENT TIMESTAMP - 24 HOURS;

STMT_TEXT
----------------------------------------------------------------------
CREATE INDEX I1 ON T1 (ONE)

1 record(s) selected.
```

例: 変更履歴イベント・モニターを使用した STMM により実行された変更のリスト:

変更履歴イベント・モニターを使用すると、セルフチューニング・メモリー・マネージャー (STMM) により実行された変更をリストすることができます。

シナリオ

この例では、データベース管理者 (DBA) は、 STMM により実行された変更をモニターすることが必要です。 STMM により構成パラメーターおよびバッファー・プール・サイズが変更される可能性があるため、 DBA は、HIST というイベント・モニターを作成して、構成変更および DDL 変更をキャプチャーしています。

STMM により開始された変更は、このイベント・モニターで以下のいずれかの情報を含むレコードを照会することで、調べることができます。

- アプリケーション名 (appl_name) が db2stmm である。
- DDL ステートメント・テキスト (stmt_text) に、キーワード db2stmm を含むコメントが入っている。一部の DDL 変更は、STMM の代わりに他のアプリケーションが実行することに注意してください。

```
SELECT A.EVENT_TIMESTAMP,
      VARCHAR(A.EVENT_TYPE, 20) AS EVENT_TYPE, A.MEMBER
   FROM CHANGESUMMARY_HIST A LEFT OUTER JOIN
      DDLSTMTEXEC_HIST B
   ON A.EVENT_TIMESTAMP = B.EVENT_TIMESTAMP AND
      A.MEMBER = B.MEMBER AND
      A.EVENT_ID = B.EVENT_ID
   WHERE (A.APPL_NAME = 'db2stmm' OR
      B.STMT_TEXT LIKE '%db2stmm%');
```

この照会により、以下の例のような出力が戻されます。

```
EVENT_TIMESTAMP            EVENT_TYPE           MEMBER
-------------------------- -------------------- ------
2011-04-22-12.12.17.832316 DBCFG                     0
2011-04-22-12.22.35.227550 DBCFG                     0
2011-04-22-12.12.17.530274 DBCFG                     0
2011-04-22-12.12.17.721403 DBCFG                     0
2011-04-22-12.12.17.776889 DBCFG                     0
2011-04-22-12.22.35.172119 DBCFG                     0
2011-04-22-12.12.17.665098 DBCFG                     0
2011-04-22-12.22.35.116343 DBCFG                     0
2011-04-22-12.29.47.092822 DBCFG                     0
2011-04-22-12.29.47.037709 DBCFG                     0
2011-04-22-12.12.17.600511 DBCFG                     0
2011-04-22-12.22.35.283320 DBCFG                     0
2011-04-22-12.29.46.752477 DBCFG                     0
2011-04-22-12.29.47.148562 DBCFG                     0

 14 record(s) selected.
```

リリース間でのイベント・モニター・データの保持

DB2 バージョン 10.1 以降、 DB2 製品をアップグレードした後に、イベント・モニターの出力表をアップグレードできます。この機能により、アップグレードする前に存在していたイベント・モニター表のデータを保持できます。

DB2 製品でイベント・モニターが拡張されたために、イベント・モニターによって生成される表が変更される場合があります。例えば、新しいモニター・エレメントのレポート用に、新しい列が表に追加される場合があります。バージョン 10.1 より前は、表に書き込む既存のイベント・モニターがあり、それらの表のデータを保持する必要がある場合、新しく追加された列のデータ収集を行うには、新規リリースにアップグレードした後に、表を手動で変更する必要がありました。この変更で、使用する新規列を追加していました。新規列を追加しない場合、イベント・モニターは前のリリースと同様に機能して、前のリリースのイベント・モニターでサポートされていたデータのみをキャプチャーします。

変更された未フォーマット・イベント表は、一切アップグレードできませんので、ドロップして再作成する必要があります。

EVMON_UPGRADE_TABLES ストアード・プロシージャーは、既存のイベント・モニター表の定義をアップグレードして、現行レベルの DB2 製品で生成される表と一致するようにします。このフィーチャーにより、既存の表を、中に入っているすべてのデータと一緒に保持できます。表を手動で変更したり、ドロップして再作成したりする必要がなくなります。

注: また、バージョン 10.1 以降、 ALTER EVENT MONITOR ステートメントを使用して新規の論理グループをイベント・モニターに追加することもできます。この方法を EVMON_UPGRADE_TABLES の代わりに使用して、新規リリースで導入された論理データ・グループを追加することもできます。ただし、ALTER EVENT MONITOR では、既にイベント・モニターに関連付けられている論理グループを変更することはできません。既にイベント・モニターに関連付けられている論理データ・グループに変更があった場合、イベント・モニターを変更するには、EVMON_UPGRADE_TABLES プロシージャーを使用する以外に方法はありません。

EVMON_UPGRADE_TABLES プロシージャーは、通常の表および UE 表の両方に使用できます。通常の表の場合、このプロシージャーは、必要な新規列の追加、不要になった古い列のドロップ、および列の変更を必要に応じて行います。 UE 表の場合、このプロシージャーは、新規列の追加および既存の列への変更を必要に応じて行い、 **db2evmonfmt** ツールまたは EVMON_FORMAT_UE_TO_TABLES や EVMON_FORMAT_UE_TO_XML ルーチンで UE 表を処理できるようにします。

重要: アップグレード・プロセスが正常に処理されるように、すべてのアクティブなイベント・モニターを非アクティブ化する必要があります。EVMON_UPGRADE_TABLES プロシージャーは、表のアップグレードを開始する前に、すべてのアクティブなイベント・モニターを自動的に非アクティブ化します。表を EVMON_UPGRADE_TABLES によって処理しているイベント・モニターは再活動化しないでください。再活動化すると、アップグレード・プロセスは失敗します。アップグレードの前にアクティブであったイベント・モニターは、アップグレードが完了すると再び活動化されます。

イベント・モニター表をアップグレードしない場合の影響

過去のリリースと同じように、イベント・モニター表をアップグレードしないという選択も可能です。ただし、新規リリースでイベント・モニターに導入された新しい列にデータは設定されず、照会することもできません。また、古いリリースにも既に存在していたモニター・エレメントで、新規リリースでサイズが増加したモニター・エレメントは、切り捨てられる可能性があります。例えば、新規リリースで、あるモニター・エレメントのサイズが VARCHAR(20) から VARCHAR(128) に増加された場合に、既存の表をアップグレードしないと、システムがそのモニター・エレメントとして 128 バイトのデータをイベント・モニターに送信しているにもかかわらず、モニター・エレメント値が入る列は以前と同様 20 文字のデータしか格納しません。

EVMON_FORMAT_UE_TO_TABLES によって生成される表のアップグレード

UE 表を使用していた場合、EVMON_UPGRADE_TABLES プロシージャーは、 UE 表自体をアップグレードします。 EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用して通常の表を作成しているかもしれませんが、通常の表には作用しません。EVMON_UPGRADE_TABLES を使用して UE 表をアップグレードした後に、 EVMON_FORMAT_UE_TO_TABLES によって生成された出力表もアップグレードすることができます。DB2 バージョン 10.1 以降、EVMON_FORMAT_UE_TO_TABLES プロシージャーは、新規オプション UPGRADE_TABLES をサポートしています。このオプションを指定して EVMON_FORMAT_UE_TO_TABLES プロシージャーを実行すると、このプロシージャーによって生成された既存の表が変更されて、新しいバージョンの EVMON_FORMAT_UE_TO_TABLES プロシージャーで生成される出力と表の列が一致するようになります。

詳細については、 EVMON_FORMAT_UE_TO_TABLES に関する参照資料を参照してください。

第 4 章 その他のモニター・インターフェース

MONREPORT モジュールを使用して生成されるレポート

MONREPORT モジュールは、SQL のパフォーマンス上の問題をトラブルシューティングするために使用できるモニター・データのテキスト・レポートを生成します。

MONREPORT モジュールを使用して、以下のレポートを生成できます。

表 109. MONREPORT モジュールを使用して生成されるレポートのリスト

レポート名	レポートを作成するプロシージャー	主なデータ・ソース / 表関数
サマリー・レポート	MONREPORT.DBSUMMARY	MON_GET_SERVICE_SUBCLASS および MON_GET_CONNECTION と MON_GET_WORKLOAD から選択された詳細
接続レポート	MONREPORT.CONNECTION	MON_GET_CONNECTION
現行アプリケーション・レポート	MONREPORT.CURRENTAPPS	MON_GET_CONNECTION、MON_GET_UNIT_OF_WORK、WLM_GET_SERVICE_CLASS_AGENTS、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES のフィールドを含む
現行 SQL レポート	MONREPORT.CURRENTSQL	MON_GET_PKG_CACHE_STMT (WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数から取得された **executable_id** に関するもの)
パッケージ・キャッシュ・レポート	MONREPORT.PKGCACHE	MON_GET_PKG_CACHE_STMT
現行のロック待機のレポート	MONREPORT.LOCKWAIT	大部分は MON_GET_APPL_LOCKWAIT からのデータ。他は MON_GET_CONNECTION、WLM_GET_SERVICE_CLASS_AGENTS、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES、MON_GET_PKG_CACHE_STMT、 MON_GET_TABLE からのデータ。

ほとんどのレポートは、そのレポートの各項目に関するキーとなる情報を 1 行で表すサマリー・セクションで始まります。例えば、接続レポートには、各接続に関する 1 行 のサマリーがあります。レポートの本文は、そのサマリーの各項目に関する詳細なセクションで構成されます。

レポート内の各メトリックは、基礎となるモニター・エレメント名 (例えば、CLIENT_IDLE_WAIT_TIME = 44) でラベルされています。メトリックが何を表しているか調べるには、インフォメーション・センターでモニター・エレメント名を検索してください。

MONREPORT モジュールによって生成されるレポートを、カスタマイズすることができます。MONREPORT モジュールは、 SQL の使用のみで実装されているため、このモジュール・コードをデータベース・カタログから取得し、カスタマイズしたバージョンを作成することができます。

初期診断のためのレポート

これらのレポートの重要な使用目的は、 SQL のパフォーマンス低下をトラブルシューティングすることです。各レポートは、特定の診断項目に関して結果を返すように設計されています。 初期診断をサポートするレポートもあれば、特定のタイプの問題について、その後の詳細な診断をサポートするレポートもあります。

初期診断では以下が行われます。

- スローダウンする処理の状況または段階が明らかになるまで問題を絞り込むことによって、問題のカテゴリーを特定します。
- 問題に関与している SQL ステートメントを特定し、さらに分析するためにその SQL ステートメントに関する情報を収集します。

表 110. 初期診断に適した MONREPORT モジュール・レポート

プロシージャー名	提供される情報および使用方法
MONREPORT.DBSUMMARY 第 1 部: システム・パフォーマンス	サマリー・レポートの第 1 部では、データベース全体から集約された処理のさまざまな状況に関するモニター・データが提供されます。 この情報は、スローダウンする処理の状況または段階を明らかにするのに有用です。例えば、 • 問題はデータ・サーバー内にあるのか、外にあるのか。 • コンピューターのリソースにボトルネックがあるのか。 • 要求が待ち状態になっているのか。その場合、どのリソースを待っているのか。 • スローダウンは、コンポーネントを処理する特定のデータ・サーバーで発生しているのか。
MONREPORT.DBSUMMARY 第 2 部: アプリケーションのパフォーマンス	サマリー・レポートの第 2 部では、接続、ワークロード、およびサービス・クラスのそれぞれについてのキーとなるパフォーマンス標識が提供されます。 この情報は、スローダウンに関与しているアプリケーション要求の範囲を明らかにするのに有用です。例えば、 • このスローダウンは、大部分またはすべてのワークロードに影響を与えるシステム全体のスローダウンであるのか。 • このスローダウンは、特定の接続、DB2 ワークロード、または DB2 サービス・クラスなど、特定のソースから実行される SQL ステートメントに限定されているか。
MONREPORT.DBSUMMARY 第 3 部: メンバー・レベルの情報	サマリー・レポートの第 3 部では、各メンバーについてのキーとなるパフォーマンス標識が提供されます。 この情報は、スローダウンが 1 つあるいはいくつかのメンバーに限定されているのかどうかを特定するのに有用です。

表 110. 初期診断に適した MONREPORT モジュール・レポート (続き)

プロシージャー名	提供される情報および使用方法
MONREPORT.CURRENTSQL	現行 SQL レポートでは、現在実行中のステートメントの情報が、上位 *N* 個のアクティビティーを示す複数のリストという形式で提供されます。ステートメントは、次のようなさまざまなメトリックでランク付けされます: 処理リソース、処理した行数、直接読み取り数、および直接書き込み数 この情報は、スローダウンが 1 つあるいはいくつかの SQL ステートメントに限定されているのかどうかを特定するのに有用です。スローダウンが 1 つあるいはいくつかの SQL ステートメントに限定されている場合、それらのステートメントは、このレポートの上位ステートメントに表示されると考えられます。
MONREPORT.PKGCACHE	パッケージ・キャッシュ・レポートでは、最近実行されてパッケージ・キャッシュに保管されているステートメントに関する情報が提供されます。このレポートには、複数のサマリーがあり、それぞれのサマリーで上位 *N* 個のアクティビティーがリストされます。アクティビティーは、以下のようなモニター・エレメントでランク付けされます • CPU • 待機時間 • 処理された行数 • num_coord_exec_with_metrics - メトリック付きの、コーディネーター・エージェントによる実行数 : モニター・エレメント (メンバーが指定されていない場合) または num_exec_with_metrics - メトリックが収集された実行数 : モニター・エレメント (メンバーが指定されている場合) • 入出力待機時間 このレポートには、上記の各メトリックについてのサマリー、および各実行についての報告が含まれます。 この情報は、スローダウンが 1 つあるいはいくつかの SQL ステートメントに限定されているのかどうかを特定するのに有用です。その場合、それらのステートメントは、このレポートのトップに表示されると考えられます。実行単位の情報は、最もコストの高いステートメントを特定するのに役立ちます。一方、すべての実行について集計された情報は、ステートメントのコストおよび実行頻度を考慮した上で、累積的にシステムに対して最も影響を与えるステートメントを特定するのに役立ちます。

表 110. 初期診断に適した MONREPORT モジュール・レポート (続き)

プロシージャー名	提供される情報および使用方法
MONREPORT.CURRENTAPPS	現行アプリケーション・レポートでは、作業単位、エージェント、およびアクティビティーの現在の処理状態が示されます。このレポートは、現在の接続数、アクティビティー数を示すサマリー・セクションで始まり、現在の作業単位についてワークロード・オカレンス状態別に示すサマリーなどの、一連のサマリーが後に続きます。このレポートの本文は、接続ごとに、接続の詳細を示す 1 つのセクションで構成されます。 この情報は、現在システムで実行中のすべての処理を調べるのに有用です。これによって、問題のカテゴリーを特定する可能性のあるパターンを調べることができます。

詳細な診断のためのレポート

初期診断の完了後、場合によっては、初期診断の段階で特定した問題のカテゴリーについて、特化した、または詳細なトラブルシューティング分析を続けて行う必要があります。

表 111. 詳細な診断に適した MONREPORT モジュール・レポート

プロシージャー名	提供される情報および使用方法
MONREPORT.CONNECTION	MONREPORT.DBSUMMARY レポートで、特定の接続から実行される SQL ステートメントにスローダウンが限定されていることが示された場合、その接続に関する詳細な情報を確認します。 このレポートには、 MONREPORT.DBSUMMARY レポートの第 1 部と同じメトリックが含まれますが、このレポートでは、接続ごとにこの情報を示します。
MONREPORT.LOCKWAIT	初期診断で確認したレポートに、ロック待機の問題が示された場合、現在進行中の各ロック待機に関する詳細な情報を確認します。 この情報には、ロック保有者とロック要求者の詳細情報、および保有されているロックと要求されたロックの特性が含まれます。

MONREPORT モジュール・レポートのカスタマイズ

MONREPORT モジュールによって生成された既存のレポートをカスタマイズしたり、既存のレポートに基づいて新しいレポートを作成したりできます。

このタスクについて

レポートの言葉遣いや編成を変更したり、レポートに含まれるモニター・エレメントを追加、削除、または変更したりできます。また、既存のレポートに基づいて新しいレポートを作成することも可能です。

MONREPORT モジュールは、SQL ステートメント (ストアード・プロシージャー、データ・タイプを含む) を使用して実装します。MONREPORT モジュールのカスタマイズ・バージョンを作成するには、データベース・カタログからモジュール・コードを取得してから、それを変更してデプロイします。MONREPORT モジュールは SYSIBMADM スキーマで使用できます。

MONREPORT モジュールの各プロシージャーを使用して、それぞれ 1 つのレポートを生成します。例えば、CONNECTION プロシージャーを使用して接続レポートを生成します。レポートを生成するためにプロシージャー (CONNECTION プロシージャーなど) を呼び出すと、プロシージャーによって他の内部ルーチンが呼び出され、レポートの最終的な構成と書式設定が実行されます。内部ルーチンは、以下の関数を実行します。

- DB2 モニタリング表関数を呼び出して、モニター・データを取得します。多くのレポートでは、ルーチンは表関数を呼び出し、インターバルを待機してから、表関数を再び呼び出します。
- 差分値を生成して、モニター・インターバルの最初と最後のモニター値における違いを記録します。
- パーセンテージ、比率、合計、集約の値を取得するために計算を実行します。

以下の内部ルーチンをカスタマイズできます。

表 112. カスタマイズ可能なルーチン

名前	説明	使用されるレポート
CONNDELTAS	このストアード・プロシージャーは、接続メトリックの差分値の結果セットを戻します。	このルーチンは、接続レポートと一緒に使用されます。
COMMONREQMETRICS	このストアード・プロシージャーは、接続レポートとサマリー・レポートに共通のメトリックを計算し、そのメトリックのレポート出力を書式設定します。	MONREPORT.CONNECTION および MONREPORT.DBSUMMARY

MONREPORT モジュールには、このモジュールをカスタマイズするために変更する必要のないその他のオブジェクトが含まれます。

表 113. カスタマイズする必要のないルーチン

名前	説明	使用されるレポート
INITMSGCACHE	このストアード・プロシージャーは、レポートに表示される翻訳可能なストリングを取得します。	すべてのレポートで使用されます。
SAVE_EXEC_INFO	実行された SQL ステートメント・セクションについての情報を格納するために内部的に使用されるストアード・プロシージャー。	MONREPORT.PKGCACHE で使用されます。

表 113. カスタマイズする必要のないルーチン (続き)

名前	説明	使用されるレポート
db2monreport.src	このファイルはバイナリー・フォーマットで、INITMSGCACHE ルーチンがアクセスします。このファイルには、レポートに表示されるテキスト・ストリングが含まれます。	すべてのレポートで使用されます。

表 114. カスタマイズできない、ルーチンで使用されるデータ・タイプ

名前	説明	使用されるレポート
MONMETRICS_CHAR255_TYPE および MONMETRICS_CHAR32_TYPE などの名前を持つデータ・タイプ	こうした配列データ・タイプは、表関数によって戻されるモニター・エレメントを格納するのに使用されます。	ほとんどのレポートまたはすべてのレポートで使用されます。
REPORT_TYPE	ルーチンの終了時に戻されて表示されるテキスト出力を格納するために使用する配列データ・タイプ。	ほとんどのレポートまたはすべてのレポートで使用されます。
MONEXEC_TYPE	実行された SQL ステートメント・セクションを一意的に識別する実行可能な ID を格納するために使用する配列データ・タイプ。	ほとんどのレポートまたはすべてのレポートで使用されます。

手順

1. コードを取得します。
2. 必要に合わせてコードをカスタマイズします。 元のモジュールとカスタマイズされたバージョンを識別するためにモジュールの名前を変更し、適切な SQL エディターまたはストアード・プロシージャー・ビルダーを使用してコードを変更します。
   - レポートにテキストを追加するには、レポート・プロシージャーに直接追加します。INITMSGCACHE ルーチンを使用してテキスト・ストリングを管理しないでください。
   - INITMSGCACHE ルーチンを使用して取得したレポート・テキストを削除するには、キャッシュされたストリングを使用する関連コードを削除します。

   例えば、MONREPORT.CONNECTION レポートを変更するには、次のようにします。
   - メトリックを追加または削除するには、CONNDELTAS プロシージャーの SQL 照会を変更します。メトリックを追加する場合、選択する追加列を指定します。
   - 接続レポート出力の詳細セクションを変更するには、COMMONREQMETRICS ルーチンを更新して計算を追加し、新しいメトリックが出力に表示されるようにします。

- 接続レポートのサマリー・セクションを変更するには、CONNECTION ルーチンを更新して、新しいメトリックが出力に表示されるようにします。

3. MONREPORT モジュールのカスタマイズ・バージョンをデプロイします。 モジュールとそのルーチン用の一連の CREATE ステートメントまたは ALTER ステートメントを作成します。グラフィカル SQL エディターを使用している場合、エディターがデプロイメント・ステップの一部を自動的に行います。

スナップショット・モニター

スナップショット・モニターを使用して、データベースおよび特定の時刻に接続しているアプリケーションに関する情報をキャプチャーすることができます。スナップショットは、データベース・システムの状況を判別するのに役立ちます。

一定間隔でスナップショットを取れば、傾向の監視や潜在的な問題の予測にも有用です。スナップショット・モニターの一部のデータはシステム・モニターから取得しています。 システム・モニターからどのデータが取得されるかは、システム・モニター・スイッチによって決定されます。

システム・モニターは、データベースがアクティブのときにのみ、それに関する情報を集計します。データベースからすべてのアプリケーションが切断して、データベースが非アクティブ化すると、そのデータベースのシステム・モニター・データは入手不能になります。 ACTIVATE DATABASE コマンドを使用してデータベースを開始するか、データベースに対する永続接続を保守することにより、最後のスナップショットが取られるまでデータベースをアクティブにしておくこともできます。

スナップショット・モニターでは、インスタンス・アタッチメントが必要です。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。インスタンス・アタッチメントは通常、初めてデータベース・システム・モニター API を呼び出す時に、DB2INSTANCE 環境変数で指定されたインスタンスに対して暗黙的に行われます。また、ATTACH TO コマンドを使用して明示的にアタッチすることもできます。一度アプリケーションがアタッチされると、そのアプリケーションが呼び出すシステム・モニター要求は、すべてアタッチ先のインスタンスにあてられます。したがって、リモートのサーバー上のインスタンスにアタッチするだけで、そのクライアントからリモート・サーバーをモニターできるようになります。

パーティション・データベース環境では、スナップショットは、インスタンスの任意のパーティションでとることも、単一のインスタンス接続を使用してグローバルにとることもできます。グローバル・スナップショットは、それぞれのパーティションで収集されたデータを集約して単一の値セットを戻します。

DB2 pureScale 環境では、スナップショットは任意のメンバーでとることも、グローバルにとることもできます。グローバル・スナップショットは、それぞれのメンバーで収集されたデータを集約して単一の値セットを戻します。

スナップショットは CLP または SQL 表関数からキャプチャーしたり、 C または C++ で作成されたスナップショット・モニター API を使用することによってキャプチャーすることができます。さまざまなスナップショットの要求タイプが使用可能になっており、それぞれは特定のタイプのモニター・データを戻します。例えば、バッファー・プール情報だけを戻すスナップショットや、データベース・マネ

ージャー情報を戻すスナップショットをキャプチャーすることができます。スナップショットを取り込む前に、モニター・スイッチの制御下にあるモニター・エレメントからの情報が必要かどうかを考慮してください。あるモニター・スイッチが OFF の場合には、その制御下にあるモニター・エレメントは収集されません。

システム・モニター・データに対するアクセス権: SYSMON 権限

SYSMON データベース・マネージャー・レベルのグループに属するユーザーには、データベース・システム・モニター・データにアクセスできる権限があります。システム・モニター・データにアクセスするには、スナップショット・モニター API、CLP コマンド、または SQL 表関数を使用します。

SYSMON 権限グループは、システム管理権限またはシステム制御権限を持たないユーザーがデータベース・システム・モニター・データにアクセスできるようにする手段になります。

スナップショット・モニターを使用してシステム・モニター・データにアクセスする方法としては、 SYSMON 権限以外には、システム管理またはシステム制御権限を持つことしかありません。

SYSMON グループに属するユーザーや、システム管理またはシステム制御権限を持つユーザーは、以下のスナップショット・モニター関数を実行できます。

- CLP コマンド:
  - GET DATABASE MANAGER MONITOR SWITCHES
  - GET MONITOR SWITCHES
  - GET SNAPSHOT
  - LIST ACTIVE DATABASES
  - LIST APPLICATIONS
  - LIST DCS APPLICATIONS
  - LIST UTILITIES
  - RESET MONITOR
  - UPDATE MONITOR SWITCHES
- API:
  - db2GetSnapshot - スナップショットの取得
  - db2GetSnapshotSize - db2GetSnapshot() 出力バッファーに必要なサイズの見積もり
  - db2MonitorSwitches - モニター・スイッチの入手/更新
  - db2ResetMonitor - モニターのリセット
- 以前に SYSPROC.SNAP_WRITE_FILE を実行していないスナップショット SQL 表関数

スナップショット管理ビューおよび表関数を使用したデータベース・システムのスナップショットのキャプチャー

許可ユーザーは、スナップショット管理ビューまたはスナップショット表関数を使用することにより、 DB2 インスタンスに関するモニター情報のスナップショット

をキャプチャーできます。スナップショット管理ビューは、接続されたデータベースのすべてのデータベース・パーティションにおいてデータにアクセスするための簡単な方法を備えています。

スナップショット表関数は、特定のデータベース・パーティション、グローバル集合データ、またはすべてのデータベース・パーティションのデータに対して、データを要求できるようにします。スナップショット表関数の中は、すべてのアクティブ・データベースのデータを要求できるものもあります。

始める前に

データベース・スナップショットを使用するには、SYSADM、 SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。リモート・インスタンスのスナップショットを取得するには、まず、そのインスタンスに属しているローカル・データベースに接続する必要があります。

このタスクについて

新しいモニター・データが使用できるようになった場合には将来のリリースで新しいスナップショット表関数が必要になるかもしれませんが、スナップショット管理ビューのセットはそのままで、ビューに新しい列を追加するだけです。そのため管理ビューを使用すると、アプリケーションの保守を長期間に渡って行えるという利点があります。

各スナップショット・ビューは、各データベース・パーティションのモニター対象のオブジェクトごとに 1 つの行があり、各列がモニター・エレメントを表す表を戻します。各表関数は、指定されたパーティションにおいてモニター対象のオブジェクトごとに 1 つの行を持つ表を戻します。戻される表の列名は、モニター・エレメント名と相関しています。

例えば、SAMPLE データベースに関する一般アプリケーション情報のスナップショットは、SNAPAPPL 管理ビューを使用して次のようにしてキャプチャーされます。

```
SELECT * FROM SYSIBMADM.SNAPAPPL
```

戻り表から個々のモニター・エレメントを選択することもできます。例えば、次のステートメントの場合は、**agent_id** と **db_name** のモニター・エレメントだけが戻されます。

```
SELECT agent_id, db_name FROM SYSIBMADM.SNAPAPPL
```

制約事項

スナップショット管理ビューおよび表関数は以下のいずれとも併用できません。

- モニター・スイッチ・コマンドまたはモニター・スイッチ API
- モニター・リセット・コマンドまたはモニター・リセット API

この制約事項には、以下のコマンドが含まれます。

- **GET MONITOR SWITCHES**
- **UPDATE MONITOR SWITCHES**
- **RESET MONITOR**

この制約事項がある理由は、これらのコマンドが ATTACH コマンドを使用するのに対して、スナップショット表関数は CONNECT ステートメントを使用するからです。

手順

- スナップショット管理ビューを使用してスナップショットをキャプチャーするには、以下のようにします。
  1. データベースに接続します。 これは、モニターする必要のあるインスタンス内のどのデータベースでもかまいません。スナップショット管理ビューを使用した SQL 照会は、データベースに接続していなければ発行できません。
  2. キャプチャーする必要のあるスナップショットのタイプを決定します。 現在接続中のデータベース以外のデータベースのスナップショットをキャプチャーする場合、または単一のデータベース・パーティションやグローバル集合データからデータを取得する場合には、代わりにスナップショット表関数を使用する必要があります。
  3. 該当するスナップショット管理ビューを使用して照会を発行します。 例えば、次の照会では、現在接続しているデータベースのロック情報のスナップショットをキャプチャーします。

     ```
     SELECT * FROM SYSIBMADM.SNAPLOCK
     ```
- スナップショット表関数を使用してスナップショットをキャプチャーするには、以下のようにします。
  1. データベースに接続します。 これは、モニターする必要のあるインスタンス内のどのデータベースでもかまいません。スナップショット表関数を使用した SQL 照会は、データベースに接続していなければ発行できません。
  2. キャプチャーする必要のあるスナップショットのタイプを決定します。
  3. 該当するスナップショット表関数を使用して照会を発行します。 例えば、次の照会では、現在接続しているデータベース・パーティションの SAMPLE データベースに関するロック情報のスナップショットをキャプチャーします。

     ```
     SELECT * FROM TABLE(SNAP_GET_LOCK('SAMPLE',-1)) AS SNAPLOCK
     ```

     SQL 表関数には、以下の 2 つの入力パラメーターがあります。

     データベース名
     VARCHAR(255)。 NULL が入力された場合は、現在接続しているデータベースの名前が使用されます。

     パーティション番号
     SMALLINT。データベース・パーティション番号のパラメーターには、モニターする必要のあるデータベース・パーティションの番号に対応する整数 (0 から 999 の間の値) を入力します。現在接続しているデータベース・パーティションのスナップショットをキャプチャーする場合は、値 -1 を入力します。グローバル集合スナップショットをキャプチャーする場合は、値 -2 を入力します。すべてのデータベース・パーティションでスナップショットをキャプチャーする場合は、このパラメーターに値を指定しないでください。

     注:

a. ただし、次に挙げるスナップショット表関数の場合は、現在接続しているデータベースを指定するために NULL を入力すると、インスタンス内のすべてのデータベースに関するスナップショット情報が戻されます。
   - SNAP_GET_DB
   - SNAP_GET_DB_MEMORY_POOL
   - SNAP_GET_DETAILLOG
   - SNAP_GET_HADR
   - SNAP_GET_STORAGE_PATHS
   - SNAP_GET_APPL
   - SNAP_GET_APPL_INFO
   - SNAP_GET_AGENT
   - SNAP_GET_AGENT_MEMORY_POOL
   - SNAP_GET_STMT
   - SNAP_GET_SUBSECTION
   - SNAP_GET_BP
   - SNAP_GET_BP_PART

b. データベース名パラメーターは、データベース・マネージャー・レベルのスナップショット表関数には適用されません。あるのは、データベース・パーティション番号用のパラメーターだけです。データベース・パーティション番号パラメーターはオプションです。

SNAP_WRITE_FILE ストアード・プロシージャーを使用した、データベース・システム・スナップショット情報のファイルへの取り込み

SNAP_WRITE_FILE ストアード・プロシージャーを使用すると、モニター・データのスナップショットを取り込み、この情報をデータベース・サーバー上のファイルに保管することができます。また、SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限を持たないユーザーがデータにアクセスすることを許可できます。

これにより、すべてのユーザーがスナップショット表関数を使用した照会を行い、これらのファイルにあるスナップショット情報にアクセスできるようになります。そのため、スナップショット・モニター・データへのアクセスをオープンにすると、スナップショット表関数の実行権限を持つすべてのユーザーが、接続したユーザーのリストや、それらのユーザーがデータベースにサブミットした SQL ステートメントなどの機密情報にアクセス可能になってしまいます。スナップショット表関数の実行権限は、デフォルトで、PUBLIC に付与されています。 (ただし、表の実データやユーザー・パスワードがスナップショット・モニター表関数によって漏えいすることはありません。)

始める前に

SNAP_WRITE_FILE ストアード・プロシージャーを使用してデータベース・スナップショットを取り込むには、 SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。

このタスクについて

SNAP_WRITE_FILE ストアード・プロシージャーへの呼び出しを発行するときは、モニターするデータベースとパーティションを識別することに加えて、スナップショット要求タイプ を指定する必要があります。各スナップショット要求タイプは、収集するモニター・データの有効範囲を決定します。各ユーザーが実行する必要のあるスナップショット表関数に基づいて、スナップショット要求タイプを選択してください。次の表は、スナップショット表関数とそれに対応する要求タイプのリストです。

表 115. スナップショット要求タイプ

スナップショット表関数	スナップショット要求タイプ
SNAP_GET_AGENT	APPL_ALL
SNAP_GET_AGENT_MEMORY_POOL	APPL_ALL
SNAP_GET_APPL	APPL_ALL
SNAP_GET_APPL_INFO	APPL_ALL
SNAP_GET_STMT	APPL_ALL
SNAP_GET_SUBSECTION	APPL_ALL
SNAP_GET_BP_PART	BUFFERPOOLS_ALL
SNAP_GET_BP	BUFFERPOOLS_ALL
SNAP_GET_DB	DBASE_ALL
SNAP_GET_DETAILLOG	DBASE_ALL
SNAP_GET_DB_MEMORY_POOL	DBASE_ALL
SNAP_GET_HADR	DBASE_ALL
SNAP_GET_STORAGE_PATHS	DBASE_ALL
SNAP_GET_DBM	DB2
SNAP_GET_DBM_MEMORY_POOL	DB2
SNAP_GET_FCM	DB2
SNAP_GET_FCM_PART	DB2
SNAP_GET_SWITCHES	DB2
SNAP_GET_DYN_SQL	DYNAMIC_SQL
SNAP_GET_LOCK	DBASE_LOCKS
SNAP_GET_LOCKWAIT	APPL_ALL
SNAP_GET_TAB	DBASE_TABLES
SNAP_GET_TAB_REORG	DBASE_TABLES
SNAP_GET_TBSP	DBASE_TABLESPACES
SNAP_GET_TBSP_PART	DBASE_TABLESPACES
SNAP_GET_CONTAINER	DBASE_TABLESPACES
SNAP_GET_TBSP_QUIESCER	DBASE_TABLESPACES
SNAP_GET_TBSP_RANGE	DBASE_TABLESPACES
SNAP_GET_UTIL	DB2
SNAP_GET_UTIL_PROGRESS	DB2

手順

1. データベースに接続します。これは、モニターする必要のあるインスタンス内のどのデータベースでもかまいません。ストアード・プロシージャーは、データベースに接続していなければ呼び出せません。
2. スナップショット要求タイプ、およびモニターする必要のあるデータベースとパーティションを決定します。
3. スナップショット要求タイプ、データベース、およびパーティションを指定する適切なパラメーターを設定して、 SNAP_WRITE_FILE ストアード・プロシージャーを呼び出します。例えば、次の呼び出しでは、現在接続しているパーティションの SAMPLE データベースに関するアプリケーション情報のスナップショットを取り込みます。

   ```
   CALL SNAP_WRITE_FILE('APPL_ALL','SAMPLE',-1)
   ```

   SNAP_WRITE_FILE ストアード・プロシージャーには、3 つの入力パラメーターがあります。

   - スナップショット要求タイプ (526 ページの表 115を参照してください。この表は、スナップショット表関数とそれに対応する要求タイプを相互参照させたものです。)
   - VARCHAR (128): データベース名。 NULL が入力された場合は、現在接続しているデータベースの名前が使用されます。

     注: このパラメーターは、データベース・マネージャー・レベルのスナップショット表関数には適用されません。あるのは、要求タイプとパーティション番号のパラメーターだけです。
   - SMALLINT: パーティション番号 (0 から 999 の間の値)。パーティション番号パラメーターの場合は、モニターするパーティション番号に対応する整数を入力します。現在接続しているパーティションのスナップショットを取り込む場合は、値 -1 または NULL を入力します。グローバル・スナップショットをキャプチャーするには、値 -2 を入力します。

タスクの結果

スナップショット・データをファイルに保管した後、すべてのユーザーは、対応するスナップショット表関数を使用して照会を発行することができます。その際、データベース・レベルの表関数の入力値として (NULL, NULL) を指定し、データベース・マネージャー・レベルの表関数には (NULL) を指定します。受け取るモニター・データは、SNAP_WRITE_FILE ストアード・プロシージャーによって生成されたファイルからプルされます。

注: これにより、ユーザーによる機密モニター・データへのアクセスを制限することができますが、このアプローチにはいくつかの制限があります。

- SNAP_WRITE_FILE ファイルから入手できるスナップショット・モニター・データは、最後に SNAP_WRITE_FILE ストアード・プロシージャーが呼び出されたときと同じくらい新しいものになる。通常のインターバルで SNAP_WRITE_FILE ストアード・プロシージャーへの呼び出しを行うことによって、最新のスナップショット・モニター・データが使用可能であることを確認することができます。例えば、UNIX システム上では、これを行うために cron ジョブを設定することができます。

- スナップショット表関数を使用して照会を発行しているユーザーは、モニターするデータベースまたはパーティションを識別することができない。SNAP_WRITE_FILE 呼び出しを発行しているユーザーによって識別されるデータベース名およびパーティション番号が、スナップショット表関数によってアクセス可能なファイルの内容を決定します。
- ユーザーが、対応する SNAP_WRITE_FILE 要求タイプが実行されていないスナップショット表関数を含む SQL 照会を発行すると、現在接続されているデータベースおよびパーティションに対して直接スナップショットが試行されます。この操作は、ユーザーが SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限を持っている場合にのみ成功します。

SQL 照会のスナップショット表関数を使用したデータベース・システムのスナップショットへのアクセス (ファイル・アクセス使用)

許可ユーザーが SNAP_WRITE_FILE ストアード・プロシージャーを呼び出したすべての要求タイプについては、どのユーザーも、相当するスナップショット表関数を使用した照会を発行できます。ユーザーが受け取るモニター・データは、SNAP_WRITE_FILE ストアード・プロシージャーによって生成されたファイルから取り出されます。

始める前に

SNAP_WRITE_FILE ファイルにアクセスすることを目的として使用されるすべてのスナップショット表関数については、許可ユーザーが、該当するスナップショット要求タイプを設定して SNAP_WRITE_FILE ストアード・プロシージャーを発行している必要があります。発行された SQL 照会に、該当する SNAP_WRITE_FILE 要求タイプが実行されていないスナップショット表関数が含まれている場合は、現在接続されているデータベースおよびパーティションへの直接のスナップショットが試行されます。この操作は、ユーザーが SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限を持っている場合にのみ成功します。

このタスクについて

スナップショット表関数を使用して SNAP_WRITE_FILE ファイルのスナップショット・データにアクセスするユーザーには、モニターするデータベースやパーティションは識別できません。 SNAP_WRITE_FILE ファイルの内容は、ユーザーが SNAP_WRITE_FILE 呼び出しを実行して識別するデータベース名とパーティション番号によって決定されます。 SNAP_WRITE_FILE ファイルから得られるスナップショット・モニター・データは、直前に呼び出された SNAP_WRITE_FILE ストアード・プロシージャーでキャプチャーされたスナップショットだけです。

手順

1. データベースに接続します。これは、モニターする必要のあるインスタンス内のどのデータベースでもかまいません。スナップショット表関数を使用した SQL 照会は、データベースに接続していなければ発行できません。
2. キャプチャーする必要のあるスナップショットのタイプを決定します。
3. 該当するスナップショット表関数を使用して照会を発行します。例えば、次の照会では、表スペース情報のスナップショットがキャプチャーされます。

```
SELECT * FROM TABLE(SNAP_GET_TBSP(CAST(NULL AS VARCHAR(1)),
                    CAST (NULL AS INTEGER))) AS SNAP_GET_TBSP
```

注: データベース名やパーティション番号のパラメーターには、NULL 値を入力してください。スナップショットの対象となるデータベース名やパーティションは、 SNAP_WRITE_FILE ストアード・プロシージャーの呼び出しで決定されます。また、データベース名のパラメーターは、データベース・マネージャー・レベルのスナップショット表関数には適用されません。あるのは、パーティション番号のパラメーターだけです。
各スナップショット表関数は、1 つ以上の行があり、各列がモニター・エレメントを表す表を戻します。したがって、各モニター・エレメントの列名は、モニター・エレメントの名前に対応しています。

4. 戻り表から個々のモニター・エレメントを選択することもできます。例えば、次のステートメントの場合は、**agent_id** のモニター・エレメントだけが戻されます。

```
SELECT agent_id FROM TABLE(
                          SNAP_GET_APPL(CAST(NULL AS VARCHAR(1)),
                                        CAST(NULL AS INTEGER)))
                          as SNAP_GET_APPL
```

スナップショット・モニター SQL 管理ビュー

利用可能なスナップショット・モニター SQL 管理ビューはたくさんの種類があり、それぞれがデータベース・システムの特定の領域に関するモニター・データを戻します。

例えば、SYSIBMADM.SNAPBP SQL 管理ビューはバッファー・プール情報のスナップショットを取り込みます。次の表は、利用可能な各スナップショット・モニター管理ビューをリストしています。

表 116. スナップショット・モニター SQL 管理ビュー

モニター・レベル	SQL 管理ビュー	戻される情報
データベース・マネージャー	SYSIBMADM.SNAPDBM	データベース・マネージャー・レベル情報。
データベース・マネージャー	SYSIBMADM.SNAPFCM	高速コミュニケーション・マネージャー (FCM) に関するデータベース・マネージャー・レベル情報。
データベース・マネージャー	SYSIBMADM.SNAPFCM_PART	高速コミュニケーション・マネージャー (FCM) に関する、あるパーティションのデータベース・マネージャー・レベル情報。
データベース・マネージャー	SYSIBMADM.SNAPSWITCHES	データベース・マネージャーのモニター・スイッチの設定値。
データベース・マネージャー	SYSIBMADM.SNAPDBM_MEMORY_POOL	メモリー使用量についてのデータベース・マネージャー・レベル情報。
データベース	SYSIBMADM.SNAPDB	データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。

表 116. スナップショット・モニター SQL 管理ビュー (続き)

モニター・レベル	SQL 管理ビュー	戻される情報
データベース	SYSIBMADM.SNAPDB_MEMORY_POOL	メモリー使用量についてのデータベース・レベル情報 (UNIX プラットフォームのみ)。
アプリケーション	SYSIBMADM.SNAPAPPL	データベースに接続されている各アプリケーションについての汎用アプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SYSIBMADM.SNAPAPPL_INFO	データベースに接続されている各アプリケーションについての汎用アプリケーション・レベル識別情報。
アプリケーション	SYSIBMADM.SNAPLOCKWAIT	データベースに接続されているアプリケーションのロック待機についてのアプリケーション・レベル情報。
アプリケーション	SYSIBMADM.SNAPSTMT	データベースに接続されているアプリケーションのステートメントについてのアプリケーション・レベル情報。これには、最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SYSIBMADM.SNAPAGENT	データベースに接続されているアプリケーションに関連したエージェントについてのアプリケーション・レベル情報。
アプリケーション	SYSIBMADM.SNAPSUBSECTION	データベースに接続されているアプリケーションのアクセス・プランのサブセクションについてのアプリケーション・レベル情報。
アプリケーション	SYSIBMADM.SNAPAGENT_MEMORY_POOL	エージェント・レベルでのメモリー使用量に関する情報。
表	SYSIBMADM.SNAPTAB	データベースに接続された各アプリケーションのデータベース・レベルとアプリケーション・レベルの表アクティビティー情報。データベースに接続されたアプリケーションがアクセスした各表の表レベルの表アクティビティー情報。表スイッチが必要です。
表	SYSIBMADM.SNAPTAB_REORG	データベース内で再編成を実行している各表に関する、表レベルでの表再編成情報。
ロック	SYSIBMADM.SNAPLOCK	データベースに接続された各アプリケーションについての、データベース・レベルおよびアプリケーション・レベルのロック情報。ロック・スイッチが必要です。
表スペース	SYSIBMADM.SNAPTBSP	データベース・レベルの表スペース・アクティビティーに関する情報。また、データベースに接続された各アプリケーションのアプリケーション・レベル、およびデータベースに接続された各アプリケーションがアクセスしている各表スペースの表スペース・レベルの情報も含みます。バッファー・プール・スイッチが必要です。

表 116. スナップショット・モニター SQL 管理ビュー (続き)

モニター・レベル	SQL 管理ビュー	戻される情報
表スペース	SYSIBMADM.SNAPTBSP_PART	表スペース構成に関する情報。
表スペース	SYSIBMADM.SNAPTBSP_QUIESCER	表スペース・レベルでの静止プログラムに関する情報。
表スペース	SYSIBMADM.SNAPCONTAINER	表スペース・レベルでの表スペース・コンテナー構成に関する情報。
表スペース	SYSIBMADM.SNAPTBSP_RANGE	表スペース・マップの範囲に関する情報。
バッファー・プール	SYSIBMADM.SNAPBP	指定されたデータベースのバッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
バッファー・プール	SYSIBMADM.SNAPBP_PART	パーティションごとに計算したバッファー・サイズおよび使用量についての情報。
動的 SQL	SYSIBMADM.SNAPDYN_SQL	データベースの SQL ステートメント・キャッシュからのポイント・イン・タイム指定ステートメント情報。
データベース	SYSIBMADM.SNAPUTIL	ユーティリティーに関する情報。
データベース	SYSIBMADM.SNAPUTIL_PROGRESS	ユーティリティーの進行に関する情報。
データベース	SYSIBMADM.SNAPDETAILLOG	ログ・ファイルについてのデータベース・レベル情報。
データベース	SYSPROC.ADMIN_GET_STORAGE_PATHS	データベースの自動ストレージ・パスのリスト、および各ストレージ・パスのファイル・システム情報を戻します。

スナップショットを取り込む前に、モニター・スイッチの制御下にあるモニター・エレメントからの情報が必要かどうかを考慮してください。あるモニター・スイッチが OFF の場合には、その制御下にあるモニター・エレメントは収集されません。必要とするエレメントをスイッチで制御できるかどうか判別するには、個々のモニター・エレメントを参照してください。

すべてのスナップショット・モニター管理ビューおよび関連した表関数は、現行セッションで使用されている接続とは異なる、独立したインスタンス接続を使用します。したがって、暗黙のインスタンス接続を確立でき、デフォルトのデータベース・マネージャー・モニター・スイッチのみ有効です。無効なモニター・スイッチには、現行セッションまたはアプリケーションから動的に ON/OFF されるスイッチが含まれます。

また、個々のモニター・エレメントの値を戻すだけでなく、モニター・タスクで一般的に必要とされる計算値も戻す管理ビューのセットも用意されています。例えば、SYSIBMADM.BP_HITRATIO 管理ビューはバッファー・プールのヒット率に関する計算値を戻しますが、これは複数の別個のモニター・エレメントを組み合わせたものです。

表 117. スナップショット・モニター SQL 管理用のビュー

SQL 管理用のビュー	戻される情報
SYSIBMADM.APPLICATIONS	接続されたデータベース・アプリケーションに関する情報。

表 117. スナップショット・モニター SQL 管理用のビュー (続き)

SQL 管理用のビュー	戻される情報
SYSIBMADM.APPL_PERFORMANCE	選択された行とアプリケーションによって読み取られた行数の比率に関する情報。
SYSIBMADM.BP_HITRATIO	データベース内のバッファー・プールのヒット率 (合計、データ、および索引を含む)。
SYSIBMADM.BP_READ_IO	バッファー・プールの読み取りパフォーマンスに関する情報。
SYSIBMADM.BP_WRITE_IO	バッファー・プールの書き込みパフォーマンスに関する情報。
SYSIBMADM.CONTAINER_UTILIZATION	表スペース・コンテナーと使用率に関する情報
SYSIBMADM.LOCKS_HELD	現在保持されているロックに関する情報。
SYSIBMADM.LOCKWAITS	ロック取得のために待機しているアプリケーションのために作動している DB2 エージェントについての情報。
SYSIBMADM.LOG_UTILIZATION	現在接続中のデータベースのログ使用率に関する情報。
SYSIBMADM.LONG_RUNNING_SQL	現在接続しているデータベース内で最も長時間実行されている SQL に関する情報。
SYSIBMADM.QUERY_PREP_COST	様々な SQL ステートメントの準備に必要な時間に関する情報。
SYSIBMADM.TBSP_UTILIZATION	表スペースの構成および使用率の情報。
SYSIBMADM.TOP_DYNAMIC_SQL	実行数、平均実行時間、ソート数、またはステートメントごとのソート数を尺度にした場合に上位を占める動的 SQL ステートメント群。これらの項目に従ってソート可能。

データベース・システム・スナップショットへの SQL アクセス

スナップショット・モニター SQL 表関数 (*スナップショット表関数* と呼ばれる) を使用して、スナップショット・モニター・データにアクセスするには、直接アクセスおよびファイル・アクセスの 2 とおりの方法があります。

このタスクについて

直接アクセス

許可ユーザーは、スナップショット表関数で照会を発行し、モニター・データを含む結果セットを受けとることができます。この方法では、スナップショット・モニター・データへのアクセスは、 SYSADM、 SYSCTRL、 SYSMAINT、または SYSMON 権限を持つユーザーにのみ可能です。

直接アクセスを使用してスナップショット情報をキャプチャーするには、以下のようにします。

1. オプション: モニター・スイッチの状況を設定および確認します。詳しくは、 557 ページの『CLP からのシステム・モニター・スイッチの設定』を参照してください。
2. SQL を使用してデータベース・システムのスナップショットをキャプチャーします。詳しくは、 522 ページの『スナップショット管理ビューおよび表関数を使用したデータベース・システムのスナップショットのキャプチャー』を参照してください。

ファイル・アクセス

許可ユーザーは、スナップショット要求タイプ、および影響を受けるパーティションやデータベースを識別して、 SNAPSHOT_FILEW ストアード・プロシージャーを呼び出します。次に、SNAPSHOT_FILEW ストアード・プロシージャーは、データベース・サーバー上のファイルにモニター・データを保管します。

許可ユーザーが SNAPSHOT_FILEW ストアード・プロシージャーを呼び出せる各要求タイプ

これは、すべてのユーザーにスナップショット・モニター・データへのアクセスを提供する安全な方法ですが、この方法には以下の制限があります。

- SNAPSHOT_FILEW ファイルから入手できるスナップショット・モニター・データは、最後に SNAPSHOT_FILEW ストアード・プロシージャーが呼び出されたときと同じくらい新しいものになる。通常のインターバルで SNAPSHOT_FILEW ストアード・プロシージャーへの呼び出しを行うことによって、最新のスナップショット・モニター・データが使用可能であることを確認することができます。例えば、UNIX オペレーティング・システム上では、これを行うために cron ジョブを設定することができます。
- スナップショット表関数を使用して照会を発行しているユーザーは、モニターするデータベースまたはパーティションを識別することができない。SNAPSHOT_FILEW 呼び出しを発行しているユーザーによって識別されるデータベース名およびパーティション番号が、スナップショット表関数によってアクセス可能なファイルの内容を決定します。
- ユーザーが、対応する SNAPSHOT_FILEW 要求タイプが実行されていないスナップショット表関数を含む SQL 照会を発行すると、現在接続されているデータベースおよびパーティションに対して直接スナップショットが試行されます。この操作は、ユーザーが
  SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限を持っている場合にのみ成功します。

以下のタスクは、データベース・システム・スナップショット情報をファイルにキャプチャーする SYSADM、SYSCTRL、SYSMAINT、または SYSMON ユーザーによって実行されます。

手順

1. スナップショット要求を発行するユーザーの必要を調べます。 特に、必要なモニター・データ、収集元のデータベース、および収集が特定のパーティションに限定される必要があるかどうかを判別します。
2. オプション: モニター・スイッチの状況を設定および確認します。 詳しくは、557 ページの『CLP からのシステム・モニター・スイッチの設定』を参照してください。
3. ファイルへのデータベース・システム・スナップショット情報のキャプチャーを行います。 詳しくは、525 ページの『SNAP_WRITE_FILE ストアード・プロシージャーを使用した、データベース・システム・スナップショット情報のファイルへの取り込み』を参照してください。

次のタスク

いったん、SYSADM、SYSCTRL、SYSMAINT、または SYSMON ユーザーが上記のステップを完了したら、すべてのユーザーは、SQL 照会内のスナップショット表関数を使用してデータベース・システム・スナップショット情報にアクセスすることができます。

CLP からのデータベース・スナップショットのキャプチャー

CLP から GET SNAPSHOT コマンドを使用して、データベース・スナップショットをキャプチャーすることができます。多数の異なるスナップショット要求タイプを使用することができます。それらには、GET SNAPSHOT コマンドに特定のパラメーターを指定することによってアクセスできます。

始める前に

データベース・スナップショットを使用するには、SYSADM、 SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。

データベース・スナップショットをキャプチャーするには、インスタンス・アタッチメントがなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのスナップショットを取得するには、まずそのインスタンスにアタッチする必要があります。

手順

- オプション: モニター・スイッチの状況を設定および確認します。
- CLP から、GET SNAPSHOT コマンドを必要なパラメーターを指定して発行します。次の例では、スナップショットはすべてのデータベースに関する情報をキャプチャーします。

  ```
  db2 get snapshot for all databases
  ```

  特定のデータベースについてのデータベース・スナップショットをキャプチャーするには、次のコマンドを使用します。

  ```
  db2 get snapsot for database on db-name
  ```

  db-name は、対象となるデータベースの名前です。

- 次の例では、データベース・マネージャー・レベルの情報をキャプチャーします。

  ```
  db2 get snapshot for dbm
  ```

- パーティション・データベース・システムの場合、特定のパーティションについてデータベース・スナップショットを固有にキャプチャーすることも、すべてのパーティションについてグローバルなスナップショットをキャプチャーすることもできます。特定のパーティション (例えば、パーティション番号 2) 上のすべてのアプリケーションについてのデータベース・スナップショットをキャプチャーするには、次のコマンドを発行します。

  ```
  db2 get snapshot for all applications at dbpartitionnum 2
  ```

- すべてのパーティション上のすべてのアプリケーションについてのデータベース・スナップショットをキャプチャーするには、次のコマンドを発行します。

  ```
  db2 get snapshot for all applications global
  ```

  パーティション・データベース上のグローバル・スナップショットの場合、すべてのパーティションからのモニター・データが集約されます。

スナップショット・モニター CLP コマンド

コマンド行プロンプトから、スナップショット・モニターを制御できます。いくつかの要求タイプの場合、一部の情報は、関連したモニター・スイッチが ON に設定されている場合にだけ戻されます。スイッチで必須エレメントを制御できるかどうか判別するには、個々のモニター・エレメントを参照してください。

次の表に、サポートされているスナップショット要求のタイプをすべてリストします。

表 118. スナップショット・モニター CLP コマンド

モニター・レベル	CLP コマンド	戻される情報
接続リスト	`list applications [show detail]`	スナップショットが取られたパーティション上の DB2 インスタンスが管理するデータベースに現在接続されている、すべてのアプリケーションのアプリケーション識別情報。
接続リスト	`list applications for database` *dbname* `[show detail]`	指定のデータベースに現在接続されている各アプリケーションに関するアプリケーション識別情報。
接続リスト	`list dcs applications`	スナップショットが取られたパーティション上の DB2 インスタンスが管理するデータベースに現在接続されている、すべての DCS アプリケーションのアプリケーション識別情報。
データベース・マネージャー	`get snapshot for dbm`	インスタンス・レベルのモニター・スイッチの設定値を含む、データベース・マネージャー・レベルの情報。
データベース・マネージャー	`get dbm monitor switches`	インスタンス・レベルのモニター・スイッチの設定値。
データベース	`get snapshot for database on` *dbname*	データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。

表 118. スナップショット・モニター CLP コマンド (続き)

モニター・レベル	CLP コマンド	戻される情報
データベース	`get snapshot for all databases`	パーティション上でアクティブな各データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	`list active databases`	個々のアクティブなデータベースに対する接続数。ACTIVATE DATABASE コマンドを使用して開始したものの、接続されていないデータベースを含みます。
データベース	`get snapshot for dcs database on dbname`	特定の DCS データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	`get snapshot for remote database on dbname`	特定のフェデレーテッド・システム・データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	`get snapshot for all remote databases`	パーティション上でアクティブな各フェデレーテッド・システム・データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
アプリケーション	`get snapshot for application applid appl-id`	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for application agentid appl-handle`	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for applications on dbname`	パーティション上のデータベースに接続されている各アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for all applications`	パーティション上でアクティブな各アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for dcs application applid appl-id`	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。

表 118. スナップショット・モニター *CLP* コマンド *(続き)*

モニター・レベル	CLP コマンド	戻される情報
アプリケーション	`get snapshot for all dcs applications`	パーティション上にあるアクティブな各 DCS アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for dcs application agentid` *`appl-handle`*	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for dcs applications on` *`dbname`*	パーティション上にあるデータベースに接続されている各 DCS アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for remote applications on` *`dbname`*	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for all remote applications`	パーティション上にあるアクティブな各フェデレーテッド・システム・アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
表	`get snapshot for tables on` *`dbname`*	データベースに接続された各アプリケーションのデータベース・レベルとアプリケーション・レベルの表アクティビティー情報。データベースに接続されたアプリケーションが**アクセスした**各表の表レベルの表アクティビティー情報。表スイッチが必要です。
ロック	`get snapshot for locks for application applid` *`appl-id`*	アプリケーションによって保持されているロックのリスト。ロック待機情報にはロック・スイッチが必要です。
ロック	`get snapshot for locks for application agentid` *`appl-handle`*	アプリケーションによって保持されているロックのリスト。ロック待機情報にはロック・スイッチが必要です。
ロック	`get snapshot for locks on` *`dbname`*	データベースに接続された各アプリケーションについての、データベース・レベルおよびアプリケーション・レベルのロック情報。ロック・スイッチが必要です。
表スペース	`get snapshot for tablespaces on` *`dbname`*	データベースの表スペースのアクティビティーに関する情報。バッファー・プール・スイッチが必要です。コンテナー、静止プログラム、および範囲に関する情報も含まれます。この情報はスイッチの制御下にはありません。

表 118. スナップショット・モニター CLP コマンド (続き)

モニター・レベル	CLP コマンド	戻される情報
バッファー・プール	`get snapshot for all bufferpools`	バッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
バッファー・プール	`get snapshot for bufferpools on dbname`	指定されたデータベースのバッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
動的 SQL	`get snapshot for dynamic sql on dbname`	データベースの SQL ステートメント・キャッシュからのポイント・イン・タイム指定ステートメント情報。リモート・データ・ソースからの情報である場合もあります。

クライアント・アプリケーションからのデータベース・スナップショットのキャプチャー

C、C++、または COBOL アプリケーションでスナップショット・モニター API を使用して、データベース・スナップショットをキャプチャーすることができます。C および C++ では、db2GetSnapshot() に特定のパラメーターを指定することにより、多数の異なるスナップショット要求タイプにアクセスすることができます。

始める前に

db2MonitorSwitches API を使用するには、SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。

データベース・スナップショットをキャプチャーするには、インスタンス・アタッチメントがなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのスナップショットを取得するには、まずそのインスタンスにアタッチする必要があります。

手順

1. オプション: モニター・スイッチの状況を設定および確認します。
2. DB2 ライブラリー、sqlmon.h および db2ApiDf.h を組み込みます。これらは `sqllib` の下の `include` サブディレクトリーにあります。

   ```
   #include <db2ApiDf.h>
   #include <sqlmon.h>
   ```

3. スナップショットのバッファー単位サイズを 100 KB に設定します。

   ```
   #define SNAPSHOT_BUFFER_UNIT_SZ 102400
   ```

4. sqlca、sqlma、db2GetSnapshotData、および sqlm_collected 構造体を宣言します。また、スナップショット・バッファーを含むようにポインターを初期化し、バッファーのサイズを設定します。

   ```
   struct sqlma *pRequestedDataGroups;
   struct sqlca sqlca;
   memset (&sqlca, '¥0', sizeof(struct sqlca));
   struct sqlm_collected collectedData;
   memset (&collectedData, '¥0', sizeof(collectedData));
   db2GetSnapshotData getSnapshotParam;
   memset (&getSnapshotParam, '¥0', sizeof(getSnapshotParam));
   ```

```
static sqluint32 snapshotBufferSize = SNAPSHOT_BUFFER_UNIT_SZ;
sqluint32 outputFormat;
char *snapshotBuffer;
```

5. sqlma 構造体を初期化し、キャプチャーするスナップショットがデータベース・マネージャー・レベル情報のものであることを指定します。

```
pRequestedDataGroups = (struct sqlma *)malloc(SQLMASIZE(1));
memset(pRequestedDataGroups, '¥0', SQLMASIZE(1));
pRequestedDataGroups->obj_num = 1;
pRequestedDataGroups->obj_var[0].obj_type = SQLMA_DB2;
```

6. スナップショット出力を保持するバッファーを初期化します。

```
snapshotBuffer = (char *)malloc(snapshotBufferSize);
memset (snapshotBuffer, '¥0', snapshotBufferSize);
```

7. db2GetSnapshotData 構造体に、スナップショット要求タイプ (sqlma 構造体から)、バッファー情報、およびスナップショットをキャプチャーするために必要な他の情報を含めます。

```
getSnapshotParam.piSqlmaData = pRequestedDataGroups;
getSnapshotParam.poCollectedData = &collectedData;
getSnapshotParam.poBuffer = snapshotBuffer;
getSnapshotParam.iVersion = SQLM_DBMON_VERSION9_5;
getSnapshotParam.iBufferSize = snapshotBufferSize;
getSnapshotParam.iStoreResult = 0;
getSnapshotParam.iNodeNumber = SQLM_CURRENT_NODE;
getSnapshotParam.poOutputFormat = &outputFormat;
getSnapshotParam.iSnapshotClass = SQLM_CLASS_DEFAULT;
```

8. スナップショットをキャプチャーします。 db2GetSnapshotData 構造体を渡します。これには、スナップショットをキャプチャーするのに必要な情報に加えて、スナップショット出力の宛先となるバッファーの参照も含まれています。

```
db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
```

9. バッファーのオーバーフローを処理するためのロジックを組み込みます。スナップショットが取られた後、バッファー・オーバーフローについて sqlcode がチェックされます。バッファー・オーバーフローが発生した場合には、バッファーがクリアされて再初期化され、スナップショットが再度取られます。

```
while (sqlca.sqlcode == 1606)
{
  free(snapshotBuffer);
  snapshotBufferSize = snapshotBufferSize +
  SNAPSHOT_BUFFER_UNIT_SZ;
  snapshotBuffer = (char *)malloc(snapshotBufferSize);
  if (snapshotBuffer == NULL)
  {
    printf("¥nMemory allocation error.¥n");
    return 1;
  }
  getSnapshotParam.iBufferSize = snapshotBufferSize;
  getSnapshotParam.poBuffer = snapshotBuffer;
  db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
}
```

10. スナップショット・モニターのデータ・ストリームを処理します。
11. バッファーをクリアします。

```
free(snapshotBuffer);
free(pRequestedDataGroups);
```

スナップショット・モニター API 要求タイプ

いくつかの要求タイプの場合、一部の情報は、関連したモニター・スイッチが ON に設定されている場合にだけ戻されます。スイッチで必須エレメントを制御できるかどうか判別するには、個々のモニター・エレメントを参照してください。

次の表に、サポートされているスナップショット要求のタイプをすべてリストします。

表 119. スナップショット・モニター API 要求タイプ

モニター・レベル	API 要求タイプ	戻される情報
接続リスト	SQLMA_APPLINFO_ALL	スナップショットが取られたパーティション上の DB2 インスタンスが管理するデータベースに現在接続されている、すべてのアプリケーションのアプリケーション識別情報。
接続リスト	SQLMA_DBASE_APPLINFO	指定のデータベースに現在接続されている各アプリケーションに関するアプリケーション識別情報。
接続リスト	SQLMA_DCS_APPLINFO_ALL	スナップショットが取られたパーティション上の DB2 インスタンスが管理するデータベースに現在接続されている、すべての DCS アプリケーションのアプリケーション識別情報。
データベース・マネージャー	SQLMA_DB2	インスタンス・レベルのモニター・スイッチの設定値を含む、データベース・マネージャー・レベルの情報。
データベース	SQLMA_DBASE	データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DBASE_ALL	パーティション上でアクティブな各データベースのデータベース・レベル情報およびカウンター。個々のアクティブなデータベースに対する接続数。 ACTIVATE DATABASE コマンドを使用して開始したものの、接続されていないデータベースを含みます。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DCS_DBASE	特定の DCS データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DCS_DBASE_ALL	パーティション上でアクティブな各 DCS データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。

表 119. スナップショット・モニター API 要求タイプ (続き)

モニター・レベル	API 要求タイプ	戻される情報
データベース	SQLMA_DBASE_REMOTE	特定のフェデレーテッド・システム・データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DBASE_REMOTE_ALL	パーティション上でアクティブな各フェデレーテッド・システム・データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
アプリケーション	SQLMA_APPL	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_AGENT_ID	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DBASE_APPLS	パーティション上のデータベースに接続されている各アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_APPL_ALL	パーティション上でアクティブな各アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DCS_APPL	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DCS_APPL_ALL	パーティション上にあるアクティブな各 DCS アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DCS_APPL_HANDLE	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。

表 119. スナップショット・モニター API 要求タイプ (続き)

モニター・レベル	API 要求タイプ	戻される情報
アプリケーション	SQLMA_DCS_DBASE_APPLS	パーティション上にあるデータベースに接続されている各 DCS アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DBASE_APPLS_REMOTE	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_APPL_REMOTE_ALL	パーティション上にあるアクティブな各フェデレーテッド・システム・アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
表	SQLMA_DBASE_TABLES	データベースに接続された各アプリケーションのデータベース・レベルとアプリケーション・レベルの表アクティビティー情報。データベースに接続されたアプリケーションが**アクセスした**各表の表レベルの表アクティビティー情報。表スイッチが必要です。
ロック	SQLMA_APPL_LOCKS	アプリケーションによって保持されているロックのリスト。ロック待機情報にはロック・スイッチが必要です。
ロック	SQLMA_APPL_LOCKS_AGENT_ID	アプリケーションによって保持されているロックのリスト。ロック待機情報にはロック・スイッチが必要です。
ロック	SQLMA_DBASE_LOCKS	データベースに接続された各アプリケーションについての、データベース・レベルおよびアプリケーション・レベルのロック情報。ロック・スイッチが必要です。
表スペース	SQLMA_DBASE_TABLESPACES	データベース・レベルの表スペース・アクティビティーに関する情報。また、データベースに接続された各アプリケーションのアプリケーション・レベル、およびデータベースに接続された各アプリケーションがアクセスしている各表スペースの表スペース・レベルの情報も含みます。バッファー・プール・スイッチが必要です。
バッファー・プール	SQLMA_BUFFERPOOLS_ALL	バッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
バッファー・プール	SQLMA_DBASE_BUFFERPOOLS	指定されたデータベースのバッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。

表 119. スナップショット・モニター API 要求タイプ (続き)

モニター・レベル	API 要求タイプ	戻される情報
動的 SQL	SQLMA_DYNAMIC_SQL	データベースの SQL ステートメント・キャッシュからのポイント・イン・タイム指定ステートメント情報。

スナップショット・モニターの出力例

スナップショット・モニターで作業中に、サンプル出力を検討するのが役立つ場合があります。

スナップショット・モニターの性質を示すため、以下に、CLP を使用して取られるスナップショットの例とそれに対応する出力を示します。この例の目的は、SAMPLE データベースに接続されたアプリケーションが保持しているロックのリストを入手することです。行われるステップは次のとおりです。

1. 次のようにしてサンプル・データベースに接続します。

   ```
   db2 connect to sample
   ```

2. 次のように、UPDATE MONITOR SWITCHES コマンドを使用して「ロック」スイッチを ON にし、ロックのために待機した時間を収集します。

   ```
   db2 update monitor switches using LOCK on
   ```

3. データベース・カタログでロックを必要とするコマンドまたはステートメントを発行します。この場合、次のようにカーソルの宣言、オープン、およびフェッチを行います。

   ```
   db2 -c- declare c1 cursor for
                          select * from staff where job='Sales' for update
   db2 -c- open c1
   db2 -c- fetch c1
   ```

4. 次のように GET SNAPSHOT コマンドを使用して、データベース・ロックのスナップショットを取ります。

   ```
   db2 get snapshot for locks on sample
   ```

CLP から GET SNAPSHOT コマンドが発行された後、スナップショット出力が画面に送られます。

```
            Database Lock Snapshot

Database name                              = SAMPLE
Database path                              = C:¥DB2¥NODE0000¥SQL00001¥
Input database alias                       = SAMPLE
Locks held                                 = 5
Applications currently connected           = 1
Agents currently waiting on locks          = 0
Snapshot timestamp                         = 06-05-2002 17:08:25.048027

Application handle                         = 8
Application ID                             = *LOCAL.DB2.0098C5210749
Sequence number                            = 0001
Application name                           = db2bp.exe
CONNECT Authorization ID                   = DB2ADMIN
Application status                         = UOW Waiting
Status change time                         = Not Collected
Application code page                      = 1252
Locks held                                 = 5
Total wait time (ms)                       = 0
```

```
List Of Locks
 Lock Name                   = 0x02000300050000000000000052
 Lock Attributes             = 0x00000000
 Release Flags               = 0x00000001
 Lock Count                  = 1
 Hold Count                  = 0
 Lock Object Name            = 5
 Object Type                 = Row
 Tablespace Name             = USERSPACE1
 Table Schema                = DB2ADMIN
 Table Name                  = STAFF
 Mode                        = U

 Lock Name                   = 0x02000300000000000000000054
 Lock Attributes             = 0x00000000
 Release Flags               = 0x00000001
 Lock Count                  = 1
 Hold Count                  = 0
 Lock Object Name            = 3
 Object Type                 = Table
 Tablespace Name             = USERSPACE1
 Table Schema                = DB2ADMIN
 Table Name                  = STAFF
 Mode                        = IX

 Lock Name                   = 0x0100000001000000100810056
 Lock Attributes             = 0x00000000
 Release Flags               = 0x40000000
 Lock Count                  = 1
 Hold Count                  = 0
 Lock Object Name            = 0
 Object Type                 = Internal Variation Lock
 Mode                        = S

 Lock Name                   = 0x41414141414A48520000000041
 Lock Attributes             = 0x00000000
 Release Flags               = 0x40000000
 Lock Count                  = 1
 Hold Count                  = 0
 Lock Object Name            = 0
 Object Type                 = Internal Plan Lock
 Mode                        = S

 Lock Name                   = 0x434F4E544F4B4E310000000041
 Lock Attributes             = 0x00000000
 Release Flags               = 0x40000000
 Lock Count                  = 1
 Hold Count                  = 0
 Lock Object Name            = 0
 Object Type                 = Internal Plan Lock
 Mode                        = S
```

このスナップショットを見ると、現在 1 つのアプリケーションが SAMPLE データベースに接続しており、 5 つのロックを保持していることが分かります。

```
    Locks held                                = 5
    Applications currently connected          = 1
```

Application status が UOW Waiting になった時刻 (Status change time) は Not Collected として戻されることに注意してください。これは、「作業単位」スイッチが OFF であるためです。

ロック・スナップショットは、このデータベースに接続しているアプリケーションがロックのために待機している、現在までの合計時間も戻します。

```
Total wait time (ms)                        = 0
```

サブセクション・スナップショット

パーティション間並列処理を使用しているシステムでは、SQL コンパイラーによって、 SQL ステートメントのアクセス・プランがサブセクションに区分化されます。個々のサブセクションは別々の DB2 エージェント (または SMP のエージェント) によって実行されます。

コンパイル時に DB2 コード生成プログラムによって生成される SQL ステートメントのアクセス・プランを取得するには、 db2expln コマンドを使用します。一例として、複数のパーティションにパーティション化されている表の行をすべて選択すると、アクセス・プランは以下の 2 つのサブセクションに分けられます。

1. サブセクション 0。コーディネーター・サブセクション。この役割は、他の DB2 エージェント (サブエージェント) にフェッチされた行を収集してアプリケーションに戻すことです。
2. サブセクション 1。この役割は、表スキャンを実行して、行をコーディネーター・エージェントに戻すことです。

この単純な例では、サブセクション 1 はすべてのデータベース・パーティションに分散されます。この表が属するデータベース・パーティション・グループの個々の物理パーティションに、このサブセクションを実行するサブエージェントがあります。

データベース・システム・モニターを使用すると、ランタイムの情報とアクセス・プラン (コンパイル時の情報) とを関連付けることができます。パーティション間並列処理により、モニターは情報をサブセクションのレベルに分類します。例えば、ステートメント・モニターのスイッチが ON になっている場合に、 GET SNAPSHOT FOR APPLICATION を実行すると、ステートメント全体の情報と、このパーティション上で実行される個々のサブセクションに関する情報が戻されます。

アプリケーションのスナップショットを実行すると、以下のサブセクション情報が戻されます。

- 読み書きされた表の行数
- CPU の使用量
- 経過時間
- このステートメントで作業する他のエージェントとの間で送受信される表キューの行数。これにより、スナップショットを連続してとることで、実行時間の長い照会の実行状況を追跡できます。
- サブセクションの状況。サブセクションが WAIT 状態 (別のエージェントがデータを送受信するのを待機している) の場合は、この情報によって、サブセクションの実行を妨げているパーティションも識別できます。識別した後にそのパーティションでスナップショットを取って状態を調べることができます。

この情報は、ステートメント・イベント・モニターによって、サブセクションごとに実行完了後に記録されます。この情報には CPU 使用量、合計実行時間、および他の複数のカウンターが含まれます。

パーティション・データベース・システムでのグローバル・スナップショット

パーティション・データベース・システムでは、スナップショット・モニターを使用して現行パーティション、指定したパーティション、またはすべてのパーティションのスナップショットをとることができます。パーティション・データベースのすべてのパーティションに渡ってグローバル・スナップショットをとる場合は、データが集約されてから、結果が戻されます。

データは、以下のようなさまざまなエレメント・タイプについて集約されます。

- **カウンター、時間、ゲージ**

  インスタンス内のそれぞれのパーティションから収集されたすべての同種値の合計が入っています。例えば、GET SNAPSHOT FOR DATABASE XYZ ON TEST GLOBAL は、パーティション・データベース・インスタンス内のすべてのパーティションについて、データベースから読み取られた行数 (rows_read) を戻します。

- **水準点**

  パーティション・データベース・システム内のパーティションについて検出される最高値 (高位水準点) または最低値 (低位水準点) を戻します。戻された値に注目する場合は、個々のパーティションのスナップショットを取って、特定のパーティションが使用過剰になっているのか、またはインスタンス全体に問題があるのかを判別することができます。

- **タイム・スタンプ**

  スナップショット・モニター・エージェントがアタッチされているパーティションのタイム・スタンプ値に設定します。すべてのタイム・スタンプ値は、「**タイム・スタンプ**」モニター・スイッチの制御下にあります。

- **情報**

  作業を妨害している可能性のあるパーティションに関する最も重要な情報を戻します。例えば、エレメント `appl_status` について、あるパーティションでの状況が「UOW 実行」であり、別のパーティションでは「ロック待機」の場合には、「ロック待機」が戻されます。なぜなら、これはアプリケーションの実行を保留にしている状況だからです。

さらに、カウンターのリセット、モニター・スイッチの設定、モニター・スイッチ設定値の検索はパーティション・データベース内の個々のパーティション、またはすべてのパーティションに対して行うことができます。

注: グローバル・スナップショットをとる際に、1 つ以上のパーティションでエラーが生じると、データはスナップショットを正常にとることのできたパーティションから収集され、さらに警告 (sqlcode 1629) が戻されます。モニター・スイッチのグローバル取得または更新が失敗したり、 1 つ以上のパーティションでカウンター・リセットが失敗すると、それらのパーティションではスイッチの設定やデータのリセットは行われません。

スナップショット・モニター自己記述型データ・ストリーム

db2GetSnapshot API でスナップショットをキャプチャーした後、スナップショット出力が自己記述型データ・ストリームとして戻されます。

図 6 では、データ・ストリームの構造を示し、 548 ページの表 120 では、戻される論理データ・グループおよびモニター・エレメントのいくつかの例を示します。

注: これらの例や表では、ID として記述名を使用しています。これらの名前は、実際のデータ・ストリームでは **SQLM_ELM_** という接頭部が付きます。例えば collected は、スナップショット・モニター出力で SQLM_ELM_COLLECTED と表示されます。タイプは、実際のデータ・ストリームでは **SQLM_TYPE_** という接頭部が付きます。例えば、ヘッダーはデータ・ストリームで SQLM_TYPE_HEADER と表示されます。

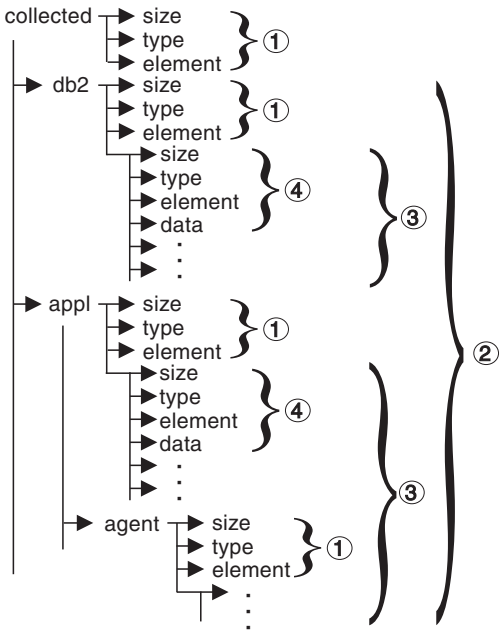


図 6. スナップショット・モニター・データ・ストリーム

1. 各論理データ・グループは、サイズと名前を示すヘッダーで始まります。このサイズには、ヘッダー自体に取られるデータ・ボリュームは組み込まれません。
2. collected ヘッダー内のサイズは、スナップショットの合計サイズを戻します。
3. ほかのヘッダー内のサイズ・エレメントは、その論理データ・グループのデータ全体のサイズを示します (従属のグループをすべて含む)。
4. モニター・エレメント情報が、論理データ・グループ・ヘッダーに続きます。これも自己記述型です。

表 120. スナップショット・データ・ストリームの例

論理データ・グループ	データ・ストリーム	説明
collected		
	1000	スナップショット・データのサイズ (バイト)。
	header	論理データ・グループが始まることを示す。
	collected	論理データ・グループの名前。
	4	このモニター・エレメントに入っているデータのサイズ。
	u32bit	モニター・エレメント・タイプ
	server_db2_type	- 符号なし 32 ビット数値。
	sqlf_nt_server	収集されたモニター・エレメントの名前。
		このエレメントに対して収集された値。
	2	このモニター・エレメントに入っているデータのサイズ。
	u16bit	モニター・エレメント・タイプ
	node_number	- 符号なし 16 ビット数値。
	3	収集されたモニター・エレメントの名前。
		このエレメントに対して収集された値。
db2		
	200	スナップショットのデータの DB2
	header	レベル部分のサイズ。
	db2	論理データ・グループが始まることを示す。
		論理データ・グループの名前。
	4	このモニター・エレメントに入っているデータのサイズ。
	u32bit	モニター・エレメント・タイプ
	sort_heap_allocated	- 符号なし 32 ビット数値。
	16	収集されたモニター・エレメントの名前。
		このエレメントに対して収集された値。
	4	このモニター・エレメントに入っているデータのサイズ。
	u32bit	モニター・エレメント・タイプ
	local_cons	- 符号なし 32 ビット数値。
	3	収集されたモニター・エレメントの名前。
		このエレメントに対して収集された値。
	. . .	. . .
appl		
	100	スナップショットの appl エレメント・データのサイズ。
	header	論理データ・グループが始まることを示す。
	appl	論理データ・グループの名前。
	4	このモニター・エレメントに入っているデータのサイズ。
	u32bit	モニター・エレメント・タイプ
	locks_held	- 符号なし 32 ビット数値。
	3	収集されたモニター・エレメントの名前。
		このエレメントに対して収集された値。
	. . .	. . .

表 120. スナップショット・データ・ストリームの例 (続き)

論理データ・グループ	データ・ストリーム	説明
agent		
	50	appl 構造のエージェント部分のサイズ。
	header	論理データ・グループが始まることを示す。
	agent	論理データ・グループの名前。
	4	このモニター・エレメントに入っているデータのサイズ。
	u32bit	モニター・エレメント・タイプ
	agent_pid	- 32 ビット数値。
	12	収集されたモニター・エレメントの名前。 このエレメントに対して収集された値。
	. . .	. . .

db2GetSnapshot() ルーチンは、自己記述型スナップショット・データを、ユーザー提供のバッファーに戻します。データは、キャプチャーされたスナップショットのタイプに関連する論理データ・グループに分けられて戻されます。

スナップショット要求によって戻される各アイテムには、そのサイズおよびタイプを指定するフィールドが含まれます。サイズを使用して、戻りデータ全体を解析できます。また、フィールド・サイズを使用して、論理データ・グループを読み飛ばすこともできます。例えば、DB2 レコードを読み飛ばすには、データ・ストリーム内のバイト数を判別する必要があります。読み飛ばすバイト数を計算するには、次の公式を使用してください。

```
db2 論理データ・グループのサイズ + sizeof(sqlm_header_info)
```

対話モードで db2top を実行してモニターするときに使用できるコマンド

db2top モニター・ユーティリティーは、複雑な DB2 環境を短時間で効率的にモニターするためのツールです。すべてのデータベース・パーティションの DB2 スナップショット情報を結合して、実行中の DB2 システムの動的なリアルタイム・ビューを提供できます。このツールの操作には、テキスト・ベースのユーザー・インターフェースを使用します。

このタスクについて

対話モードで **db2top** を実行する場合は、以下のコマンドを実行できます。

A HADR クラスターの 1 次データベースと 2 次データベースのいずれかをモニターします。

a エージェントのアプリケーション詳細に移動します (ステートメント画面ではエージェントに関する制限に移動します)。**db2top** コマンドから、エージェント ID を入力するためのプロンプトが表示されます。

B 重要なサーバー・リソースの主なコンシューマーを表示します (ボトルネック分析)

c このオプションによって、画面に表示する列の順序を変更できます。構文は

1,2,3,... という形式です (1,2,3 は、表示する 1 番目、2 番目、3 番目の列に相当します)。これらの数字は、ソート基準を指定するときに使用する列番号にもなります。

c スイッチ・キーを使用すると、画面に表示する列の順序を指定するための画面が表示されます。画面の左側にはデフォルトの順序と列番号が表示され、画面の右側には現在の配列が表示されます。列の順序を変更するには、画面の下部にあるテキスト・フィールドに新しい列の順序を入力します。次に、左側に表示されている列の相対的な位置を入力し、それぞれをコンマで区切ります。すべての列を指定しなければならないわけではありません。w を選択すれば、この列の配列を *$DB2TOPRC* に保存して、後続の **db2top** モニター・セッションで使用できるようになります。画面に表示する列をソートして、どの順序で配列するかを選択できます。 `.db2toprc` ファイルの中で列の配列のために使用できる有効なキーワードは、以下のとおりです。

- sessions=
- tables=
- tablespaces=
- bufferpools=
- dynsql=
- statements=
- locks=
- utilities=
- federation=

b バッファー・プール画面に移動します。

C スナップショット・データ収集プログラムのオン/オフを切り替えます。

d データベース画面に移動します。

D 動的 SQL 画面に移動します。

f 画面をフリーズします。

F 1 次サーバーでフェデレーテッド照会をモニターします。

G グラフのオン/オフを切り替えます。

h ヘルプ画面に移動します。

H 履歴画面に移動します。

i アイドル・セッションのオン/オフを切り替えます。

k 実際の値と差分の値を切り替えます。

l セッション画面に移動します。

L SQL 画面から完全な照会テキストを表示します。その後、e オプションまたは X オプションを使用して、通常の DB2 Explain を実行できます。

m メモリー・プールを表示します。

o セッション・セットアップを表示します。

p パーティション画面に移動します。

P スナップショットを実行するデータベース・パーティションを選択します。

q db2top を終了します。

R スナップショット・データをリセットします。

s ステートメント画面に移動します。

S ネイティブ DB2 スナップショットを実行します。

t 表スペース画面に移動します。

T 表画面に移動します。

u アクティブなユーティリティーを表示して、各データベース・パーティションの情報を集約します。

U ロック画面に移動します。

V デフォルトの Explain スキーマを設定します。

w セッション設定を .db2toprc に書き込みます。

W agent_id、os_user、db_user、application、netname の監視モード。セッション・スナップショットから返されるステートメント (オプション l) は、agent.sql、os_user-agent.sql、db_user-agent.sql、application-agent.sql、netname-agent.sql に書き込まれます。動的 SQL 画面から実行すると (オプション D)、ステートメントは、db2advis と互換性のある形式で db2adv.sql に書き込まれます。

X 拡張モードのオン/オフを切り替えます。

z|Z 昇順または降順でソートします。

/ データのフィルターとして使用する式を入力します。式は正規表現に準拠している必要があります。機能 (画面) ごとに別々のフィルターを適用できます。行全体に regexp チェックが適用されます。

<|> 画面の左側または右側に移動します。

以下のスイッチは、アプリケーション画面だけに該当します。

r 前の機能に戻ります。

R 自動リフレッシュを切り替えます。

g グラフのオン/オフを切り替えます。

X 拡張モードのオン/オフを切り替えます。

d エージェントを表示します。

対話モードで **db2top** を開始する場合は、以下のコマンドを実行します。

```
db2top -d <database name>
```

以下のコマンドを入力します。

```
db2top -d sample
```

以下の出力が表示されます。

```
[¥]11:57:10,refresh=2secs(0.000) Inactive,part=[1/1],<instanceName>:sample
[d=Y,a=N,e=N,p=ALL] [qp=off]

[/]: When rotating, it means that db2top is waiting between two snapshots, otherwise, it means db2top is waiting
     from an answer from DB2
11:57:10: current time
```

```
refresh=2secs: time interval
refresh=!secs: Exclamation mark means the time to process the snapshot by DB2 is longer than the refresh interval.
     In this case, db2top will increase the interval by 50%. If this occurs too often because the system is too busy,
     you can either increase the snapshot interval (option I), monitor a single database partition (option P), or turn
     off extended display mode (option x)
0.000 : time spent inside DB2 to process the snapshot
d=Y/N : delta or cumulative snapshot indicator (command option -k or option k).
a=Y/N : Active only or all objects indicator (-a command option set or i)
e=Y/N : Extended display indicator
p=ALL : All database partitions
p=CUR: Current database partition (-P command option with no partition number specified)
p=3 : target database partition number: say 3

Inactive: : Shows inactive if DB2 is not running, otherwise displays the platform on which DB2 is running
part=[1/1] : active database partition number vs total database partition number. For example, part=[2,3] means one
      database partition out of 3 is down (2 active, 3 total)
<instanceName> : instance name
sample : database name
```

例

パーティション・データベース環境で **db2top** モニター・ユーティリティーを対話モードで実行する場合の例を以下に示します。

```
db2top -d TEST -n mynode -u user -p passwd -V skm4 -B -i 1
The command parameters are as follows:
-d TEST     # database name
-n mynode   #  node name
-u user     #  user id
-p passwd   #  password
-V skm4     #  Schema name
-B          #  Bold enabled
-i 1        #  Screen update interval: 1 second
```

.db2toprc 構成ファイル

.db2toprc 構成ファイルは、**db2top** モニター・ユーティリティーの初期化時にパラメーターを設定するために使用するユーザー生成ファイルです。

db2top ユーティリティーは、ユーザー定義変数 *$db2topRC* を使用して、.db2toprc ファイルの場所を検索します。その変数が設定されていない場合、**db2top** はまず現行ディレクトリーで .db2toprc ファイルを検索してから、home ディレクトリーの中を検索します。.db2toprc は、ユーザー生成ファイルです。

環境変数

以下の環境変数を設定できます。

- **DB2TOPRC**

  .db2toprc ファイルの場所を格納するユーザー定義環境変数。例えば、Linux の場合は、**DB2TOPRC** を export db2topRC="~/db2top" と定義できます。

  その変数がユーザーによって設定されていない場合、**db2top** はまず現行ディレクトリーで .db2toprc ファイルを検索してから、home ディレクトリーの中を検索します。

- **DB2DBDFT**

  この変数では、暗黙接続で使用するデータベースのデータベース別名を指定します。コマンド行または .db2toprc 構成ファイルでデータベース名が指定されていない場合に、その変数が使用されます。

- **EDITOR**

このシステム環境変数では、Explain スナップショットまたはネイティブ・スナップショットの結果を表示するためのテキスト・エディターを開始するときに使用するコマンドを指定します。

この変数が設定されていない場合は、**vi** が使用されます。

構造

ここでは、.db2toprc ファイルのいくつかの項目を取り上げます。

cpu=command
この項目を使用して、画面出力の右側の第 2 行に CPU アクティビティーの結果を表示します。例えば、

`cpu=vmstat 2 2 | tail -1 | awk '{printf("%d(usr+sys)",$14+$15);}'` 画面の右側に `Cpu=2(usr+sys)` が表示されます。

io=command
この項目を使用して、コマンドを指定し、画面出力の左側の第 2 行に結果を表示します。例えば、

`io=vmstat 2 2 | tail -1 | awk '{printf("%d(bi+bo)",$10+$11);}'` 画面の左側に `Disk=76(bi+bo)` が表示されます。

どちらのコマンドもバックグラウンド・プロセスとして実行され、画面の各フィールドが非同期モードで更新されます。

shell alias=command
このシェル項目を使用して、ユーザー定義コマンドを指定します。例えば、`shell M=top` を指定した場合は、M を入力すると、**db2top** セッションからトップが作成されます。終了時には、現在の画面に戻ります。

function alias=command
この項目を使用して、ユーザー定義コマンドを指定します。例えば、`function N=netstat` を指定した場合は、**netstat** の出力を繰り返し表示する N という新しい関数が作成されます。function 項目は、複数記述できます。それぞれの項目を別々の行に配置する必要があります。例えば、

```
function Q=netstat
function N=df -k
```

sort=command
この項目を使用して、ソート順を指定します。例えば、sort=command を指定すると、その関数のデフォルトのソート順が作成されます (command は列の番号です)。昇順と降順のいずれかになります。ソートは、sessions、tables、tablespace、bufferpool、dynsql、statements、locks、utilities、federation で有効です。

サンプル .db2toprc ファイル

デフォルトの .db2toprc 構成ファイルはありません。ただし、「W」を押せば、現在のセットアップの .db2toprc を作成できます。以下のサンプル **.db2toprc** ファイルを参考にしてください。すべての項目にコメントを追加しています。

```
# db2top configuration file
# On UNIX, should be located in $HOME/.db2toprc
# File generated by db2top-1.0a
```

```
#
node= # [-n] nodename
database=sample # [-d] databasename
user= # [-u] database user
password= # [-p] user password (crypted)
schema= # [-V] default schema for explains
interval=2 # [-i] sampling interval
active=OFF # [-a] display active sessions only (on/off)
reset=OFF # [-R] Reset snapshot at startup (on/off)
delta=ON # [-k] Toggle display of delta/cumulative values (on/off)
gauge=ON # display graph on sessions list (on/off)
colors=ON
# True if terminal supports colors. Informs GE_WRS if it can display information with colors
graphic=ON  # True if terminal supports semi graphical characters (on/off).
port= # Port for network collection
streamsize=size # Max collection size per hour (eg. 1024 or 1K : K, M or G)
# Command to get cpu usage information from OS
cpu=vmstat 2 2 | tail -1 | awk '{printf("%d(usr+sys)",$14+$15);}'
# Command to get IO usage information from OS
io=vmstat 2 2 | tail -1 | awk '{printf("%d(bi+bo)",$10+$11);}'
# Ordering of information in sessions screen
# Column order for the session screen (option l)
sessions=0,1,18,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19,20,21,22,23
# Column order for the tables screen (option T)
tables=0,1,2,4,3,5,6,7
# Column order for the tablespaces screen (option t).
# The display will be sorted in ascending order on column #22
tablespaces=0,1,18,2,3,4,5,6,7,8, sort=22a
# Column order for the bufferpool screen (option b)
bufferpools=0,1,18,2,3,4,5,6,7,8,9,10
# Column order for the Dynamic SQL screen (option D)
dynsql=0,1,18,2,3,4,5,6,7,8,9
statements=0,1
locks=0,1
utilities=0 # contains the default column and sort order for the utility screen
federation=0,2,4  # contains the default column and sort order for the federation screen

# User defined commands
shell P=top
function N=date && netstat -t tcp
```

スイッチ・ベースのモニターの概念

システム・モニター・スイッチ

システム・モニター・スイッチはスナップショット・モニターや一部のイベント・モニターがデータを収集する方法を制御します。

注: これらのシステム・モニター・スイッチによる、作業単位イベント・モニターおよびロック・イベント・モニター (DB2 バージョン 9.7 で導入されています) への影響はありません。

スナップショット・モニターおよび一部のイベント・モニターは、システム・モニターが収集したデータを報告します。 システム・モニター・データを収集すると、データベース・マネージャーの処理が余分に発生します。例えば、SQL ステートメントの実行時間を計算するには、すべてのステートメントの実行前後にデータベース・マネージャーがオペレーティング・システムを呼び出して、タイム・スタンプを入手しなければなりません。この種のシステム呼び出しには通常はコストがかかります。システム・モニターを使用すると、メモリーの消費量も増加します。システム・モニターによって追跡されるすべてのモニター・エレメントについて、データベース・マネージャーはメモリーを使用し、収集されたデータを保管します。

モニター情報の保守に関連したプロセッサー時間の増加を最小限にとどめるために、モニター・スイッチを使って、高コストになりかねないデータベース・マネー

ジャーによるデータ収集を制御します。各スイッチには、ON と OFF という 2 つの設定値だけがあります。モニター・スイッチが OFF の場合には、そのスイッチの制御下にあるモニター・エレメントは情報を収集しません。スイッチの制御下にはなく、またスイッチの設定に関係なく常に収集される、基本モニター・データは相当量あります。

それぞれのモニター・アプリケーションには、モニター・スイッチ (およびシステム・モニター・データ) の独自の論理ビューがあります。始動時に、各アプリケーションはそのモニター・スイッチ設定値を、(インスタンス・レベルの) データベース・マネージャー構成ファイル内の dft_monswitches パラメーターから継承します。モニター・アプリケーションは UPDATE MONITOR SWITCHES USING MONSWITCH OFF/ON コマンドを使用して、モニター・スイッチ設定値を変更することができます。 MONSWITCH パラメーターは、次のセクションに示す「スナップショット・モニター・スイッチ」表の「モニター・スイッチ」欄の値を保持しています。スイッチの設定値をアプリケーション・レベルで変更すると、スイッチを変更したアプリケーションだけに影響が及びます。

インスタンス・レベルのモニター・スイッチの変更は、データベース管理システムを停止せずに行うことができます。これを行うには、UPDATE DBM CFG USING DBMSWITCH OFF/ON コマンドを使用します。 DBMSWITCH パラメーターは、次のセクションに示す「スナップショット・モニター・スイッチ」表の「DBM パラメーター」欄の値を保持しています。このようにスイッチを動的に更新する際、更新を動的に有効にするためには、更新を実行しているアプリケーションを明示的にインスタンスにアタッチする必要があります。動的な更新を行っても、他の既存のスナップショット・アプリケーションへの影響はありません。新規のモニター・アプリケーションは、更新済みのインスタンス・レベルのモニター・スイッチ設定値を継承します。既存のモニター・アプリケーションが新規のデフォルトのモニター・スイッチ値を継承するには、いったん終了して、そのアタッチメントを再構築する必要があります。データベース・マネージャー構成ファイルのスイッチを更新すると、パーティション・データベース内のすべてのパーティションのスイッチも更新されます。

データベース・マネージャーでは、スナップショット・モニター・アプリケーションとそのスイッチの設定値がすべて追跡されます。 1 つのアプリケーションの構成内でスイッチが ON に設定されている場合は、データベース・マネージャーは必ずモニター・データを収集します。したがって、アプリケーションの構成内で、あるスイッチが OFF になっている場合でも、その同じスイッチが ON になっているアプリケーションが少なくとも 1 つある限り、データベース・マネージャーはデータを収集します。

時間およびタイム・スタンプ・エレメントの収集は、「タイム・スタンプ」スイッチによって制御されます。このスイッチを OFF にすると (デフォルトでは ON)、時間またはタイム・スタンプに関連したモニター・エレメントを判別するときにオペレーティング・システムが呼び出すタイム・スタンプをすべてスキップするよう、データベース・マネージャーに指示されます。 CPU の使用率が 100% に近づいたときには、このスイッチを OFF にすることが重要となります。そのような状況では、タイム・スタンプの発行によって、かなりのパフォーマンス低下が生じます。「タイム・スタンプ」スイッチと他のスイッチによって制御できるモニター・エレメントの場合は、それらのスイッチのいずれかを OFF にするとデータは収集

されません。そのため、「タイム・スタンプ」スイッチを OFF にすると、他のモニター・スイッチの制御下にあるデータの全体コストが大幅に減少します。

イベント・モニターは、スナップショット・モニター・アプリケーションと同じような、モニター・スイッチによる影響は受けません。イベント・モニターが定義されると、指定されたイベント・タイプにより必要とされるインスタンス・レベルのモニター・スイッチを、自動的に ON にします。例えば、デッドロック・イベント・モニターは、「ロック」モニター・スイッチを自動的に ON にします。イベント・モニターが活動化されると、必要なモニター・スイッチが ON になります。イベント・モニターが非アクティブ化されると、モニター・スイッチは OFF になります。

「タイム・スタンプ」モニター・スイッチは、イベント・モニターによって自動的に設定されません。これは、イベント・モニターの論理データ・グループに属するモニター・エレメントの収集を制御する、唯一のモニター・スイッチです。「タイム・スタンプ」スイッチが OFF の場合は、イベント・モニターによって収集されるタイム・スタンプおよび時間モニター・エレメントの大半が収集されません。これらのエレメントは、依然として指定された表やファイル、またはパイプに書き込まれますが、その値はゼロとなります。

表 121. スナップショット・モニター・スイッチ

モニター・スイッチ	DBM パラメーター	提供される情報
BUFFERPOOL	DFT_MON_BUFPOOL	読み取りおよび書き込みの数、かかった時間
LOCK	DFT_MON_LOCK	ロック待機時間、デッドロック
SORT	DFT_MON_SORT	使用されたヒープ数、ソート・パフォーマンス
STATEMENT	DFT_MON_STMT	開始/停止時刻、ステートメント識別
TABLE	DFT_MON_TABLE	アクティビティーの程度 (読み取られた/書き込まれた行)
UOW	DFT_MON_UOW	開始/終了時刻、完了状況
TIMESTAMP	DFT_MON_TIMESTAMP	タイム・スタンプ

スナップショットをキャプチャーしたり、イベント・モニターを使用したりする前に、データベース・マネージャーに収集させる必要のあるデータを指定する必要があります。次の特殊タイプ・データのいずれかをスナップショットで収集する場合には、該当するモニター・スイッチを設定する必要があります。

- バッファー・プール・アクティビティー情報
- ロック、ロック待機、および時間関連のロック情報
- ソート情報
- SQL ステートメント情報
- 表アクティビティー情報
- 時間およびタイム・スタンプ情報
- 作業単位情報

上記の情報タイプに対応するスイッチは、デフォルトではすべて OFF になっています。ただし、時間およびタイム・スタンプ情報に対応するスイッチだけは、デフォルトで ON になっています。

イベント・モニターは、時間およびタイム・スタンプ情報スイッチによってのみ影響を受けます。その他のすべてのスイッチ設定は、イベント・モニターによって収集されるデータには影響しません。

CLP からのシステム・モニター・スイッチの設定

システム・モニター・スイッチは、システム・モニターによるデータの収集を制御します。 特定のモニター・スイッチを ON に設定することにより、特定のタイプのモニター・データを収集できます。

始める前に

モニター・スイッチの更新を実行するアプリケーションには、インスタンス・アタッチメントがなければなりません。次のコマンドを使用するには、SYSADM、SYSCTRL、SYSMAINT、または SYSMON のいずれかの権限が必要です。

- UPDATE MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET DATABASE MANAGER MONITOR SWITCHES

UPDATE DBM CFG コマンドを使用するには、SYSADM 権限を持っている必要があります。

このタスクについて

手順

- ローカル・モニター・スイッチを活動化するには、UPDATE MONITOR SWITCHES コマンドを使用する。アプリケーション (CLP) がデタッチするか、または別の UPDATE MONITOR SWITCHES コマンドによってスイッチが非アクティブ化されるまで、スイッチはアクティブのままです。次の例では、ローカル・モニター・スイッチをすべて ON に更新します。

  ```
  db2 update monitor switches using BUFFERPOOL on LOCK on
      SORT on STATEMENT on TIMESTAMP on TABLE on UOW on
  ```

- ローカル・モニター・スイッチを非アクティブ化するには、UPDATE MONITOR SWITCHES コマンドを使用する。次の例では、ローカル・モニター・スイッチをすべて OFF に更新します。

  ```
  db2 update monitor switches using BUFFERPOOL off, LOCK off,
      SORT off, STATEMENT off, TIMESTAMP off, TABLE off, UOW off
  ```

  以下は、上記の UPDATE MONITOR SWITCH コマンドを発行した後に表示される出力例です。

  ```
            Monitor Recording Switches

  Switch list for db partition number 1
  Buffer Pool Activity Information (BUFFERPOOL) = OFF
  Lock Information                         (LOCK) = OFF
  Sorting Information                      (SORT) = OFF
  SQL Statement Information              (STATEMENT) = OFF
  ```

```
Table Activity Information            (TABLE) = OFF
Unit of Work Information                (UOW) = OFF
Get timestamp information         (TIMESTAMP) = OFF
```

- モニター・スイッチをデータベース・マネージャー・レベルで操作することもできる。これには、UPDATE DBM CFG コマンドを使用して、データベース・マネージャー構成ファイル内の dft_monswitches パラメーターを変更することが必要となります。以下の例では、基本情報に加えて、ロック・スイッチによって制御される情報だけが収集されます。

  ```
  db2 update dbm cfg using DFT_MON_LOCK on
  ```

  モニター・アプリケーションが開始されると、必ずデータベース・マネージャーからそのモニター・スイッチ設定を継承します。データベース・マネージャーのモニター・スイッチ設定を変更しても、実行中のモニター・アプリケーションには影響を与えません。モニター・アプリケーションは、自分自身をインスタンスに再度アタッチして、モニター・スイッチ設定の変更を取り出す必要があります。

- パーティション・データベース・システムの場合、特定のパーティションについてモニター・スイッチを固有に設定するか、またはすべてのパーティションについてグローバルに設定することができる。

  1. 特定のパーティション (例えば、パーティション番号 3) に対してモニター・スイッチ (例えば、BUFFERPOOL) を設定するには、次のコマンドを発行します。

     ```
     db2 update monitor switches using BUFFERPOOL on
         at dbpartitionnum 3
     ```

  2. すべてのパーティションについてモニター・スイッチ (例えば、SORT) を設定するには、次のコマンドを発行します。

     ```
     db2 update monitor switches using SORT on global
     ```

- ローカル・モニター・スイッチの状況をチェックするには、GET MONITOR SWITCHES コマンドを使用する。

  ```
  db2 get monitor switches
  ```

- パーティション・データベース・システムの場合、特定のパーティションについてモニター・スイッチ設定を固有に表示するか、またはすべてのパーティションについてグローバルな設定を表示することができる。

  1. 特定のパーティション (例えば、パーティション番号 2) に対してモニター・スイッチ設定を表示するには、次のコマンドを発行します。

     ```
     db2 get monitor switches at dbpartitionnum 2
     ```

  2. すべてのパーティションについてのモニター・スイッチ設定を表示するには、次のコマンドを発行します。

     ```
     db2 get monitor switches global
     ```

- データベース・マネージャー・レベル (またはインスタンス・レベル) でモニター・スイッチの状況をチェックするには、 GET DATABASE MANAGER MONITOR SWITCHES コマンドを使用する。このコマンドは、モニター対象となっているインスタンスのスイッチ設定全体を表示します。

  ```
  db2 get database manager monitor switches
  ```

  以下は、上記のコマンドを発行した後に表示される出力例です。

```
                DBM System Monitor Information Collected

  Switch list for db partition number 1
  Buffer Pool Activity Information (BUFFERPOOL) = OFF
  Lock Information                        (LOCK) = ON    10-25-2001 16:04:39
  Sorting Information                     (SORT) = OFF
  SQL Statement Information           (STATEMENT) = OFF
  Table Activity Information              (TABLE) = OFF
  Unit of Work Information                 (UOW) = OFF
  Get timestamp information           (TIMESTAMP) = OFF
```

タスクの結果

これで、目的のモニター・スイッチを設定し、スイッチ設定を確認したので、モニター・データのキャプチャーおよび収集の準備ができました。

クライアント・アプリケーションからのシステム・モニター・スイッチの設定

システム・モニター・スイッチは、システム・モニターによるデータの収集を制御します。 特定のモニター・スイッチを ON に設定することにより、特定のタイプのモニター・データを収集できます。

始める前に

モニター・スイッチの更新を実行するアプリケーションには、インスタンス・アタッチメントがなければなりません。 db2MonitorSwitches API を使用するには、SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。

手順

1. 次の DB2 ライブラリー sqlutil.h および db2ApiDf.h を組み込みます。これらは sqllib の下の include サブディレクトリーにあります。

   ```
   #include <sqlutil.h>
   #include <db2ApiDf.h>
   #include <string.h>
   #include <sqlmon.h>
   ```

2. スイッチ・リストのバッファー単位サイズを 1 KB に設定します。

   ```
   #define SWITCHES_BUFFER_UNIT_SZ 1024
   ```

3. sqlca、db2MonitorSwitches、および sqlm_recording_group 構造体を初期化します。また、スイッチ・リスト・バッファーを含むようにポインターを初期化し、バッファーのサイズを設定します。

   ```
   struct sqlca sqlca;
   memset (&sqlca, '¥0', sizeof(struct sqlca));
   db2MonitorSwitchesData switchesData;
   memset (&switchesData, '¥0', sizeof(switchesData));
   struct sqlm_recording_group switchesList[SQLM_NUM_GROUPS];
   memset(switchesList, '¥0', sizeof(switchesList));
   sqluint32 outputFormat;
   static sqluint32 switchesBufferSize = SWITCHES_BUFFER_UNIT_SZ;
   char *switchesBuffer;
   ```

4. スイッチ・リスト出力を保持するバッファーを初期化します。

   ```
   switchesBuffer = (char *)malloc(switchesBufferSize);
   memset(switchesBuffer, '¥0', switchesBufferSize));
   ```

5. ローカル・モニター・スイッチの状態を変更するには、 sqlm_recording_group 構造体内のエレメント (前のステップでは switchesList という名前) を変更しま

す。モニター・スイッチをオンにするには、パラメーター input_state を SQLM_ON に設定する必要があります。モニター・スイッチを OFF にするには、パラメーター input_state を SQLM_OFF に設定する必要があります。

```
switchesList[SQLM_UOW_SW].input_state = SQLM_ON;
switchesList[SQLM_STATEMENT_SW].input_state = SQLM_ON;
switchesList[SQLM_TABLE_SW].input_state = SQLM_ON;
switchesList[SQLM_BUFFER_POOL_SW].input_state = SQLM_OFF;
switchesList[SQLM_LOCK_SW].input_state = SQLM_OFF;
switchesList[SQLM_SORT_SW].input_state = SQLM_OFF;
switchesList[SQLM_TIMESTAMP_SW].input_state = SQLM_OFF;
switchesData.piGroupStates = switchesList;
switchesData.poBuffer = switchesBuffer;
switchesData.iVersion = SQLM_DBMON_VERSION9_5;
switchesData.iBufferSize = switchesBufferSize;
switchesData.iReturnData = 0;
switchesData.iNodeNumber = SQLM_CURRENT_NODE;
switchesData.poOutputFormat = &outputFormat;
```

注: iVersion が SQLM_DBMON_VERSION8 より小さい場合には、SQLM_TIMESTAMP_SW を使用できません。

6. スイッチ設定の変更を実行するには、db2MonitorSwitches() 関数を呼び出します。 db2MonitorSwitches API に対するパラメーターとして、db2MonitorSwitchesData (この例では switchesData) 構造体を渡します。switchesData には、パラメーターとして sqlm_recording_group 構造体が含まれています。

   ```
   db2MonitorSwitches(db2Version810, &switchesData, &sqlca);
   ```

7. スイッチ・リスト・バッファーからのスイッチ・リスト・データ・ストリームを処理します。
8. スイッチ・リスト・バッファーをクリアします。

   ```
   free(switchesBuffer);
   free(pRequestedDataGroups);
   ```

タスクの結果

これで、目的のモニター・スイッチを設定し、スイッチ設定を確認したので、モニター・データのキャプチャーおよび収集の準備ができました。

システム・モニター・スイッチ自己記述型データ・ストリーム

現行のシステム・モニター・スイッチ設定値を db2MonitorSwitches API を使って更新または表示すると、この API はスイッチ設定値を自己記述型データ・ストリームとして戻します。

561 ページの図 7 では、パーティション・データベース環境の場合に戻されるスイッチ・リスト情報の構造を示します。

注:

1. これらの例や表では、ID として記述名を使用しています。これらの名前は、実際のデータ・ストリームでは **SQLM_ELM_** という接頭部が付きます。例えば db_event は、イベント・モニター出力では SQLM_ELM_DB_EVENT と表示されます。タイプは、実際のデータ・ストリームでは **SQLM_TYPE_** という接頭部が付きます。例えば、ヘッダーはデータ・ストリームで SQLM_TYPE_HEADER と表示されます。

2. グローバル・スイッチ要求では、それぞれのスイッチ要求ごとに、情報が戻されるパーティションの順番が異なる可能性があります。この場合、パーティション ID がデータ・ストリームに組み込まれます。

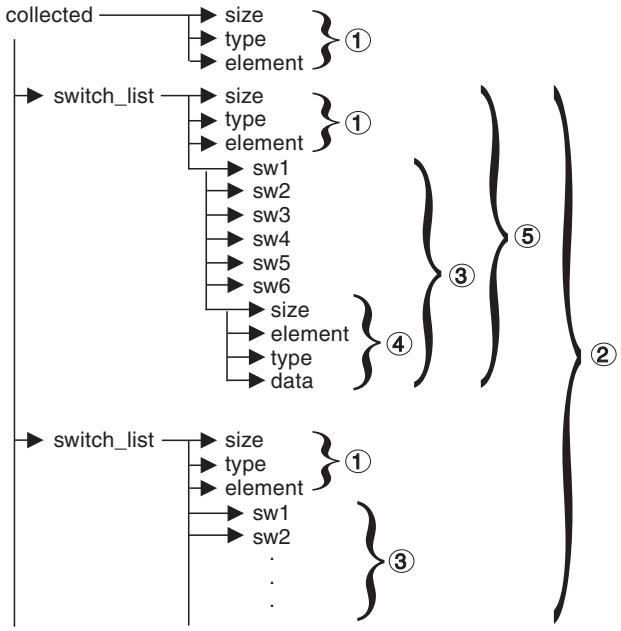


図 7. スイッチ・リスト・モニターのデータ・ストリーム

1. 各論理データ・グループは、サイズと名前を示すヘッダーで始まります。このサイズには、ヘッダー自体に取られるデータ・ボリュームは組み込まれません。
2. 収集したヘッダー内のサイズは、すべてのパーティションのすべてのモニター・スイッチ・リストの合計サイズを戻します。
3. スイッチ・リストのヘッダー内にあるサイズ・エレメントは、そのパーティションのスイッチ・データのサイズを表します。
4. スイッチ情報は、自己記述型です。
5. 非パーティション・データベースでは、独立型パーティションのスイッチ設定値が戻されます。つまり、スイッチ・リストが 1 つだけ戻されます。

データベース・システム・モニターのデータ編成

システム・モニターは情報を収集し保管します。情報はスナップショット・モニターおよび一部のイベント・モニターへのインターフェースを使用してアクセスできます。 データベース・システム・モニターは、収集した情報を、モニター・エレメント と呼ばれるエンティティーに保管します (モニター・エレメントは、以前にはデータ・エレメントと呼ばれていました)。

各モニター・エレメントは、データベース・システムの状態の特定のある局面に関する情報を保管します。さらに、モニター・エレメントは固有の名前によって識別され、特定のタイプの情報を保管します。

データを保管するためのシステム・モニターで使用可能なエレメント・タイプは、以下のとおりです。

カウンター
あるアクティビティーが起こった回数を数えます。モニター中は、カウンター値は増加します。大部分のカウンター・エレメントはリセットできます。

ゲージ ある項目の現行値を示します。ゲージ値は、データベース・アクティビティーによって上下します (例えば、保持されているロックの数)。ゲージ・エレメントはリセットできません。

水準点 水準点は、モニターを開始してからエレメントが到達した最高 (最大) または最低 (最小) 値を示します。水準点エレメントはリセットできません。

情報 モニター・アクティビティーの参照タイプの詳細を提供します。これには、パーティション名、別名、およびパス詳細などが含まれます。情報エレメントはリセットできません。

タイム・スタンプ
1970 年の 1 月 1 日以降の経過した秒およびマイクロ秒数を提供することによって、アクティビティーが起こった日時を示します。スナップショット・モニターおよびイベント・モニターの場合、タイム・スタンプ・エレメントの収集は「タイム・スタンプ」モニター・スイッチによって制御されます。このスイッチはデフォルトでは ON ですが、データベース・インスタンスでの CPU 使用率が 100% に近づいた場合には、パフォーマンス上の理由でそれを OFF にする必要があります。タイム・スタンプ・エレメントはリセットできません。

タイム・スタンプ・エレメントの値 0 は、「利用不可」を意味します。このデータをインポートしようとすると、この値によって範囲外エラー (SQL0181) が生成されます。このエラーを回避するには、データをエクスポートする前に、値を任意の有効なタイム・スタンプ値に更新します。

時間 アクティビティーで使用された秒数およびマイクロ秒数を戻します。スナップショット・モニターおよびイベント・モニターの場合、大半時間エレメントの収集は「タイム」モニター・スイッチによって制御されます。このスイッチはデフォルトでは ON ですが、データベース・インスタンスでの CPU 使用率が 100% に近づいた場合には、パフォーマンス上の理由でそれを OFF にする必要があります。時間エレメントのいくつかはリセットできます。

モニター・エレメントは、1 つ以上の論理データ・グループのデータを収集します。論理データ・グループは、データベース・アクティビティーの特定の範囲のデータベース・システム・モニター情報を収集するモニター・エレメントの集合です。モニター・エレメントは、それが提供する情報のレベルに基づいて論理データ・グループにソートされます。例えば、スナップショット・モニター中に、ソート合計時間モニター・エレメントがデータベース (dbase)、アプリケーション (appl)、およびステートメント (stmt) 情報を戻します。したがって、このモニター・エレメントは括弧内に示した論理データ・グループに属します。

多くのモニター・エレメントは、スナップショット・モニターとイベント・モニターの両方によって使用されますが、それらはそれぞれ別個のセットの論理データ・グループを使用します。これは、スナップショットをキャプチャーできるデータベース・アクティビティーの範囲が、イベント・データを収集できるものの範囲とは異なるからです。実際、スナップショット・モニターからアクセス可能なモニタ

ー・エレメントの全体の内容は、イベント・モニターからアクセス可能なモニター・エレメントの内容とは異なっています。

カウンターの状況および可視性

システム・モニターによって収集されるモニター・エレメントには、いくつかの累積カウンターが含まれています。カウンターは、データベースまたはデータベース・マネージャーの操作時 (例えば、アプリケーションがトランザクションをコミットするたび) に累積されます。

カウンターが初期設定されるのは、その当該オブジェクトがアクティブになった時点です。例えば、データベース用に読み取られるバッファー・プール・ページ数 (基本モニター・エレメント) は、データベースが活動化されるとゼロに設定されます。

システム・モニターが収集できるカウンターには、モニター・スイッチによって制御されているものがあります。 あるモニター・スイッチが OFF の場合には、その制御下にあるモニター・エレメントはデータを収集しません。モニター・スイッチが ON になると、関連したすべてのカウンターがゼロにリセットされます。

イベント・モニターによって戻されるカウンターは、イベント・モニターを活動化するとゼロにリセットされます。

イベント・モニターのカウントは、次の 1 つの開始点以後のカウントを表します。

- データベース、表スペース、および表に対する、イベント・モニターの始動。
- 既存の接続に対する、イベント・モニターの始動。
- モニターが開始された後で確立された、アプリケーションの接続。
- モニターが開始された後の、次のトランザクション (作業単位) またはステートメントの開始。
- モニターが開始された後のデッドロックの発生。

イベント・モニターと、モニター・アプリケーション (スナップショット・モニター API を使用するアプリケーション) には、システム・モニター・データに関する独自の論理ビューがあります。したがって、カウンターのリセットや初期設定が行われる際には、そのリセットや初期設定を行うイベント・モニターかアプリケーションだけが関係することになります。イベント・モニター・カウンターをリセットするには、イベント・モニターをいったん OFF にしてから ON にしなければなりません。スナップショットをとるアプリケーションの場合、 RESET MONITOR コマンドを使用するといつでもカウンターのビューをリセットできます。

ステートメントが開始された後でステートメントのイベント・モニターを開始すると、モニターは次の SQL ステートメントの開始時に情報の収集を開始します。その結果、モニターの開始時にデータベース・マネージャーが実行していたステートメントについての情報は、イベント・モニターは戻しません。これはトランザクション情報についても同様です。

システム・モニター出力 : 自己記述型データ・ストリーム

システム・モニター・データを画面に表示したり、SQL 表に保管したりすることのほかに、それを処理するクライアント・アプリケーションを開発することができま

す。システム・モニターは、スナップショット・モニターとイベント・モニターの両方について、自己記述型データ・ストリームを介してモニター・データを戻します。

スナップショット・モニター・アプリケーションでは、スナップショット API を呼び出してスナップショットをキャプチャーした後、そのデータ・ストリームを直接処理することができます。

イベント・モニター・データの処理は、これとは異なり、データベース・イベントが発生するたびにイベント・データがアプリケーションに送信されます。パイプ・イベント・モニターの場合は、アプリケーションはイベント・データの到着を待機し、到着時にそれを処理します。ファイル・イベント・モニターの場合は、アプリケーションはイベント・ファイルを解析するため、イベント・レコードをバッチで処理します。

この自己記述型データ・ストリームによって、戻りデータを、一度に 1 つのエレメントずつ解析することができます。このことは、特定のアプリケーションや特定のデータベース状態に関する情報の検索など、モニターに関連した数多くの可能性を実現させるものとなります。

戻されるモニター・データは以下の形式になります。

size モニター・エレメントまたは論理データ・グループに保管されているデータのサイズ (バイト)。論理データ・グループの場合、これは論理グループ内の全データのサイズです。例として、データベース論理グループ (*db*) には、個々のモニター・エレメント (*total_log_used* など) やロールフォワード情報 (*rollforward*) のようなほかの論理データ・グループが含まれます。これには、'size'、'type'、および 'element' 情報に取られるサイズは組み込まれません。

type データに保管されたエレメントのタイプ (例えば、可変長ストリングまたは符号付き 32 ビット数値)。 *header* のエレメント・タイプは、エレメントの論理データ・グループを示します。

element id
モニターによってキャプチャーされたモニター・エレメントの ID。論理データ・グループの場合、これはグループの ID です (例えば、 *collected*、*dbase*、または *event_db*)。

データ あるモニター・エレメントに対して、モニターによって収集された値。論理データ・グループの場合、データはそれに属するモニター・エレメントから成っています。

モニター・エレメントのすべてのタイム・スタンプは、 2 つの符号なし 4 バイト・モニター・エレメント (秒およびマイクロ秒) で戻されます。これらは GMT (グリニッジ標準時) で 1970 年 1 月 1 日以降の秒数で表されます。

モニター・エレメント内のストリングのサイズ・エレメントは、ストリング・エレメントの実際のデータ・サイズを表します。このサイズには、NULL 終止符は入っていません。ストリングが NULL で終了することはないからです。

モニター・データのメモリー要件

モニター・データに必要なメモリーは、モニター・ヒープから割り当てられます。モニター・ヒープのサイズを制御するには **mon_heap_sz** データベース構成パラメーターを使用します。 このパラメーターのデフォルト値は AUTOMATIC であるため、instance_memory 限度に達するまで、モニター・ヒープが必要に応じて増加します。

手動で **mon_heap_sz** パラメーターを構成する場合には、次の要因を考慮に入れてください。

- モニター・アプリケーションの数
- イベント・モニターの数と性質
- 設定されたモニター・スイッチ
- データベース・アクティビティーのレベル

モニター・コマンドが SQLCODE -973 で失敗するときは、**mon_heap_sz** パラメーターの値を大きくすることを検討してください。

次の公式を使うと、モニター・ヒープに必要なページの概数を求めることができます。

```
(アプリケーションが使用するメモリー              +
 イベント・モニターが使用するメモリー            +
 モニター・アプリケーションが使用するメモリー    +
 ゲートウェイ・アプリケーションが使用するメモリー)      / 4096
```

アプリケーションごとに使用するメモリー

- STATEMENT スイッチが OFF の場合はゼロ。
- STATEMENT スイッチが ON の場合:
  - 同時に実行されるステートメントごとに 400 バイトを追加する。 (つまりアプリケーションが持つ可能性があるオープン・カーソルの数) これはアプリケーションが実行するステートメントの累計ではありません。
  - パーティション・データベースの場合は、ステートメントごとに次を追加する。
    - 200 バイト * (サブセクションの平均数)
- アプリケーションが sqleseti() 情報を発行する場合は、ユーザー ID、アプリケーション名、ワークステーション名、およびアカウンティング・ストリングのサイズを追加する。

イベント・モニターごとに使用するメモリー

イベント・モニターのタイプが ACTIVITIES の場合:

- 3500 バイト
- イベント・モニターのタイプが TABLES の場合、36K * (CPU のコア数 + 1) を追加する。
- イベント・モニターのタイプが FILE または PIPE の場合、2K * (CPU のコア数 + 1) を追加する。

容量が大きくなることが予想される場合には、イベント・レコード用に 250 MB を追加してください。 あるいは、予想される作業量に合わせて分数を追加してください。

イベント・モニターのタイプが LOCKING または UOW の場合:

- 3500 バイト
- 3K * (CPU のコア数 + 1)

容量が大きくなることが予想される場合には、イベント・レコード用に 250 MB を追加してください。 あるいは、予想される作業量に合わせて分数を追加してください。

イベント・モニターのタイプが
DATABASE、TABLES、TABLESPACES、BUFFERPOOLS、CONNECTIONS、または DEADLOCK の場合:

- 4100 バイト
- 2 * バッファー・サイズ
- イベント・モニターがファイルに書き込まれる場合は、550 バイトを追加する。
- イベント・モニターのタイプが「データベース」の場合:
  - 6000 バイトを追加
  - ステートメント・キャッシュ内のステートメントごとに 100 バイトを追加
- イベント・モニターのタイプが「表」の場合:
  - 1500 バイトを追加
  - アクセスされる表ごとに 70 バイトを追加
- イベント・モニターのタイプが「表スペース」の場合:
  - 450 バイトを追加
  - 表スペースごとに 350 バイトを追加
- イベント・モニターのタイプが「バッファー・プール」の場合:
  - 450 バイトを追加
  - バッファー・プールごとに 340 バイトを追加
- イベント・モニターのタイプが「接続」の場合:
  - 1500 バイトを追加
  - 接続されるアプリケーションごとに:
    - 750 バイトを追加
  - 565 ページの『アプリケーションごとに使用するメモリー』からの値を追加することを忘れない。
- イベント・モニターのタイプが DEADLOCK の場合:
  - および WITH DETAILS HISTORY が実行中の場合:
    - X*475 バイトに、実行中であると予想される同時アプリケーションの最大数を掛けたものを追加。ここで、X はアプリケーションの作業単位内で予想されるステートメントの最大数。
  - および WITH DETAILS HISTORY VALUES が実行中の場合:

- X*Y バイトに、実行中であると予想される同時アプリケーションの最大数を掛けたものを追加。ここで、Y は SQL ステートメントに結び付けられているパラメーター値の予想される最大サイズ。

モニター・アプリケーションごとに使用するメモリー

- 250 バイト
- リセットされるデータベースごとに:
  - 350 バイト
  - リモート・データベースごとに 200 バイトを追加
  - SORT スイッチが ON の場合は 25 バイトを追加
  - LOCK スイッチが ON の場合は 25 バイトを追加
  - TABLE スイッチが ON の場合:
    - 600 バイトを追加
    - アクセスされる表ごとに 75 バイトを追加
  - BUFFERPOOL スイッチが ON の場合:
    - 300 バイトを追加
    - アクセスされる表スペースごとに 250 バイトを追加
    - アクセスされるバッファー・プールごとに 250 バイトを追加
  - STATEMENT スイッチが ON の場合:
    - 2100 バイトを追加
    - ステートメントごとに 100 バイトを追加
  - データベースに接続されるアプリケーションごとに:
    - 600 バイトを追加
    - アプリケーションの接続先のリモート・データベースごとに 200 バイトを追加
    - SORT スイッチが ON の場合は 25 バイトを追加
    - LOCK スイッチが ON の場合は 25 バイトを追加
    - BUFFERPOOL スイッチが ON の場合は 250 バイトを追加
- リセットされる DCS データベースごとに:
  - そのデータベース用に 200 バイトを追加
  - そのデータベースに接続されるアプリケーションごとに 200 バイトを追加
  - STATEMENT スイッチが ON で、伝送レベルのデータがリセットされる場合:
    - データベースごとに、伝送レベルごとに 200 バイトを追加
    - アプリケーションごとに、伝送レベルごとに 200 バイトを追加

ゲートウェイ・アプリケーションが使用するメモリー

- ホスト・データベースごとに 250 バイト (すべてのスイッチが OFF の場合でも)
- アプリケーションごとに 400 バイト (すべてのスイッチが OFF の場合でも)
- STATEMENT スイッチが ON の場合:

– アプリケーションごとに、同時に実行されるステートメント (つまりアプリケーションが持つ可能性があるオープン・カーソルの数) ごとに 200 バイトを追加する。これはアプリケーションが実行するステートメントの累計ではありません。
– 伝送レベルのデータは次のように計算する。
  - データベースごとに、伝送レベルごとに 200 バイトを追加
  - アプリケーションごとに、伝送レベルごとに 200 バイトを追加

- UOW スイッチが ON の場合:
  – アプリケーションごとに 50 バイトを追加
- TMDB (SYNCPOINT TWOPHASE アクティビティー用) を使用するアプリケーションごとに:
  – 20 バイトにさらに XID 自体のサイズを追加
- sqleseti を発行してクライアント名、アプリケーション名、ワークステーション、またはアカウンティングを設定するアプリケーションの場合:
  – 800 バイトにさらにアカウンティング・ストリング自体のサイズを追加

バッファー・プール・アクティビティーのモニター

データベース・サーバーは、バッファー・プールのすべてのデータについて読み取りと更新を行います。アプリケーションの要求に応じて、データはディスクからバッファー・プールにコピーされます。

ページは、次のようにバッファー・プール内に置かれます。

- エージェントによる。同期入出力です。
- 入出力サーバー (プリフェッチャー) による。非同期入出力です。

ページは、次のようにバッファー・プールからディスクに書き込まれます。

- エージェントにより、同期で。
- ページ・クリーナーにより、非同期で。

サーバーが 1 つのデータ・ページを読み取る必要があり、そのページがすでにバッファー・プール内にある場合は、ページをディスクから読み取るときよりも高速にそのページにアクセスできます。バッファー・プール内でできるだけたくさんのページをヒットすることが必要になります。ディスク I/O の回避はデータベースのパフォーマンスにおいて重要な要因であるため、バッファー・プールを適切に構成することが、パフォーマンスの調整について最も重要な考慮事項の 1 つになります。

バッファー・プールのヒット率は、データベース・マネージャーがページ要求を処理するときに、そのページがすでにバッファー・プール内に存在したためにディスクからページをロードする必要がなかった場合のパーセンテージを示します。バッファー・プール・ヒット率が高いほど、ディスク入出力の頻度は低くなります。

注: 以下で取り上げられているのは、DB2 pureScale 環境以外の環境におけるバッファー・プールについての説明です。DB2 pureScale 環境におけるバッファー・プールの動作は異なります。詳しくは、「*データベースのモニタリング ガイドおよびリファレンス*」の『DB2 pureScale 環境におけるバッファー・プールのモニター』を参照してください。

例えば、全体のバッファー・プール・ヒット率は、次のように計算できます。

```
     ((pool_data_lbp_pages_found + pool_index_lbp_pages_found
 + pool_xda_lbp_pages_found
-
pool_async_data_lbp_pages_found - pool_async_index_lbp_pages_found
- pool_async_xda_lbp_pages_found)
/ (pool_data_l_reads
+ pool_index_l_reads + pool_xda_l_reads + pool_temp_data_l_reads +
pool_temp_xda_l_reads + pool_temp_index_l_reads)) × 100
```

この計算には、バッファー・プールによってキャッシュされているすべてのページ (索引とデータ) が含まれます。

BP_HITRATIO 管理ビューを、バッファー・プールのヒット率をモニターする便利な方法として使用することもできます。

大規模なデータベースでは、バッファー・プールを大きくしても、バッファー・プール・ヒット率の効果が得られないことがあります。データ・ページ数が大きすぎるために、そのサイズを大きくしてもヒットの統計的な確率が高くならないことがあります。代わりに、索引のバッファー・プール・ヒット率を調整すると、よい結果を得られることがあります。これには 2 つの方法があります。

1. データと索引を 2 つの異なるバッファー・プールに分割し、それぞれを個別に調整します。
2. 1 つのバッファー・プールを使用し、索引のヒット率がこれ以上高くならないというところまでそのバッファー・プールのサイズを大きくします。索引のバッファー・プール・ヒット率は、次のように計算できます。

   ```
   ((pool_index_lbp_pages_found
   - pool_async_index_lbp_pages_found - pool_temp_index_l_reads)
   / (pool_index_l_reads) × 100
   ```

最初の方法のほうが効果が上がることが多いのですが、索引とデータを別の表スペースに置く必要があるので、既存データベースには利用できないことがあります。さらに、1 つではなく、2 つのバッファー・プールを調整する必要があるので、特にメモリーに制約があるときなどは困難な作業となります。

さらに、プリフェッチャーのヒット率に対する影響を考慮する必要があります。プリフェッチャーは、アプリケーションの必要性を予想して、データ・ページをバッファー・プール内に読み取ります (非同期)。多くの場合、これらのページは必要になる直前に読み込まれます (必要な場合)。ただし、プリフェッチャーは、使用されないページをバッファー・プール内に読み込むことで不要な入出力を行うこともあります。例えば、あるアプリケーションが表の読み取りを開始するとします。このことが検出されると、プリフェッチが開始しますが、アプリケーションはアプリケーション・バッファーを充てんして読み取りを停止します。この間に、その他の多数のページがプリフェッチされます。結局使用されることのないページについて入出力が行われ、バッファー・プールの一部がこうしたページで占められることになってしまいます。

ページ・クリーナーは、バッファー・プールをモニターし、ディスクにページを非同期で書き込みます。これには次の目的があります。

- エージェントがバッファー・プール内でフリー・ページを必ず見つけられるようにする。エージェントがバッファー・プール内でフリー・ページを見つけられな

い場合は、エージェント自身がページをクリーニングしなければならないため、関連アプリケーションに対してよい応答を返せないことになります。

- システムがクラッシュした場合に、データベースのリカバリーを迅速に行う。ディスクに書き込まれたページ数が多いほど、データベースのリカバリーで処理が必要となるログ・ファイル・レコードの数は少なくなります。

ダーティー・ページはディスクに書き出されますが、バッファー・プールに新しいページを読み取るためのスペースが必要な場合を除いて、このページはバッファー・プールからすぐには除去されません。

注: バッファー・プールに関する情報は、通常は表スペース・レベルで収集されますが、データベース・システム・モニターの機能により、この情報はバッファー・プールおよびデータベースのレベルにまでまとめることができます。実行する分析の種類にもよりますが、いずれかのレベルまたはすべてのレベルでこのデータを調査する必要があります。

データベース・システム・モニター・インターフェース

モニター・データの収集を制御し、結果を検討するために使用できるシステム・モニター・ツールが多数あります。

モニター・タスク	API
スナップショットのキャプチャー	db2GetSnapshot
自己記述型データ・ストリームの変換	db2ConvMonStream
データベース・システム・モニター・スイッチの表示	db2MonitorSwitches
スナップショットのサイズ見積もり	db2GetSnapshotSize
モニター・スイッチの取得/更新	db2MonitorSwitches
モニター・カウンターのリセット	db2ResetMonitor
データベース・システム・モニター・スイッチの更新	db2MonitorSwitches

モニター・タスク	CLP コマンド
スナップショットのキャプチャー	GET SNAPSHOT
データベース・マネージャー・モニター・スイッチの表示	GET DATABASE MANAGER MONITOR SWITCHES
モニター・アプリケーションのモニター・スイッチの表示	GET MONITOR SWITCHES
イベント・モニター・トレースのフォーマット	db2evmon
表書き込み CREATE EVENT MONITOR ステートメントに関する SQL 例の生成	db2evtbl
アクティブなデータベースのリスト	LIST ACTIVE DATABASES
データベースに接続されたアプリケーションのリスト	LIST APPLICATIONS
DCS アプリケーションのリスト	LIST DCS APPLICATIONS
モニター・カウンターのリセット	RESET MONITOR

モニター・タスク	CLP コマンド
データベース・システム・モニター・スイッチの更新	UPDATE MONITOR SWITCHES

モニター・タスク	SQL ステートメント
イベント・モニターの活動化	SET EVENT MONITOR STATE
イベント・モニターの作成	CREATE EVENT MONITOR
イベント・モニターの非アクティブ化	SET EVENT MONITOR STATE
イベント・モニターの削除	DROP
イベント・モニター値の書き込み	FLUSH EVENT MONITOR

モニター・タスク	SQL 関数
イベント・モニターの状態の判別	EVENT_MON_STATE スカラー関数
データベース・マネージャー・レベルのスナップショットの取得	SNAPDBM 管理ビューおよび SNAP_GET_DBM 表関数
データベース・マネージャー・レベルでの、現行のモニター・スイッチ設定値の取得	SNAPSWITCHES 管理ビューおよび SNAP_GET_SWITCHES 表関数
高速コミュニケーション・マネージャーのスナップショットの取得	SNAPFCM 管理ビューおよび SNAP_GET_FCM 表関数
指定されたパーティションについての、高速コミュニケーション・マネージャーのスナップショットの取得	SNAPFCM_PART 管理ビューおよび SNAP_GET_FCM_PART 表関数
データベース・レベルのスナップショットの取得	SNAPDB 管理ビューおよび SNAP_GET_DB 表関数
アプリケーション・レベルのスナップショットの取得	SNAPAPPL 管理ビューおよび SNAP_GET_APPL 表関数
アプリケーション・レベルのスナップショットの取得	SNAPAPPL_INFO 管理ビューおよび SNAP_GET_APPL_INFO 表関数
ロック待機情報に関する、アプリケーション・レベルのスナップショットの取得	SNAPLOCKWAIT 管理ビューおよび SNAP_GET_LOCKWAIT 表関数
ステートメント情報に関する、アプリケーション・レベルのスナップショットの取得	SNAPSTMT 管理ビューおよび SNAP_GET_STMT 表関数
エージェント情報に関する、アプリケーション・レベルのスナップショットの取得	SNAPAGENT 管理ビューおよび SNAP_GET_AGENT 表関数
サブセクション情報に関する、アプリケーション・レベルのスナップショットの取得	SNAPSUBSECTION 管理ビューおよび SNAP_GET_SUBSECTION 表関数
バッファー・プール・レベルのスナップショットの取得	SNAPBP 管理ビューおよび SNAP_GET_BP 表関数
表スペース・レベルのスナップショットの取得	SNAPTBSP 管理ビューおよび SNAP_GET_TBSP 表関数
構成情報に関する、表スペース・レベルのスナップショットの取得	SNAPTBSP_PART 管理ビューおよび SNAP_GET_TBSP_PART 表関数
コンテナー情報に関する、表スペース・レベルのスナップショットの取得	SNAPCONTAINER 管理ビューおよび SNAP_GET_CONTAINER 表関数
静止プログラム情報に関する、表スペース・レベルのスナップショットの取得	SNAPTBSP_QUIESCER 管理ビューおよび SNAP_GET_TBSP_QUIESCER 表関数

モニター・タスク	SQL 関数
表スペース・マップの範囲に関する、表スペース・レベルのスナップショットの取得	SNAPTBSP_RANGE 管理ビューおよび SNAP_GET_TBSP_RANGE 表関数
表レベルのスナップショットの取得	SNAPTAB 管理ビューおよび SNAP_GET_TAB 表関数
ロック・レベルのスナップショットの取得	SNAPLOCK 管理ビューおよび SNAP_GET_LOCK 表関数
SQL ステートメントのキャッシュ情報のスナップショットの取得	SNAPDYN_SQL 管理ビューおよび SNAP_GET_DYN_SQL 表関数

データベース・オブジェクトが最後に使用された日付の特定

オブジェクトが最後に使用された日付は、最終参照日 (または最終使用日) として示されます。最終参照日は、索引、パッケージ、表、表データ・パーティション、およびマテリアライズ照会表 (MQT) について取得できます。

最終参照日を使用して、長期間使用されていないオブジェクトで、削除する候補として検討しても良いオブジェクトを特定できます。

最終参照日は、オブジェクトごとに対応するカタログ表の LASTUSED 列に格納されており、その表に対するカタログ・ビューからアクセス可能です。カタログ内の使用情報は、データベース・カタログ・パーティションで実行される、**db2lused** (LASTUSED デーモン) というエンジン・ディスパッチ可能単位 (EDU) によって更新されます。 15 分ごとに、LASTUSED デーモンはすべてのパーティションのすべてのオブジェクトに関する使用情報を収集し、対応するカタログ表内の LASTUSED 列を更新して、ディスクに情報を書き込みます。 1 つのオブジェクトに関するカタログ項目は最大でも 1 日に 1 回更新されます。つまり、24 時間のインターバルが経過するまで、同じオブジェクトは再検査されません。データベース・サーバーのパフォーマンス上の影響を最小化するために 15 分のインターバルが選ばれました。ユーザーはこれを構成できません。最後の参照日の更新は非同期的に実行されるため、オブジェクト・アクセスはカタログ内に直ちには記録されません。

注: カタログ表内の対応する行がロックされている場合、次回の 15 分の収集インターバルまで、使用情報の更新が遅延する可能性があります。さらに、データベースが非アクティブ化されている場合、非アクティブ化の前に LASTUSED デーモンによって収集されなかった使用情報 (例えば、デーモンによる最後のポーリング以降に初めてアクセスされたオブジェクト) をディスクに書き込むことはできません。このフィーチャーを適切に動作させるには、データベースを明示的に活動化してください。

オブジェクトが長期間 (例えば数カ月) 使用されていない場合に、最終参照日は有用です。以下のように、最終参照日は有用です。

- 表および表データ・パーティション: 不使用スペースを再利用する機会を特定するのに役立ちます。
- 索引: 不使用スペースを再利用する機会を特定、および不必要な挿入や保守を回避するのに役立ち、また検討すべき索引選択の回数が減少することによってコンパイル時間が改善されます。
- パッケージ: 解放することが可能な不使用パッケージ・バージョンの検出に役立ちます。

- MQT: 不使用 MQT の検出、不使用スペースの再利用、または MQT が使用されない原因の調査や理解に役立ちます。

以下の例では、最終参照日が有用となる特定のシナリオについていくつか説明します。

- スペースおよび保守時間を節約するための機会を特定するために、SYSCAT.INDEXES カタログ・ビューの LASTUSED 列をチェックすることによって、索引の最終使用に関する情報を毎年調査します。最後の年に索引が使用されていなかった場合、その索引はドロップする候補と見なすことができます。索引のドロップが必要でない事情も考えられるため、索引をドロップするかどうかは、ユーザーが最終的に判断します。例えば、緊急時のみアクセスされる表、または頻繁なアクセスはなくても素早いアクセスが重要である表が考えられます。あるいは表の索引がユニークである場合、その索引が明示的に使用されたことはなくても、ユニーク制約の適用のために、使用しなければならない可能性があります。最終使用日の情報は、索引の削除を決断する際に役立ちます。
- 企業の社内アプリケーションの中には、データベースにデプロイされて数カ月または数年の期間が経過した後、他のアプリケーションに置き換わったか、もはや使用されなくなったものがあります。リタイヤしたアプリケーションは、スペースを節約する機会として特定されました。最終使用日の情報を使用して、もう使用されていないデータベース・オブジェクトで、アプリケーションがリタイヤした後にクリーンアップされなかったものを特定することができます。例えば、これらのデータベース・オブジェクトは、GUI にデータを追加するために使用される値を保管する表などが考えられます。このような表の最終使用日は、SYSCAT.TABLES カタログ・ビューの LASTUSED 列に表示されます。この日付は、スペースを再利用するために削除可能な表オブジェクトを調査するための開始点として使用できます。

特定のデータベース・オブジェクト、とりわけ結果的に更新を生じるデータベース・オブジェクト操作に関する、カタログ・ビューの LASTUSED 列について詳しくは、以下のトピックを参照してください。

- SYSCAT.DATAPARTITIONS カタログ・ビュー
- SYSCAT.INDEXES カタログ・ビュー
- SYSCAT.PACKAGES カタログ・ビュー
- SYSCAT.TABLES カタログ・ビュー

第 5 章 非推奨のモニター・ツール

ヘルス・モニターおよび Windows Management Instrumentation (WMI) は非推奨になりました。将来のリリースで除去される可能性があります。

IBM InfoSphere Optim ツールの使用を始めてください。IBM InfoSphere Optim Performance Manager は、データベースの典型的なパフォーマンス上の問題を特定して分析するために使用できる Web インターフェースを提供します。データベースの正常性の要約を表示して調べることもできます。詳細については、Optim Performance Manager を使用したモニター (http://publib.boulder.ibm.com/infocenter/idm/docv3/topic/com.ibm.datatools.perfmgmt.monitor.doc/p_monitor.html) を参照してください。

ヘルス・モニターの概要

ヘルス・モニターとは、サーバー・サイドのツールの一種で、インスタンスとアクティブ・データベースの正常性を定期的にモニターして、例外による管理機能を追加します。ヘルス・モニターは、潜在的なシステムの正常性の問題についてデータベース管理者 (DBA) にアラートを出すこともできます。

ヘルス・モニターは、ハードウェア障害や、受け入れがたいシステム・パフォーマンスまたは機能の低下を引き起こしかねない問題を事前に検出します。ヘルス・モニターには事前の対策を講じる性質があるので、ユーザーは、システム・パフォーマンスに影響する問題に発展する前にその問題に取り組むことができます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ヘルス・モニターは、ヘルス・インディケーターを使用してシステムの状態を検査し、アラートを発行する必要があるかどうかを判別します。アラートに応じて、事前構成済みのアクションが取られます。さらにヘルス・モニターは、管理通知ログにアラートを記録し、電子メールまたはページャーで通知を送信することもできます。この例外による管理モデルにより、アクティブ・モニターを必要とせずに、潜在的なシステムの正常性に関する問題に対するアラートを生成できるので、貴重な DBA リソースを解放できます。

ヘルス・モニターは、全体のパフォーマンスに与える影響を最小限に抑えつつ、システムの正常性に関する情報を定期的に収集します。情報を収集するのに、スナップショット・モニター・スイッチを ON にしません。

ヘルス・インディケーター

ヘルス・モニターは、ヘルス・インディケーターを使用して、データベース・マネージャーやデータベースのパフォーマンスの特定の性質の正常性を評価します。ヘ

ルス・インディケーターは、表スペースなど特定のクラスのデータベース・オブジェクトのある性質の正常性を測定します。正常性を判別するため、測定値に対してある基準が適用されます。ここで適用される基準は、ヘルス・インディケーターのタイプに従属します。この基準に基づいて正常稼働ではないと判別されると、アラートが生成されます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ヘルス・モニターによって、以下の 3 つのタイプのヘルス・インディケーターが戻されます。

- **しきい値ベースの**インディケーターは、オブジェクトの動作の (連続的な範囲の値の) 統計を表すメジャーである。警告およびアラームしきい値は、正常、警告、およびアラーム範囲の境界つまりゾーンを定義します。しきい値ベースのヘルス・インディケーターには、正常、警告、およびアラームの 3 つの有効な状態があります。
- **状態ベースの**インディケーターは、データベース・オブジェクトまたはリソースの操作が正常かどうかを定義するオブジェクトの、 2 つ以上の異なる状態の限定集合を表すメジャーである。これらの状態の 1 つが通常で、その他の状態はすべて通常ではないと見なされます。状態ベースのヘルス・インディケーターには、通常およびアテンションの 2 つの有効な状態があります。
- **コレクション状態ベースの**インディケーターは、データベース中の 1 つ以上のオブジェクトのコレクション状態を表すデータベース・レベルのメジャーである。コレクション中のオブジェクトごとにデータが取り込まれ、これらのオブジェクトの間で最も重大な条件が集約状態として表されます。コレクション中の 1 つ以上のオブジェクトがアラートを必要とする状態である場合は、ヘルス・インディケーターはアテンションを表示します。コレクション状態ベースのヘルス・インディケーターには、通常およびアテンションの 2 つの有効な状態があります。

ヘルス・インディケーターは、インスタンス、データベース、表スペース、および表スペース・コンテナー・レベルです。

ヘルス・モニター情報には、 CLP または API を使用してアクセスできます。これらのツールを使用してヘルス・インディケーターの構成も行えます。

アラートは、正常の状態から正ではない状態への変化、または定義されたしきい値の境界に基づくヘルス・インディケーター値の警告またはアラームのゾーンへの変化に応答して生成されます。アラートには、アテンション、警告、およびアラームの 3 つがあります。

- 特定の状態を測定するヘルス・インディケーターの場合、通常でない状態が登録されると、アテンション・アラートが発行される。
- 値の連続範囲を計測するヘルス・インディケーターの場合は、正常、警告、およびアラームの各状態の境界やゾーンは、しきい値によって定義される。例えば、

値がアラーム・ゾーンとして定義されている値のしきい値範囲に入ると、アラームのアラートが発行されて、問題に対して即時に対処が必要であることを示します。

ヘルス・モニターは、特定のヘルス・インディケーターの特定のアラート条件が最初に現れたときにのみ通知を送信し、アクションを実行します。ヘルス・インディケーターが特定のアラート条件のまま留まる場合、追加の通知は送信されず、追加のアクションも実行されません。ヘルス・インディケーターがアラート条件を変更するか、または通常の状態に戻って再びアラート条件に入る場合、新たに通知が送信され、アクションが実行されます。

以下の表は、様々なリフレッシュ・インターバルにおけるヘルス・インディケーターと、ヘルス・インディケーターの状態に対するヘルス・モニターの応答の例を示しています。この例では、デフォルトの警告しきい値として 80 %、アラームしきい値として 90 % を使用しています。

表 122. 様々なリフレッシュ・インターバルにおけるヘルス・インディケーターの状態

リフレッシュ・インターバル	ts.ts_util (表スペースの使用率) ヘルス・インディケーターの値	ts.ts_util ヘルス・インディケーターの状態	ヘルス・モニターの応答
1	80	警告	警告の通知が送信され、警告アラート条件のアクションが実行される
2	81	警告	通知は送信されず、アクションは実行されない
3	75	正常	通知は送信されず、アクションは実行されない
4	85	警告	警告の通知が送信され、警告アラート条件のアクションが実行される
5	90	アラーム	アラームの通知が送信され、アラーム条件のアクションが実行される

ヘルス・インディケーターのプロセスのサイクル

次の図は、ヘルス・インディケーターの評価プロセスを図示しています。ステップの集合は、特定のヘルス・インディケーターのリフレッシュ・インターバルが経過するたびに実行されます。

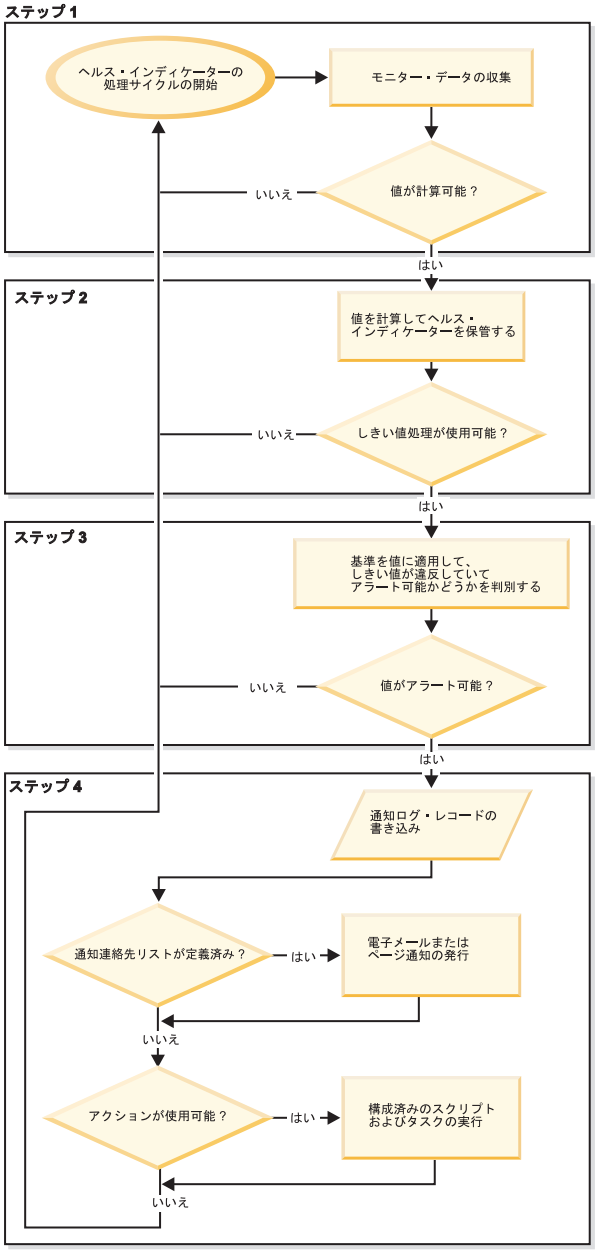


図 8. ヘルス・インディケーターのプロセスのサイクル

注:

1. NOTIFYLEVEL データベース・マネージャー構成パラメーターは、アラート通知が DB2 管理通知ログと定義済みの連絡先に送信するかどうかを制御します。アラーム通知には重大度レベル 2 以上が必要です。警告およびアテンション・アラートを送信するには、重大度レベル 3 以上が必要です。

ヘルス・インディケーターの形式

ヘルス・インディケーターによって収集されるデータの説明。

ヘルス・インディケーターのドキュメンテーションは次の標準形式で記述されます。

ID ヘルス・インディケーターの名前。この ID は、CLP からの構成で使用されます。

ヘルス・モニター・レベル
ヘルス・モニターによってヘルス・インディケーターがキャプチャーされる際のレベル。

分類 ヘルス・インディケーターのカテゴリー。

タイプ ヘルス・インディケーターのタイプ。以下の 4 つのタイプがあります。

- 上限しきい値ベース。アラートを生成する進行状況は正常、警告、アラームです。
- 下限しきい値ベース
- 状態ベース。 1 つの状態が正常で、その他の状態はすべて正常ではありません。
- コレクション状態ベース。状態は、集合中のオブジェクトの状態の集約に基づいています。

単位 パーセンテージなどの、ヘルス・インディケーターで測定されるデータの単位。状態ベースやコレクション状態ベースのヘルス・インディケーターには該当しません。

ヘルス・インディケーターの要約

このリファレンスは、ご使用のデータベース上にヘルス・インディケーターを作成し、保守し、読み取るのに役立ちます。

次の表には、すべてのヘルス・インディケーターが、カテゴリー別にグループ化されてリストされています。

表 123. データベース自動ストレージ使用率のヘルス・インディケーター

Name (名前)	ID	追加情報
データベース自動ストレージ使用率	db.auto_storage_util	583 ページの『db.auto_storage_util データベース自動ストレージ使用率 : ヘルス・インディケーター』

表 124. 表スペース・ストレージのヘルス・インディケーター

Name (名前)	ID	追加情報
表スペース自動サイズ変更状態	ts.ts_auto_resize_status	584 ページの『ts.ts_auto_resize_status 表スペース自動サイズ変更状態 : ヘルス・インディケーター』
自動サイズ変更表スペース使用率	ts.ts_util_auto_resize	585 ページの『ts.ts_util_auto_resize 自動サイズ変更表スペース使用率 : ヘルス・インディケーター』
表スペース使用率	ts.ts_util	586 ページの『ts.ts_util 表スペース使用率』
表スペース・コンテナー使用率	tsc.tscont_util	587 ページの『tsc.tscont_util 表スペース・コンテナー使用率』
表スペース操作可能状態	ts.ts_op_status	588 ページの『ts.ts_op_status 表スペース操作可能状態 :』

表 124. 表スペース・ストレージのヘルス・インディケーター (続き)

Name (名前)	ID	追加情報
表スペース・コンテナー操作可能状態	tsc.tscont_op_status	588 ページの『tsc.tscont_op_status 表スペース・コンテナー操作可能状態』
表スペース自動サイズ変更状態	ts.ts_auto_resize_status	584 ページの『ts.ts_auto_resize_status 表スペース自動サイズ変更状態 : ヘルス・インディケーター』

表 125. ソートのヘルス・インディケーター

Name (名前)	ID	追加情報
専用ソート・メモリー使用率	db2.sort_privmem_util	589 ページの『db2.sort_privmem_util 専用ソート・メモリー使用率』
共有ソート・メモリー使用率	db.sort_shrmem_util	589 ページの『db.sort_shrmem_util 共有ソート・メモリー使用率』
オーバーフローしたソートのパーセンテージ	db.spilled_sorts	590 ページの『db.spilled_sorts オーバーフローしたソートのパーセンテージ』
長期共有ソート・メモリー使用率	db.max_sort_shrmem_util	591 ページの『db.max_sort_shrmem_util 長期共有ソート・メモリー使用率』

表 126. データベース・マネージャーのヘルス・インディケーター

Name (名前)	ID	追加情報
インスタンス操作可能状態	db2.db2_op_status	592 ページの『db2.db2_op_status インスタンス操作可能状態』
インスタンス最大重大度アラート状態	-	592 ページの『インスタンス最大重大度アラート状態 :』

表 127. データベースのヘルス・インディケーター

Name (名前)	ID	追加情報
データベース操作可能状態	db.db_op_status	593 ページの『db.db_op_status データベース操作可能状態 :』
データベース最大重大度アラート状態	-	593 ページの『データベース最大重大度アラート状態』

表 128. 保守のヘルス・インディケーター

Name (名前)	ID	追加情報
再編成の必要性	db.tb_reorg_req	594 ページの『db.tb_reorg_req 再編成の必要性』
統計収集の必要性ヘルス・インディケーター	db.tb_runstats_req	595 ページの『db.tb_runstats_req 統計収集の必要性』
データベース・バックアップの必要性	db.db_backup_req	595 ページの『db.db_backup_req データベース・バックアップの必要性 :』

表 129. 高可用性災害時リカバリーのヘルス・インディケーター

Name (名前)	ID	追加情報
HADR 操作可能状態ヘルス・インディケーター	db.hadr_op_status	596 ページの『db.hadr_op_status HADR 操作可能状態 :』
HADR ログ遅延ヘルス・インディケーター	db.hadr_delay	596 ページの『db.hadr_delay HADR ログ遅延』

表 130. ロギングのヘルス・インディケーター

Name (名前)	ID	追加情報
ログ使用率	db.log_util	597 ページの『db.log_util ログ使用率』
ログ・ファイル・システム使用率	db.log_fs_util	598 ページの『db.log_fs_util ログ・ファイル・システム使用率 :』

表 131. アプリケーション並行性のヘルス・インディケーター

Name (名前)	ID	追加情報
デッドロック率	db.deadlock_rate	598 ページの『db.deadlock_rate デッドロック率』
ロック・リスト使用率	db.locklist_util	599 ページの『db.locklist_util ロック・リスト使用率』
ロック・エスカレーション率	db.lock_escal_rate	600 ページの『db.lock_escal_rate ロック・エスカレーション率』
ロック待機中のアプリケーションのパーセンテージ	db.apps_waiting_locks	601 ページの『db.apps_waiting_locks ロック待機中のアプリケーションのパーセンテージ』

表 132. パッケージ・キャッシュ、カタログ・キャッシュ、ワークスペースのヘルス・インディケーター

Name (名前)	ID	追加情報
カタログ・キャッシュ・ヒット率	db.catcache_hitratio	602 ページの『db.catcache_hitratio カタログ・キャッシュ・ヒット率 :』
パッケージ・キャッシュ・ヒット率	db.pkgcache_hitratio	602 ページの『db.pkgcache_hitratio パッケージ・キャッシュ・ヒット率 :』
共有ワークスペース・ヒット率	db.shrworkspace_hitratio	603 ページの『db.shrworkspace_hitratio 共有ワークスペース・ヒット率』

表 133. メモリーのヘルス・インディケーター

Name (名前)	ID	追加情報
モニター・ヒープ使用率	db2.mon_heap_util	603 ページの『db2.mon_heap_util モニター・ヒープ使用率 :』
データベース・ヒープ使用率	db.db_heap_util	604 ページの『db.db_heap_util データベース・ヒープ使用率』

表 134. フェデレーテッドのヘルス・インディケーター

Name (名前)	ID	追加情報
ニックネームの状態	db.fed_nicknames_op_status	605 ページの『db.fed_nicknames_op_status ニックネームの状態』
データ・ソース・サーバーの状態	db.fed_servers_op_status	605 ページの『db.fed_servers_op_status データ・ソース・サーバーの状態』

表スペース・ストレージのヘルス・インディケーター:

DMS 表スペースのヘルス・インディケーター:

表スペースの特性に基づいて、DMS 表スペースに関係しているヘルス・インディケーターを判別できます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

この表は、表スペースの特性に基づき、どの表スペース・ヘルス・インディケーターが DMS 表スペースに関係しているかを説明します。

表 135. DMS 表スペースに関係する表スペース・ヘルス・インディケーター

表スペースの特性	定義されている表スペースの最大サイズ	定義されていない表スペースの最大サイズ
自動サイズ変更が使用可能 = Yes	ts.ts_util_auto_resize - 使用されている表スペースのパーセントを、ユーザーが定義した最大サイズと比べて追跡します。アラートは、表スペースがまもなく満杯になり、ユーザーによる介入が必要であることを示します。最大サイズが妥当な値に設定されている限り (つまり、最大サイズで指定されたスペース量が存在しているなら)、この構成に最も重要なヘルス・インディケーターです。 ts.ts_util - 現在割り振られている表スペース・ストレージの使用量を追跡します。表スペースが満杯になると表スペースはサイズを大きくしようとするため、アラートが出されても、問題を解決するためにユーザーによる介入を必要としないことがあります。 ts.ts_auto_resize_status - サイズ変更の試行が正常かどうかを追跡します。アラートは、表スペースがサイズ変更に失敗した (つまり、表スペースが満杯である) ことを示します。	ts.ts_util_auto_resize - 適用されません。表スペース・サイズの上限が指定されていません。 ts.ts_util - 現在割り振られている表スペース・ストレージの使用量を追跡します。表スペースはサイズを大きくしようとするため、アラートが出されても、問題を解決するためにユーザーによる介入を必要としないことがあります。 ts.ts_auto_resize_status - サイズ変更の試行が正常かどうかを追跡します。アラートは、表スペースがサイズ変更に失敗した (つまり、表スペースが満杯である) ことを示します。 **注:** DMS 表スペースが自動ストレージを使用して定義されており、最大サイズが指定されていない場合、db.auto_storage_util ヘルス・インディケーターにも注目する必要があります。このヘルス・インディケーターは、データベース・ストレージ・パスに関連したスペースの使用率を追跡します。このスペースがいっぱいになると、表スペースを大きくすることができません。その結果、表スペースの満杯状態になることがあります。
自動サイズ変更が使用可能 = No	有効な構成ではありません。表スペースの最大サイズは、自動サイズ変更が使用可能になっている表スペースにのみ有効です。	ts.ts_util_auto_resize - 適用されません。表スペースはサイズ変更を試行しません。 ts.ts_util - 現在割り振られている表スペース・ストレージの使用量を追跡します。アラートは表スペースの満杯状態を示しており、ユーザーによる即時介入が必要です。表スペースは自動的にサイズ変更を試行しません。 ts.ts_auto_resize_status - 適用されません。表スペースはサイズ変更を試行しません。

db.auto_storage_util データベース自動ストレージ使用率 : ヘルス・インディケーター:

定義されたデータベース・ストレージ・パスのストレージの使用量を示します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.auto_storage_util

ヘルス・モニター・レベル
データベース

分類 データベース

タイプ 上限しきい値ベース

単位 パーセンテージ

自動ストレージ表スペースが作成されると、データベース・ストレージ・パス上にそれらの表スペース用のコンテナーが自動的に割り振られます。データベース・ストレージ・パスが定義されているどのファイル・システム上にもスペースが残されていない場合、自動ストレージ表スペースはサイズを大きくすることができず、満杯になります。

インディケーターは、次の公式を使用して計算されます。

```
(db.auto_storage_used / db.auto_storage_total) * 100
```

ここで、

- *db.auto_storage_used* は、データベース・ストレージ・パスのリストで指定されているすべての物理ファイル・システムの使用中のスペースの合計です。
- *db.auto_storage_total* は、データベース・ストレージ・パスのリストで指定されているすべての物理ファイル・システムの全スペースの合計です。

データベース自動ストレージ・パス使用率は、データベース・ストレージ・パスのファイル・システム上で消費されたスペースのパーセントとして測定され、高いパーセントはこのインディケーターの最適関数を下回っていることを示します。

このヘルス・インディケーターで戻される「追加情報」行の「フルになるまでの時間」は、表スペースの最大サイズに達するまでの残りの時間を予測したものです。

使用上の注意
ストレージ・グループを使用する場合、このヘルス・インディケーターは、デフォルトのストレージ・グループ内の定義済みデータベース・ストレージ・パスのストレージのみの使用量を示します。

ts.ts_auto_resize_status 表スペース自動サイズ変更状態 : ヘルス・インディケーター :

このヘルス・インディケーターは、自動サイズ変更が使用可能になっている DMS 表スペースに対して表スペースのサイズ変更操作が正常に行われているかどうかを識別します。

自動サイズ変更が使用可能になっている DMS 表スペースがサイズを大きくできない場合、その表スペースは事実上満杯になっています。この状態は、表スペース・

コンテナーが定義されているファイル・システム上にフリー・スペースがないか、または表スペース・コンテナーの自動サイズ変更設定の結果として起こります。例えば、定義された最大サイズに達している可能性や、増加量の設定が大きすぎるために残りのフリー・スペースでは収まらない可能性があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID ts.ts_auto_resize_status

ヘルス・モニター・レベル
表スペース

分類 表スペース・ストレージ

タイプ 状態ベース

単位 適用外

ts.ts_util_auto_resize 自動サイズ変更表スペース使用率 : ヘルス・インディケーター :

このヘルス・インディケーターは、最大サイズが定義されている自動サイズ変更 DMS 表スペースごとのストレージの使用量を追跡します。最大サイズに達している場合、DMS 表スペースが満杯であると見なされます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID ts.ts_util_auto_resize

ヘルス・モニター・レベル
表スペース

分類 表スペース・ストレージ

タイプ 上限しきい値ベース

単位 パーセンテージ

インディケーターは、次の公式を使用して計算されます。

```
((ts.used * ts.page_size) / ts.max_size) * 100
```

ここで、

- *ts.used* は、 1582 ページの『tablespace_used_pages 表スペース内の使用されているページ数 : モニター・エレメント』 の値です
- *ts.page_size* は、 1569 ページの『tablespace_page_size - 表スペースのページ・サイズ : モニター・エレメント』 の値です

- *ts.max_size* は、 1565 ページの『tablespace_max_size 表スペースの最大サイズ』の値です

自動サイズ変更 DMS 表スペース使用率は、消費された最大表スペース・ストレージのパーセントとして測定されます。高いパーセントは、表スペースが満杯になろうとしていることを示します。この標識の追加情報に含まれる短期間と長期間の増加率は、現行増加率が短期間の例外的なものであるか、長期間にわたる一貫した増加であるかを判別するのに使用されます。

このヘルス・インディケーターで戻される「追加情報」行の「フルになるまでの時間」は、表スペースの最大サイズに達するまでの残りの時間を予測したものです。

ts.ts_util 表スペース使用率:

このヘルス・インディケーターは、各 DMS 表スペースのストレージの使用量を追跡します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID ts.ts_util

ヘルス・モニター・レベル
表スペース

分類 表スペース・ストレージ

タイプ 上限しきい値ベース

単位 パーセンテージ

すべてのコンテナーが満杯の場合は、DMS 表スペースは満杯であると考えられます。

表スペースで自動サイズ変更が使用可能になっている場合、このヘルス・インディケーターは評価されません。代わりに、データベース自動ストレージ使用率 **db.auto_storage_util** と表スペース自動サイズ変更状態 (**ts.ts_auto_resize_status**) ヘルス・インディケーターが表スペースのストレージのモニターに関係します。この表スペースで最大サイズを定義した場合には、自動サイズ変更表スペース使用率 (**ts.ts_util_auto_resize**) ヘルス・インディケーターも使用可能になります。表スペース使用率のパーセンテージは、TBSP_UTILIZATION 管理ビューの TBSP_UTILIZATION_PERCENT 列から取得できます (必要な場合)。

インディケーターは、次の公式を使用して計算されます。

```
(ts.used / ts.usable) * 100
```

ここで、

- *ts.used* は、 1582 ページの『tablespace_used_pages 表スペース内の使用されているページ数 : モニター・エレメント』 の値です

- *ts.usable* は、 1581 ページの『tablespace_usable_pages 表スペース内の使用可能ページ数 : モニター・エレメント』 の値です

表スペース使用率は消費されたスペースのパーセントとして測定され、高いパーセントはこの標識の最適関数を下回っていることを示します。

この標識の追加情報に含まれる短期間と長期間の増加率は、現行増加率が短期間の例外的なものであるか、長期間にわたる一貫した増加であるかを判別するのに使用されます。

このヘルス・インディケーターで戻される「追加情報」行の「フルになるまでの時間」は、表スペースの最大サイズに達するまでの残りの時間を予測したものです。

tsc.tscont_util 表スペース・コンテナー使用率:

このヘルス・インディケーターは、自動ストレージを使用していない各 SMS 表スペースのストレージの使用量を追跡します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID tsc.tscont_util

ヘルス・モニター・レベル
表スペース・コンテナー

分類 表スペース・ストレージ

タイプ 上限しきい値ベース

単位 パーセンテージ

コンテナーが定義されているファイル・システムにスペースが残されていない場合は、 SMS 表スペースは満杯であると考えられます。

SMS コンテナーを拡張するためのフリー・スペースがファイル・システムで使用不可である場合は、関連する表スペースが満杯になります。

フリー・スペースを消費するファイル・システムに定義されている各コンテナーに対してアラートが発行される場合があります。

インディケーターは、次の公式を使用して計算されます。

```
(fs.used / fs.total)*100
```

ここで `fs` はコンテナーがあるファイル・システムです。

SMS 表スペース使用率は消費されたスペースのパーセントとして測定され、高いパーセントはこの標識の最適関数を下回っていることを示します。

この標識の追加情報に含まれる短期間と長期間の増加率は、現行増加率が短期間の例外的なものであるか、長期間にわたる一貫した増加であるかを判別するのに使用されます。

このヘルス・インディケーターで戻される「追加情報」行の「フルになるまでの時間」は、表スペースの最大サイズに達するまでの残りの時間を予測したものです。

ts.ts_op_status 表スペース操作可能状態 ::

表スペースの状態は、実行できるアクティビティーまたはタスクを制約します。正常から他の状態に変わると、アテンション・アラートが生成される場合があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID ts.ts_op_status

ヘルス・モニター・レベル
表スペース

分類 表スペース・ストレージ

タイプ 状態ベース

単位 適用外

tsc.tscont_op_status 表スペース・コンテナー操作可能状態:

このヘルス・インディケーターは、表スペース・コンテナーのアクセス可能性を追跡します。コンテナーのアクセス可能性は、実行できるアクティビティーまたはタスクを制約します。コンテナーがアクセス不能の場合は、アテンション・アラートが生成される場合があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID tsc.tscont_op_status

ヘルス・モニター・レベル
表スペース・コンテナー

分類 表スペース・ストレージ

タイプ 状態ベース

単位 適用外

ソートのヘルス・インディケーター:

db2.sort_privmem_util 専用ソート・メモリー使用率:

この標識は、専用ソート・メモリーの使用率を追跡します。 db2.sort_heap_allocated (システム・モニター・エレメント) >= *sheapthres* (DBM 構成パラメーター) の場合は、ソートは *sortheap* パラメーターで定義されているソート・ヒープを十分に取得していない可能性があり、アラートが生成される場合があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db2.sort_privmem_util

ヘルス・モニター・レベル
データベース

分類 ソート

タイプ 上限しきい値ベース

単位 パーセンテージ

ソートを行うためのヒープ・スペースが十分あり、ソートが不必要にオーバーフローしていなければ、ソートは正常稼働していると考えられます。

インディケーターは、次の公式を使用して計算されます。

```
(db2.sort_heap_allocated / sheapthres)*100
```

「ポストしきい値ソート」スナップショット・モニター・エレメントは、ソート・ヒープしきい値を超えた後に要求されたソート数を測定します。追加の詳細に表示されるこのインディケーターの値は、このヘルス・インディケーターの問題の重大度を示します。

「使用された最大専用ソート・メモリー」スナップショット・モニター・エレメントは、インスタンスの専用ソート・メモリー最高水準点を保持します。追加情報に示されるこの標識の値は、インスタンスが最後に再生された後の任意の時点で使用中だった専用ソート・メモリーの最大量を示します。この値は *sheapthres* の適切な値を決定するために利用できます。

db.sort_shrmem_util 共有ソート・メモリー使用率:

この標識は、共有ソート・メモリーの使用率を追跡します。 *sheapthres_shr* データベース構成パラメーターはハード・リミットです。割り振りが制限に近くなると、アラートが生成される場合があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.sort_shrmem_util

ヘルス・モニター・レベル
データベース

分類 ソート

タイプ 上限しきい値ベース

単位 パーセンテージ

ソートを行うためのヒープ・スペースが十分あり、ソートが不必要にオーバーフローしていなければ、ソートは正常稼働していると考えられます。

インディケーターは、次の公式を使用して計算されます。

```
(db.sort_shrheap_allocated / sheapthres_shr)*100
```

sheapthres_shr が 0 に設定されると、 *sheapthres* は共有ソート・ヒープしきい値として使用されることに注意してください。

「使用された最大共有ソート・メモリー」スナップショット・モニター・エレメントは、データベースの共有ソート・メモリー最高水準点を保持します。追加情報に表示されるこの標識の値は、データベースがアクティブになった後の任意の時点で使用中であった共有ソート・メモリーの最大量を示します。この値は共有ソート・メモリーしきい値の適切な値を決定するために利用できます。

現在のワークロードの必要に応じてソート・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ソート・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

***db.spilled_sorts* オーバーフローしたソートのパーセンテージ:**

ディスクにオーバーフローするソートは、重大なパフォーマンス低下の原因となる可能性があります。これが起こると、アラートが生成される場合があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.spilled_sorts

ヘルス・モニター・レベル
データベース

分類 ソート

タイプ 上限しきい値ベース

単位 パーセンテージ

ソートを行うためのヒープ・スペースが十分あり、ソートが不必要にオーバーフローしていなければ、ソートは正常稼働していると考えられます。

インディケーターは、次の公式を使用して計算されます。

```
(db.sort_overflows_t - db.sort_overflows_t-1) /
(db.total_sorts_t - db.total_sorts_t-1) * 100
```

t は現在のスナップショットで、*t-1* は 1 時間前のスナップショットです。システム・モニター・エレメント db.sort_overflows (sort_overflows モニター・エレメントが基になる) はソート・ヒープを使い果たし、一時記憶域のディスク・スペースを必要とした可能性のあるソートの合計数です。エレメント db.total_sorts (total_sorts モニター・エレメントが基になる) は実行されたソートの合計数です。

現在のワークロードの必要に応じてソート・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ソート・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

db.max_sort_shrmem_util 長期共有ソート・メモリー使用率:

この標識は構成済み共有ソート・ヒープを追跡し、 DB2 データベース・システムの他のどこかでの使用のために解放できるリソースがないかどうかを調べます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.max_sort_shrmem_util

ヘルス・モニター・レベル
データベース

分類 ソート

タイプ 下限しきい値ベース

単位 パーセンテージ

ソートを行うためのヒープ・スペースが十分あり、ソートが不必要にオーバーフローしていなければ、ソートは正常稼働していると考えられます。

使用量のパーセンテージが低い場合はアラートが生成される場合があります。

インディケーターは、次の公式を使用して計算されます。

```
(db.max_shr_sort_mem / sheapthres_shr)*100
```

システム・モニター・エレメント db.max_shr_sort_mem (sort_shrheap_top モニター・エレメントが基になる) は、共有ソート・メモリー使用量の最高水準点です。

現在のワークロードの必要に応じてソート・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ソート・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

データベース・マネージャー (DBMS) のヘルス・インディケーター:

db2.db2_op_status インスタンス操作可能状態:

インスタンス状態が実行されているアクティビティーまたはタスクを制約していないときは、インスタンスは正常稼働していると考えられます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db2.db2_op_status

ヘルス・モニター・レベル
インスタンス

分類 DBMS

タイプ 状態ベース

単位 適用外

状態は、アクティブ、静止ペンディング、静止、またはダウンのいずれかの値になります。非アクティブの状態では、アテンション・アラートが生成される場合があります。

db2.db2_op_status ヘルス・インディケーターがダウン状態に入ると、ヘルス・モニターはこのインディケーターに対するアクションを実行できません。この状態の原因になりうるのは、例えば、明示的な停止要求または異常終了に起因して、インディケーターがモニターしているインスタンスが非アクティブになった場合です。異常終了後、常にインスタンスを自動的に再始動させるには、インスタンスの操作可能状態を維持できるよう障害モニター (**db2fm**) を構成することができます。

インスタンス最大重大度アラート状態 ::

この標識は、モニター対象のインスタンスのロールアップ・アラート状態を表します。インスタンスのアラート状態は、インスタンスとそのデータベース、およびモニター対象のデータベース・オブジェクトの最高レベルのアラート状態です。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID 該当なし。このヘルス・インディケーターには、構成または推奨サポートはありません。

ヘルス・モニター・レベル
インスタンス

分類 DBMS

タイプ 状態ベース

単位 適用外

アラート状態の順序は次のようになります。

- アラーム
- 警告
- アテンション
- Normal

インスタンスのアラート状態によって、DB2 データベース・システムが全体として正常稼働しているかどうかが決まります。

データベースのヘルス・インディケーター:

db.db_op_status データベース操作可能状態 ::

データベースの状態は、実行できるアクティビティーまたはタスクを制約します。状態は、アクティブ、静止ペンディング、静止、またはロールフォワードのいずれかの値になります。アクティブから他の状態に変わると、アテンション・アラートが生成される場合があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.db_op_status

ヘルス・モニター・レベル
データベース

分類 データベース

タイプ 状態ベース

単位 適用外

データベース最大重大度アラート状態:

この標識は、モニター対象のデータベースのロールアップ・アラート状態を表します。データベースのアラート状態は、データベースとそのオブジェクトの最高レベルのアラート状態です。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性がありま

す。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID 該当なし。このヘルス・インディケーターには、構成または推奨サポートはありません。

ヘルス・モニター・レベル
データベース

分類 データベース

タイプ 状態ベース

単位 適用外

アラート状態の順序は次のようになります。

- アラーム
- 警告
- アテンション
- Normal

保守のヘルス・インディケーター:

db.tb_reorg_req 再編成の必要性:

このヘルス・インディケーターは、データベース中の表や索引を再編成する必要性を追跡します。フラグメント化されたデータを除去するには、表か、表に定義されたすべての索引を再編成する必要があります。再編成するには、情報を圧縮して、行か索引データを再構成します。

その結果、パフォーマンスが向上し、表や索引中のフリー・スペースが増えます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.tb_reorg_req

ヘルス・モニター・レベル
データベース

分類 データベース保守

タイプ コレクション状態ベース

単位 適用外

ご使用の自動保守ポリシーに、評価対象となる表の名前を指定することによって、このヘルス・インディケーターにより評価される表集合をフィルターに掛けることができます。これは、「自動保守」ウィザードを使用して行えます。

アテンション・アラートが生成されて、再編成が必要なことが示される場合もあります。 AUTO_REORG データベース構成パラメーターを ON に設定すると、再編成を自動化できます。自動再編成を使用可能にした場合、アテンション・アラートにより、 1 つ以上の自動再編成が正常に完了できなかったこと、あるいは、再編成を必要とする表があるものの、データベース・パーティションごとの表のサイズが、オフライン再編成を考慮すべき表の最大サイズ基準を超えているために、自動再編成が実行されていないことが示されます。注意が必要なオブジェクトのリストについては、このヘルス・インディケーターの集合の詳細を参照してください。

db.tb_runstats_req 統計収集の必要性:

このヘルス・インディケーターは、データベース中の表やそれらの索引の統計を収集する必要性を追跡します。照会の実行時間を改善するには、表および表に定義されたすべての索引に統計が必要です。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.tb_runstats_req

ヘルス・モニター・レベル
データベース

分類 データベース保守

タイプ コレクション状態ベース

単位 適用外

SQL 照会を使用して、このヘルス・インディケーターによって考慮する表を限定できます。この照会のシステム表に対する副選択節が、追加情報中の有効範囲に示されます。

アテンション・アラートが生成されて、統計の収集を促される場合もあります。 AUTO_RUNSTATS データベース構成パラメーターを ON に設定すると、統計を自動的に収集できます。統計の自動収集を使用可能にすると、アテンション・アラートにより、 1 つ以上の統計の自動収集アクションが正常に完了しなかったことが示されます。

db.db_backup_req データベース・バックアップの必要性 ::

このヘルス・インディケーターは、データベースのバックアップの必要性を追跡します。ハードウェアやソフトウェアの障害の場合にデータが消失する可能性から保護するために、リカバリー計画の一部として、定期的にバックアップを取る必要があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳し

くは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.db_backup_req

ヘルス・モニター・レベル
データベース

分類 データベース保守

タイプ 状態ベース

単位 適用外

このヘルス・インディケーターは、経過時間と、前回のバックアップ以後に変更されたデータの量に基づいて、データベースのバックアップが必要な時点を判別します。

アテンション・アラートが生成されて、データベースのバックアップが必要なことが示される場合もあります。 AUTO_DB_BACKUP データベース構成パラメーターを ON に設定すると、データベースのバックアップを自動化できます。自動データベース・バックアップを使用可能にすると、アテンション・アラートにより、 1 つ以上の自動データベース・バックアップが正常に完了しなかったことが示されます。

高可用性災害時リカバリー (HADR) ヘルス・インディケーター:

db.hadr_op_status HADR 操作可能状態 ::

このヘルス・インディケーターは、データベースの高可用性災害時リカバリー (HADR) 操作可能状態を追跡します。 1 次サーバーとスタンバイ・サーバーの間の状態は、接続済み、混雑、または切断のいずれかの値になります。接続済みから他の状態に変わると、アテンション・アラートが生成される場合があります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.hadr_op_status

ヘルス・モニター・レベル
データベース

分類 高可用性災害時リカバリー

タイプ 状態ベース

単位 適用外

db.hadr_delay HADR ログ遅延:

このヘルス・インディケーターは、1 次データベースに対するデータ変更と、スタンバイ・データベースに対するこれらの変更内容のレプリケーションの間の、現在の平均遅延 (分単位) を追跡します。

遅延値が大きい場合は、 1 次データベースに障害が起きた後にスタンバイ・データベースにフェイルオーバーする際に、データ損失が生じる可能性があります。さらに遅延値が大きいと、 1 次データベースがスタンバイ・データベースより優先になっているためテークオーバーが必要な場合に、ダウン時間が長くなる可能性もあります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.hadr_delay

ヘルス・モニター・レベル
データベース

分類 高可用性災害時リカバリー

タイプ 上限しきい値ベース

単位 分

ロギングのヘルス・インディケーター:

db.log_util ログ使用率:

このインディケーターは、データベースで使用されたアクティブ・ログ・スペースの合計量 (バイト数) を追跡します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.log_util

ヘルス・モニター・レベル
データベース

分類 ロギング

タイプ 上限しきい値ベース

単位 パーセンテージ

ログ使用率は消費されたスペースのパーセントとして測定され、パーセンテージが高い場合はアラートが生成される場合があります。

インディケーターは、次の公式を使用して計算されます。

```
(db.total_log_used / (db.total_log_used + db.total_log_available)) * 100
```

追加情報に示されるログ関連のデータベース構成パラメーターの値は、ログの現行割り振りを表示します。追加情報には、最も古いアクティブ・トランザクションを持つアプリケーションのアプリケーション ID も含まれます。このアプリケーションにログ・スペースの解放を強制することができます。

***db.log_fs_util* ログ・ファイル・システム使用率 ::**

ログ・ファイル・システム使用率は、トランザクション・ログが常駐するファイル・システムの使用率を追跡します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.log_fs_util

ヘルス・モニター・レベル
データベース

分類 ロギング

タイプ 上限しきい値ベース

単位 パーセンテージ

ファイル・システムに空きがなければ、DB2 データベース・システムは新規ログ・ファイルを作成できない可能性があります。

ログ使用率は消費されたスペースのパーセントとして測定されます。ファイル・システムのフリー・スペースの量が最小の場合 (つまり使用率のパーセンテージが高い場合) は、アラートが生成される場合があります。

インディケーターは、次の公式を使用して計算されます: *(fs.log_fs_used / fs.log_fs_total)*100*。ここで fs はログが常駐するファイル・システムです。

追加情報に示されるログ関連のデータベース構成パラメーターの値は、ログの現行割り振りを表示します。ユーザー出口が使用可能であるかどうかも追加の詳細情報に示されます。

追加の詳細情報に示される「ディスクが満杯になった時はログをブロック (Block on Log Disk Full)」が「はい (yes)」に設定され、使用率が 100% である場合、ログ・ファイルが正常に作成されるまでトランザクションをコミットできないアプリケーションへの影響を制限するために、アラートをできるだけ早く解決する必要があります。

アプリケーション並行性のヘルス・インディケーター:

***db.deadlock_rate* デッドロック率:**

デッドロック率は、データベースでデッドロックが起きる率と、アプリケーションで競合問題が発生する度合いを追跡します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.deadlock_rate

ヘルス・モニター・レベル
データベース

分類 アプリケーション並行性

タイプ 上限しきい値ベース

単位 時間当たりのデッドロック数

デッドロックは次の状態が原因で起こることがあります。

- データベースでロック・エスカレーションが発生している場合。
- システムが生成した行のロッキング数が十分なときに、アプリケーションが表を明示的にロッキングしている場合。
- アプリケーションがバインディングのときに不適切な分離レベルを使用している場合。
- カタログ表が反復可能読み取りのためにロックされている場合。
- 複数のアプリケーションが同じロックを異なる順序で獲得しているために、デッドロックになっている場合。

インディケーターは、次の公式を使用して計算されます。

$(db.deadlocks_t - db.deadlocks_{t-1})$

ここで *t* は現在のスナップショットで、*t-1* は現在のスナップショットの 60 分前に取られた最後のスナップショットです。

デッドロック率が高くなると、アラートを生成する可能性のある競合の度合いも大きくなります。

db.locklist_util ロック・リスト使用率:

このインディケーターは、使用されているロック・リスト・メモリーの量を追跡します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.locklist_util

ヘルス・モニター・レベル
データベース

分類 アプリケーション並行性

タイプ 上限しきい値ベース

単位 パーセンテージ

データベースごとに 1 つのロック・リストが存在し、ロック・リストには、データベースに同時に接続しているすべてのアプリケーションが保持しているロックが含まれています。ロック・リスト・メモリーには設定限界があります。一度その限界に達すると、次の状態が原因でパフォーマンスが低下します。

- ロック・エスカレーションにより行ロックから表ロックへの変換が行われ、その結果、データベースの共有オブジェクトにおける並行性が低下する。
- アプリケーションによる限定された表ロック待ちのため、アプリケーション間でさらに多くのデッドロックが起きる。その結果、トランザクションがロールバックされます。

ロック要求の最大数がデータベースの限界設定に達すると、アプリケーションにエラーが戻されます。

インディケーターは、次の公式を使用して計算されます。

```
(db.lock_list_in_use / (locklist * 4096)) * 100
```

使用率は消費されたメモリーのパーセントとして測定され、パーセンテージが高い場合は正常稼働ではない状態を示します。

現在のワークロードの必要に応じてロック・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ロック・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

***db.lock_escal_rate* ロック・エスカレーション率:**

このインディケーターは、ロックが行ロックから表ロックにエスカレートされた率を追跡します。このエスカレーションの結果、トランザクションの並行性が影響を受けます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.lock_escal_rate

ヘルス・モニター・レベル
データベース

分類 アプリケーション並行性

タイプ 上限しきい値ベース

単位 時間当たりのロック・エスカレーション数

アプリケーションが保留するロックの合計数がそのアプリケーションで使用可能なロック・リスト・スペースの最大量に達した場合、またはすべてのアプリケーションが使用するロック・リスト・スペースが合計ロック・リスト・スペースに近くなると、ロックはエスカレートされます。使用可能なロック・リスト・スペースの量は、 *maxlocks* および *locklist* データベース構成パラメーターによって決まります。

アプリケーションが許可されているロックの最大数に達し、エスカレートするロックがない場合は、アプリケーションは他のアプリケーションに割り振られたロック・リストのスペースを使用します。データベースごとに 1 つのロック・リストが存在し、ロック・リストには、データベースに同時に接続しているすべてのアプリケーションが保持しているロックが含まれています。ロック・リスト全体が満杯になるとエラーが起こります。

インディケーターは、次の公式を使用して計算されます。

$(db.lock\_escals_{t} - db.lock\_escals_{t-1})$

ここで 't' は現在のスナップショットで、't-1' は現在のスナップショットの 60 分前に取られた最後のスナップショットです。

デッドロック率が高くなると、アラートを生成する可能性のある競合の度合いも大きくなります。

現在のワークロードの必要に応じてロック・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ロック・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

***db.apps_waiting_locks* ロック待機中のアプリケーションのパーセンテージ:**

このインディケーターは、現在実行中でロック待ちのすべてのアプリケーションのパーセンテージを測定します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.apps_waiting_locks

ヘルス・モニター・レベル
データベース

分類 アプリケーション並行性

タイプ 上限しきい値ベース

単位 パーセンテージ

パーセンテージが高い場合は、パフォーマンスに悪影響を及ぼす並行性の問題がアプリケーションで発生していることを示します。

インディケーターは、次の公式を使用して計算されます。

```
(db.locks_waiting / db.appls_cur_cons) *100)
```

パッケージ・キャッシュ、カタログ・キャッシュ、ワークスペースのヘルス・インディケーター:

db.catcache_hitratio カタログ・キャッシュ・ヒット率 ::

ヒット率は、カタログ・キャッシュを使用できることによりディスク上のカタログに実際にアクセスしないで済んでいる程度を示すパーセンテージです。高い率は、実際のディスク入出力アクセスの回避に成功していることを示します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.catcache_hitratio

ヘルス・モニター・レベル
データベース

分類 パッケージおよびカタログ・キャッシュ、およびワークスペース

タイプ 下限しきい値ベース

単位 パーセンテージ

インディケーターは、次の公式を使用して計算されます。

```
(1-(db.cat_cache_inserts/db.cat_cache_lookups))*100
```

db.pkgcache_hitratio パッケージ・キャッシュ・ヒット率 ::

ヒット率は、パッケージ・キャッシュを使用できることによりシステム・カタログから静的 SQL のためのパッケージとセクションを再ロードしないで済み、また動的 SQL ステートメントを再コンパイルしないで済んでいる程度を示すパーセンテージです。高い率は、これらのアクティビティーの回避に成功していることを示します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.pkgcache_hitratio

ヘルス・モニター・レベル
データベース

分類 パッケージおよびカタログ・キャッシュ、およびワークスペース

タイプ 下限しきい値ベース

単位 パーセンテージ

インディケーターは、次の公式を使用して計算されます。

```
(1-(db.pkg_cache_inserts/db.pkg_cache_lookups))*100
```

現在のワークロードの必要に応じてパッケージ・キャッシュ・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。パッケージ・キャッシュ・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

***db.shrworkspace_hitratio* 共有ワークスペース・ヒット率:**

ヒット率は、共有 SQL ワークスペースを使用できることにより、実行されようとしている SQL ステートメントのセクションの初期化が不要になっている程度を示すパーセンテージです。高い率は、このアクティビティーの回避に成功していることを示します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

注: db.shrworkspace_hitratio ヘルス・インディケーターは、DB2 バージョン 9.5 以降、非推奨になりました。このヘルス・インディケーターを使用しても、エラーは生成されません。そして、有効な値も戻されません。この標識はもはや推奨されていません。将来のリリースでは除去される予定です。

ID db.shrworkspace_hitratio

ヘルス・モニター・レベル
データベース

分類 パッケージおよびカタログ・キャッシュ、およびワークスペース

タイプ 下限しきい値ベース

単位 パーセンテージ

インディケーターは、次の公式を使用して計算されます。

```
(1-(db.shr_workspace_section_inserts/db.shr_workspace_section_lookups))*100
```

メモリーのヘルス・インディケーター:

***db2.mon_heap_util* モニター・ヒープ使用率 ::**

このインディケーターは、ID が SQLM_HEAP_MONITOR のメモリー・プールを基に、モニター・ヒープ・メモリーの使用量を追跡します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db2.mon_heap_util

ヘルス・モニター・レベル
インスタンス

分類 メモリー

タイプ 上限しきい値ベース

単位 パーセンテージ

使用率は次の公式を使用して計算されます。

```
(db2.pool_cur_size / db2.pool_config_size) * 100
```

これはメモリー・プール ID が SQLM_HEAP_MONITOR の場合です。

一度このパーセンテージが最大の 100% に達すると、モニター操作が失敗する場合があります。

db.db_heap_util データベース・ヒープ使用率:

このインディケーターは、ID が SQLM_HEAP_DATABASE のメモリー・プールを基に、モニター・ヒープ・メモリーの使用量を追跡します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.db_heap_util

ヘルス・モニター・レベル
データベース

分類 メモリー

タイプ 上限しきい値ベース

単位 パーセンテージ

使用率は次の公式を使用して計算されます。

```
(db.pool_cur_size / db.pool_config_size) * 100
```

これはメモリー・プール ID が SQLM_HEAP_DATABASE の場合です。

一度このパーセンテージが最大の 100% に達すると、使用可能なヒープがないため、照会および操作が失敗する場合があります。

フェデレーテッドのヘルス・インディケーター:

db.fed_nicknames_op_status ニックネームの状態:

このヘルス・インディケーターは、フェデレーテッド・データベース中で定義されているニックネームをすべて検査し、無効なニックネームがあるかどうかを判別します。データ・ソース・オブジェクトがドロップされたり変更が加えられたりした場合や、ユーザー・マッピングが誤っている場合には、ニックネームが無効になることがあります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.fed_nicknames_op_status

ヘルス・モニター・レベル
データベース

分類 フェデレーテッド

タイプ コレクション状態ベース

単位 適用外

フェデレーテッド・データベース中で定義されているニックネームが無効な場合は、アテンション・アラートが生成されることがあります。注意が必要なオブジェクトのリストについては、このヘルス・インディケーターの集合の詳細を参照してください。

このヘルス・インディケーターがニックネームの状況を検査するには、FEDERATED データベース・マネージャー・パラメーターを YES に設定しなければなりません。

db.fed_servers_op_status データ・ソース・サーバーの状態:

このヘルス・インディケーターは、フェデレーテッド・データベース中で定義されているデータ・ソース・サーバーをすべて検査し、使用できないものがあるかどうかを判別します。データ・ソース・ソース・サーバーが停止したり、存在しなくなったり、構成が誤っていたりすると、データ・ソース・サーバーが使用できないことがあります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ID db.fed_servers_op_status

ヘルス・モニター・レベル
データベース

分類 フェデレーテッド

タイプ コレクション状態ベース

単位 適用外

フェデレーテッド・データベース中で定義されているニックネームが無効な場合は、アテンション・アラートが生成されることがあります。注意が必要なオブジェクトのリストについては、このヘルス・インディケーターの集合の詳細を参照してください。

このヘルス・インディケーターがデータ・ソース・サーバーの状況を検査するには、 FEDERATED データベース・マネージャー・パラメーターを YES に設定しなければなりません。

ヘルス・モニター・インターフェース

多様なヘルス・モニター・インターフェースがあります。どのインターフェースを選択するかは、ご使用のデータベース構成と現在の状態によって異なります。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

次の表は、API のヘルス・モニター・インターフェースをリストしています。

表 136. ヘルス・モニター・インターフェース: API

モニター・タスク	API
ヘルス・スナップショットのキャプチャー	db2GetSnapshot - スナップショット・クラス SQLM_CLASS_HEALTH を指定してスナップショットを取得
集合オブジェクトの全リストを含むヘルス・スナップショットのキャプチャー	db2GetSnapshot - スナップショット・クラス SQLM_CLASS_HEALTH および agent_id に SQLM_HMON_OPT_COLL_FULL を指定してスナップショットを取得
公式、追加情報、および履歴を含むヘルス・スナップショットのキャプチャー	db2GetSnapshot - スナップショット・クラス SQLM_CLASS_HEALTH_WITH_DETAIL を指定してスナップショットを取得
公式、追加情報、履歴、および集合オブジェクトの全リストを含むヘルス・スナップショットのキャプチャー	db2GetSnapshot - スナップショット・クラス SQLM_CLASS_HEALTH_WITH_DETAIL および agent_id に SQLM_HMON_OPT_COLL_FULL を指定してスナップショットを取得
自己記述型データ・ストリームの変換	db2ConvMonStream - モニター・ストリームの変換
ヘルス・スナップショットのサイズ見積もり	db2GetSnapshotSize - db2GetSnapshot 出力バッファーに必要なサイズの見積もり

GET HEALTH SNAPSHOT コマンドは、推奨されなくなったヘルス・モニター・コンポーネントの一部です。

次の表は、CLP コマンドのヘルス・モニター・インターフェースをリストしています。

表 137. ヘルス・モニター・インターフェース: CLP コマンド

モニター・タスク	CLP コマンド
ヘルス・スナップショットのキャプチャー	GET HEALTH SNAPSHOT コマンド
公式、追加情報、および履歴を含むヘルス・スナップショットのキャプチャー	GET HEALTH SNAPSHOT WITH DETAILS コマンド

ヘルス・モニター SQL 関数は、推奨されなくなったヘルス・モニター・コンポーネントの一部です。

次の表は、SQL 関数のヘルス・モニター・インターフェースをリストしています。

表 138. ヘルス・モニター・インターフェース: SQL 関数

モニター・タスク	SQL 関数
データベース・マネージャー・レベルの正常性に関する情報のスナップショット	HEALTH_DBM_INFO
データベース・マネージャー・レベルのヘルス・インディケーターのスナップショット	HEALTH_DBM_HI
データベース・マネージャー・レベルのヘルス・インディケーター履歴のスナップショット	HEALTH_DBM_HI_HIS
データベース・レベルの正常性に関する情報のスナップショット	HEALTH_DB_INFO
データベース・レベルのヘルス・インディケーターのスナップショット	HEALTH_DB_HI
データベース・レベルのヘルス・インディケーター履歴のスナップショット	HEALTH_DB_HI_HIS
データベース・レベルのヘルス・インディケーター集合のスナップショット	HEALTH_DB_HIC
データベース・レベルのヘルス・インディケーター収集履歴のスナップショット	HEALTH_DB_HIC_HIS
表スペース・レベルの正常性に関する情報のスナップショット	HEALTH_TBS_INFO
表スペース・レベルのヘルス・インディケーターのスナップショット	HEALTH_TBS_HI
表スペース・レベルのヘルス・インディケーター履歴のスナップショット	HEALTH_TBS_HI_HIS
表スペース・コンテナー・レベルの正常性に関する情報のスナップショット	HEALTH_CONT_INFO
表スペース・コンテナー・レベルのヘルス・インディケーターのスナップショット	HEALTH_CONT_HI
表スペース・コンテナー・レベルのヘルス・インディケーター履歴のスナップショット	HEALTH_CONT_HI_HIS

ヘルス・モニター SQL 表関数:

ヘルス・モニター SQL 表関数に関するこの要約を参照すると役立つ場合があります。ヘルス・モニター SQL 表関数は推奨されなくなったことに注意してください。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

次の表に、スナップショットの表関数をすべてリストします。それぞれの表関数は、ヘルス・スナップショット要求タイプに対応しています。

表 139. スナップショット・モニター SQL 表関数

モニター・レベル	SQL 表関数	戻される情報
データベース・マネージャー	HEALTH_DBM_INFO	データベース・マネージャー・レベルからのヘルス・スナップショットに関する基本情報
データベース・マネージャー	HEALTH_DBM_HI	データベース・マネージャー・レベルからのヘルス・インディケーター情報
データベース・マネージャー	HEALTH_DBM_HI_HIS	データベース・マネージャー・レベルからのヘルス・インディケーター履歴情報
データベース	HEALTH_DB_INFO	データベースからのヘルス・スナップショットに関する基本情報
データベース	HEALTH_DB_HI	データベースからのヘルス・インディケーター情報
データベース	HEALTH_DB_HI_HIS	データベースからのヘルス・インディケーター履歴情報
データベース	HEALTH_DB_HIC	データベースのコレクション・ヘルス・インディケーターに関するコレクション情報
データベース	HEALTH_DB_HIC_HIS	データベースのコレクション・ヘルス・インディケーターに関するコレクション履歴情報
表スペース	HEALTH_TBS_INFO	データベースの表スペースのヘルス・スナップショットに関する基本情報
表スペース	HEALTH_TBS_HI	データベースの表スペースに関するヘルス・インディケーター情報
表スペース	HEALTH_TBS_HI_HIS	データベースの表スペースに関するヘルス・インディケーター履歴情報
表スペース	HEALTH_CONT_INFO	データベースのコンテナーのヘルス・スナップショットに関する基本情報
表スペース	HEALTH_CONT_HI	データベースのコンテナーに関するヘルス・インディケーター情報
表スペース	HEALTH_CONT_HI_HIS	データベースのコンテナーに関するヘルス・インディケーター履歴情報

ヘルス・モニター CLP コマンド:

ヘルス・モニター・コマンドを実行すると、データベース・マネージャーおよびデータベースのヘルス状況情報を取得できます。

戻された情報は、コマンドを発行した時点でのヘルス状態のスナップショットを表しています。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

次の表に、サポートされているスナップショット要求のタイプをすべてリストします。

表 140. スナップショット・モニター CLP コマンド

モニター・レベル	CLP コマンド	戻される情報
データベース・マネージャー	`get health snapshot for dbm`	データベース・マネージャー・レベル情報。
データベース	`get health snapshot for all databases`	データベース・レベル情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
データベース	`get health snapshot for database on` *`database-alias`*	データベース・レベル情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
データベース	`get health snapshot for all on` *`database-alias`*	データベース、表スペース、および表スペース・コンテナーの情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
表スペース	`get snapshot for tablespaces on` *`database-alias`*	データベースに接続されたアプリケーションがアクセスしている各表スペースの表スペース・レベルの情報。また、表スペース中の各表スペース・コンテナーに関する正常性の情報も含みます。

ヘルス・モニター API 要求タイプ:

スナップショット・モニター API 要求タイプのこの要約は、ご使用の状況に最適なタイプを判別する際に役立ちます。

次の表に、サポートされているスナップショット要求のタイプをすべてリストします。

表 141. スナップショット・モニター API 要求タイプ

モニター・レベル	API 要求タイプ	戻される情報
データベース・マネージャー	SQLMA_DB2	データベース・マネージャー・レベル情報。

表 141. スナップショット・モニター API 要求タイプ (続き)

モニター・レベル	API 要求タイプ	戻される情報
データベース	SQLMA_DBASE_ALL	データベース・レベル情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
データベース	SQLMA_DBASE	データベース・レベル情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
表スペース	SQLMA_DBASE_TABLESPACES	データベースに接続されたアプリケーションがアクセスしている各表スペースの表スペース・レベルの情報。また、表スペース中の各表スペース・コンテナーに関する正常性の情報も含みます。

ヘルス・モニター・インターフェースの論理データ・グループへのマッピング

このリファレンスは、現在の状況で最適なヘルス・モニター・インターフェースを判別するのに役立つ可能性があります。

次の表に、サポートされているヘルス・スナップショット要求のタイプをすべてリストします。

表 142. ヘルス・モニター・インターフェースの論理データ・グループへのマッピング

API 要求タイプ	CLP コマンド	SQL 表関数	論理データ・グループ
SQLMA_DB2	get health snapshot for dbm	HEALTH_DBM_INFO	db2
		HEALTH_DBM_HI	health_indicator
	get health snapshot for dbm show detail	HEALTH_DBM_HI_HIS	health_indicator_history
SQLMA_DBASE	get health snapshot for database on *dbname*	HEALTH_DB_INFO	dbase
		HEALTH_DB_HI	health_indicator
	get health snapshot for database on *dbname* show detail	HEALTH_DB_HI_HIS	health_indicator_history
SQLMA_DBASE with SQLM_HMON_OPT_COLL_FULL in the agent_id	get health snapshot for database on *dbname* with full collection	HEALTH_DB_HIC	health_indicator, hi_obj_list
	get health snapshot for database on *dbname* show detail with full collection	HEALTH_DB_HIC_HIST	health_indicator_history, hi_obj_list
SQLMA_DBASE_ALL	get health snapshot for all databases	HEALTH_DB_INFO	dbase
		HEALTH_DB_HI	health_indicator
	get health snapshot for all databases show detail	HEALTH_DB_HI_HIS	health_indicator_history

表 142. ヘルス・モニター・インターフェースの論理データ・グループへのマッピング (続き)

API 要求タイプ	CLP コマンド	SQL 表関数	論理データ・グループ
SQLMA_DBASE_TABLESPACES	get health snapshot for tablespaces on *dbname*	HEALTH_TS_INFO	tablespace
		HEALTH_TS_HI	health_indicator
		HEALTH_CONT_INFO	tablespace_container
		HEALTH_CONT_HI	health_indicator
	get health snapshot for tablespaces on *dbname* show detail	HEALTH_TS_HI_HIS	health_indicator_history
		HEALTH_CONT_HI_HIS	health_indicator_history

以下の図は、論理データ・グループがヘルス・スナップショット・データ・ストリーム内に現れる順番を示しています。

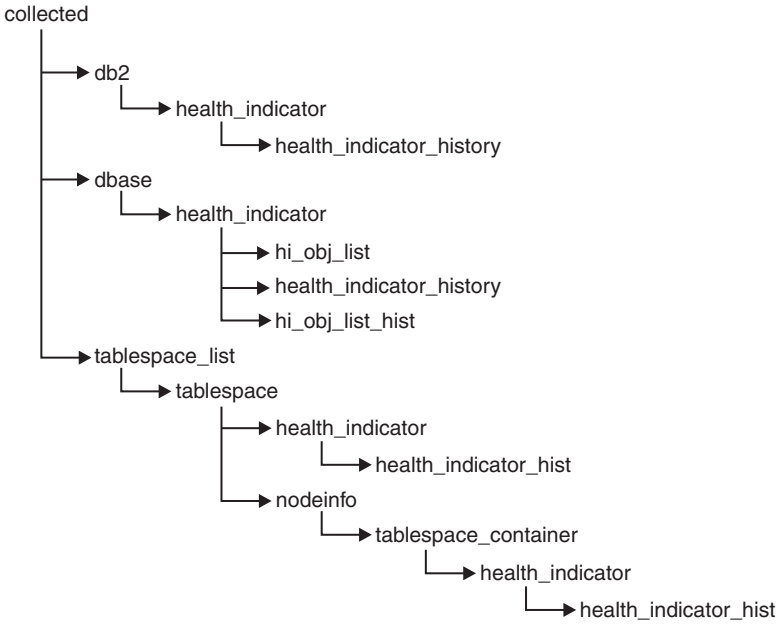


図 9. ヘルス・スナップショットの論理データ・グループ

ヘルス・アラート通知の使用可能化

アラートの生成時に電子メールまたはページャーで通知できるようにするには、構成パラメーターを設定して連絡先情報を指定しなければなりません。

始める前に

連絡先リストのあるシステム上で、 DB2 Administration Server (DAS) が実行されている必要があります。例えば、CONTACT_HOST 構成パラメーターがリモート・システムに設定されている場合は、連絡先にアラートを通知するには、リモート・システム上で DAS が実行されていなければなりません。

このタスクについて

ヘルス・アラート通知を使用可能にするには、次のようにします。

手順

1. SMTP_SERVER パラメーターを指定します。 DAS 構成パラメーター SMTP_SERVER は、電子メールとページャーの両方の通知メッセージを送信する際に使用するメール・サーバーの場所を指定します。 DB2 データベースのインストール先のシステムが非認証 SMTP サーバーとして使用可能になっている場合は、このステップを省略してください。
2. CONTACT_HOST パラメーターを指定します。 DAS 構成パラメーター CONTACT_HOST は、ローカル・システム上のすべてのインスタンス用の連絡先リストのリモート位置を指定します。このパラメーターを設定すると、複数のシステム間で 1 つの連絡先リストを共有できます。 DB2 データベースのインストール先のローカル・システム上に連絡先リストを維持する場合は、このステップを省略してください。
3. ヘルス・モニター通知のデフォルトの連絡先を指定します。アラートの生成時にヘルス・モニターから電子メールまたはページャー通知を行えるようにするには、デフォルトの管理連絡先を指定しなければなりません。この情報を指定しないことを選択した場合は、アラート条件に関する通知メッセージを送信できません。インストール中にデフォルトの管理連絡先情報を指定するか、またはインストールが完了するまでこの作業を遅らせることができます。 この作業を遅らせることを選択した場合や、通知リストにさらに連絡先かグループを追加しようとする場合は、CLP、または C API を使用して連絡先を指定できます。

   - **CLP を使用して連絡先を指定するには、次のようにしてください。**
     ヘルス・モニター通知のデフォルトとして電子メールの連絡先を定義するには、次のコマンドを発行してください。

     ```
     DB2 ADD CONTACT contact_name TYPE EMAIL ADDRESS
        email_address DESCRIPTION 'Default Contact'

     DB2 UPDATE NOTIFICATION LIST ADD CONTACT contact_name
     ```

     完全な構文の詳細については、「コマンド・リファレンス」を参照してください。

   - **C API を使用して連絡先を指定するには、次のようにしてください。**
     次の C コードの抜粋は、正常性の通知の連絡先を定義する方法を図示しています。

     ```
     ...
     #include <db2ApiDf.h>

     SQL_API_RC rc = 0;
     struct db2AddContactData addContactData;
     struct sqlca sqlca;

     char* userid = "myuser";
     char* password = "pwd";
     char* contact = "DBA1";
     char* email = "dba1@mail.com";
     char* desc = "Default contact";

     memset(&addContactData, '¥0', sizeof(addContactData));
     memset (&sqlca, '¥0', sizeof(struct sqlca));

     addContactData.piUserid = userid;
     ```

```
addContactData.piPassword = password;
addContactData.piName = contact;
addContactData.iType = DB2CONTACT_EMAIL;
addContactData.piAddress = email;
addContactData.iMaxPageLength = 0;
addContactData.piDescription = desc;

rc = db2AddContact(db2Version810, &addContactData, &sqlca);

if (rc == 0) {
   db2HealthNotificationListUpdate update;
   db2UpdateHealthNotificationListData data;
   db2ContactTypeData contact;

   contact.pName = contact;
   contact.contactType = DB2CONTACT_EMAIL;

   update.iUpdateType = DB2HEALTHNOTIFICATIONLIST_ADD;
   update.piContact = &contact;

   data.iNumUpdates = 1;
   data.piUpdates = &update;

   rc = db2UpdateHealthNotificationList (db2Version810, &data, &ca);
}
...
```

-

ヘルス・モニター

ヘルス・モニターは、データベース・マネージャー、データベース、表スペース、および表スペース・コンテナーに関する情報をキャプチャーします。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

ヘルス・モニターは、データベース・システム・モニター・エレメント、オペレーティング・システム、および DB2 データベースから取り出されるデータに基づいて、ヘルス・インディケーターを計算します。ヘルス・モニターがデータベースとそのオブジェクトに関するヘルス・インディケーターを評価できるのは、データベースがアクティブの場合に限ります。 **ACTIVATE DATABASE** コマンドを使用してデータベースを開始するか、データベースに対する永続接続を保守することにより、データベースをアクティブにしておくこともできます。

ヘルス・モニターは、ヘルス・インディケーターごとに最大 10 個の履歴レコードを保持します。この履歴は、 *instance_ path*¥hmonCache ディレクトリーに格納され、ヘルス・モニターの停止時に除去されます。ヘルス・モニターは、レコードの最大数に達した時点で、古い履歴レコードを自動的に整理します。

ヘルス・モニター・データにはヘルス・スナップショットを使用してアクセスできます。個々のヘルス・スナップショットは、最新のリフレッシュ・インターバルに

基づいて、ヘルス・インディケーターごとに状況を報告します。スナップショットは、既存のデータベースの正常性に関する問題を検出したり、データベース環境で正常性が低くなる可能性を予測したりするのに便利です。ヘルス・スナップショットは、 C または C++ アプリケーション中の API を使用して CLP からキャプチャーしたり、グラフィック管理ツールを使用することによってキャプチャーしたりすることができます。

ヘルス・モニターでは、インスタンス接続が必要です。 **ATTACH TO** コマンドを使用してインスタンス接続が確立されていない場合、ローカル・インスタンスへのデフォルトのインスタンス接続が作成されます。

パーティション・データベース環境では、スナップショットは、インスタンスの任意のパーティションでとることも、単一のインスタンス接続を使用してグローバルにとることもできます。グローバル・スナップショットは、それぞれのパーティションで収集されたデータを集約して単一の値セットを戻します。

使用上の注意

ヘルス・モニターは、DB2 データベースのすべてのエディションでサポートされています。

Windows では、DB2 インスタンスのサービスは、 SYSADM 権限のあるアカウントの下で実行する必要があります。管理者特権のあるアカウントを使用するには、**db2icrt** コマンド上で「**-u**」オプションを使用するか、または Windows の「サービス」フォルダーを使用して、「ログオン」プロパティーを編集します。

ヘルス・モニター・プロセスは、DB2 fenced モード・プロセスとして実行されます。これらのプロセスは、Windows では DB2FMP として表示されます。他のプラットフォームでは、ヘルス・モニター・プロセスは DB2ACD として表示されます。

通知が送信され、アラート・アクションが実行されるには、ヘルス・モニターのあるシステム上で、 DB2 Administration server が実行されていなければなりません。リモートのスクリプト、タスク、または連絡先リストを使用する場合は、リモート・システム上で DB2 Administration server も開始しなければなりません。

ツール・カタログ・データベースは、タスクを作成する場合のみ必要です。ヘルス・インディケーターに関するアラート・タスク・アクションを使用しない場合は、ヘルス・モニターにはツール・カタログ・データベースは必要ありません。

ヘルス・インディケーターのデータ

ヘルス・モニターは、各データベース・パーティションのヘルス・インディケーターごとにデータの集合を記録します。

このデータには、以下が含まれます。

- ヘルス・インディケーター名
- 値
- 評価タイム・スタンプ
- アラート状態
- 公式 (該当する場合)

- 追加情報 (該当する場合)
- 最新のヘルス・インディケーター評価の履歴 (最大 10)。個々の履歴項目は、次のヘルス・インディケーター評価を、現行ヘルス・インディケーターの出力に至るまでキャプチャーします。
  - 値
  - 公式 (該当する場合)
  - アラート状態
  - タイム・スタンプ

ヘルス・モニターは、インスタンス、データベース、および表スペース・レベルで、最大重大度アラート状態の追跡も行います。それぞれのレベルで、このヘルス・インディケーターは、そのレベルと、そのレベルより低いすべてのレベルのヘルス・インディケーターに関する、既存の最大重大度アラートを表します。例えば、インスタンスに関する最大重大度アラート状態には、インスタンス、そのすべてのデータベース、およびそれらのデータベースごとのすべての表スペースと表スペース・コンテナーに関するヘルス・インディケーターが含まれます。

データベースのヘルス・スナップショットのキャプチャー

SQL 表関数を使用したデータベースのヘルス・スナップショットのキャプチャー:

SQL 表関数を使用して、データベースのヘルス・スナップショットをキャプチャーすることができます。使用可能なそれぞれのヘルス・スナップショット表関数は、ヘルス・スナップショット要求タイプに対応しています。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

このタスクについて

SQL 表関数を使用して、データベースのヘルス・スナップショットをキャプチャーするには、次のようにします。

手順

1. 使用しようとしている SQL 表関数を識別します。

   SQL 表関数には、以下の 2 つの入力パラメーターがあります。

   - VARCHAR(255): データベース名。
   - INT: パーティション番号 (0 から 999 の間の値)。モニターするパーティション番号に対応する整数を入力します。現在接続しているパーティションのスナップショットをキャプチャーする場合は、値 -1 を入力します。グローバル・スナップショットをキャプチャーするには、値 -2 を入力します。

   注: データベース・マネージャーのスナップショット SQL 表関数だけはこの規則の例外です。それには 1 つのパラメーターしかありません。この 1 つのパラ

メーターは、パーティション番号のパラメーターです。データベース名パラメーターに NULL を入力すると、モニターは、表関数の呼び出しに使用される接続によって定義されたデータベースを使用します。

2. SQL ステートメントを発行します。

   以下の例では、現在接続されているパーティション、およびこの表関数呼び出しが行われた接続によって定義されたデータベース上についての、基本的なヘルス・スナップショットをキャプチャーします。

   ```
   SELECT * FROM TABLE( HEALTH_DB_INFO( cast (NULL as VARCHAR(1)), -1))
          as HEALTH_DB_INFO
   ```

   戻り表から個々のモニター・エレメントを選択することもできます。戻り表の各列は、モニター・エレメントに対応しています。したがって、各モニター・エレメントの列名は、モニター・エレメントの名前に対応しています。次のステートメントの場合は、 db_path と server_platform のモニター・エレメントだけが戻されます。

   ```
   SELECT db_path, server_platform
          FROM TABLE( HEALTH_DB_INFO( cast (NULL as VARCHAR(1)), -1 ) )
          as HEALTH_DB_INFO
   ```

CLP を使用したデータベースのヘルス・スナップショットのキャプチャー:

CLP から GET HEALTH SNAPSHOT コマンドを使用して、ヘルス・スナップショットをキャプチャーすることができます。コマンド構文は、ヘルス・モニターによってモニターされたさまざまなタイプのヘルス・スナップショット情報の取り出しをサポートします。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

始める前に

ヘルス・スナップショットをキャプチャーするには、インスタンス接続がなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのスナップショットを取得するには、まずそのインスタンスにアタッチする必要があります。

このタスクについて

CLP を使用して、データベースのヘルス・スナップショットをキャプチャーするには、次のようにします。

手順

1. CLP から、GET HEALTH SNAPSHOT コマンドを必要なパラメーターを指定して発行します。

次の例では、データベース・マネージャーの開始直後に、データベース・マネージャー・レベルのヘルス・スナップショットがキャプチャーされます。

```
db2 get health snapshot for dbm
```

2. パーティション・データベース・システムの場合、特定のパーティションについてデータベース・スナップショットを固有にキャプチャーすることも、すべてのパーティションについてグローバルなスナップショットをキャプチャーすることもできます。特定のパーティション (例えば、パーティション番号 2) 上のデータベースのヘルス・スナップショットをキャプチャーするには、次のコマンドを発行します。

   ```
   db2 get health snapshot for db on sample at dbpartitionnum 2
   ```

   すべてのパーティション上のすべてのアプリケーションについてのデータベース・スナップショットをキャプチャーするには、次のコマンドを発行します。

   ```
   db2 get health snapshot for db on sample global
   ```

   次のコマンドは、公式、追加情報、およびヘルス・インディケーター履歴を含む追加の詳細情報のある、ヘルス・スナップショットをキャプチャーします。

   ```
   db2 get health snapshot for db on sample show detail
   ```

3. コレクション状態ベースのヘルス・インディケーターの場合、状態にかかわらず、すべてのコレクション・オブジェクトについてデータベース・スナップショットをキャプチャーすることができます。正規の GET HEALTH SNAPSHOT FOR DB コマンドは、すべてのコレクション状態ベースのヘルス・インディケーターについて、アラートが必要なすべてのコレクション・オブジェクトを戻します。

   すべてのコレクション・オブジェクトをリストしてデータベースのヘルス・スナップショットをキャプチャーするには、次のコマンドを発行します。

   ```
   db2 get health snapshot for db on sample with full collection
   ```

クライアント・アプリケーションからのデータベースのヘルス・スナップショットのキャプチャー:

C または C++ アプリケーションでスナップショット・モニター API を使用して、ヘルス・スナップショットをキャプチャーすることができます。 db2GetSnapshot API にパラメーターを指定することにより、多数の異なるヘルス・スナップショット要求タイプにアクセスすることができます。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

始める前に

ヘルス・スナップショットをキャプチャーするには、インスタンスにアタッチしていなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのスナップショットを取得するには、まずそのインスタンスにアタッチする必要があります。

手順

1. コードに sqlmon.h および db2ApiDf.h DB2 ライブラリーを組み込みます。これらのライブラリーは、 sqllib¥include ディレクトリーにあります。

   ```
   #include <db2ApiDf.h>
   #include <sqlmon.h>
   ```

2. スナップショットのバッファー単位サイズを 50 KB に設定します。

   ```
   #define SNAPSHOT_BUFFER_UNIT_SZ 51200
   ```

3. sqlma、sqlca、sqlm_collected、および db2GetSnapshotData 構造体を宣言します。

   ```
   struct sqlma *pRequestedDataGroups;
   struct sqlca sqlca;
   memset (&sqlca, '¥0', sizeof(struct sqlca));
   struct sqlm_collected collectedData;
   memset (&sqlm_collected, '¥0', sizeof(struct sqlm_collected));
   db2GetSnapshotData getSnapshotParam;
   memset(&db2GetSnapshotData, '¥0', sizeof(db2GetSnapshotData));
   ```

4. スナップショット・バッファーを含み、バッファーのサイズを設定するようにポインターを初期化します。

   ```
   static sqluint32 snapshotBufferSize = SNAPSHOT_BUFFER_UNIT_SZ;
   sqluint32 outputFormat;
   char *snapshotBuffer;
   ```

5. sqlma 構造体を初期化し、キャプチャーしようとしているスナップショットがデータベース・マネージャー・レベル情報のものであることを指定します。

   ```
   pRequestedDataGroups = (struct sqlma *)malloc(SQLMASIZE(1));
   memset(&pRequestedDataGroups, '¥0', sizeof(struct pRequestedDataGroups));
   pRequestedDataGroups->obj_num = 1;
   pRequestedDataGroups->obj_var[0].obj_type = SQLMA_DB2;
   ```

6. スナップショット・リスト出力を保持するバッファーを初期化します。

   ```
   snapshotBuffer = (char *)malloc(snapshotBufferSize);
   memset (&snapshotBuffer, '¥0', sizeof(snapshotBuffer));
   ```

7. db2GetSnapshotData 構造体に、スナップショット要求タイプ (sqlma 構造体から)、バッファー情報、およびスナップショットをキャプチャーするために必要な他の情報を含めます。

   ```
   getSnapshotParam.piSqlmaData = pRequestedDataGroups;
   getSnapshotParam.poCollectedData = &collectedData;
   getSnapshotParam.poBuffer = snapshotBuffer;
   getSnapshotParam.iVersion = SQLM_DBMON_VERSION9_5;
   getSnapshotParam.iBufferSize = snapshotBufferSize;
   getSnapshotParam.iStoreResult = 0;
   getSnapshotParam.iNodeNumber = SQLM_CURRENT_NODE;
   getSnapshotParam.poOutputFormat = &outputFormat;
   getSnapshotParam.iSnapshotClass = SQLM_CLASS_HEALTH;
   ```

8. ヘルス・スナップショットをキャプチャーします。以下のパラメーターを渡します。

   - db2GetSnapshotData 構造体。これには、スナップショットをキャプチャーするのに必要な情報が含まれています。
   - スナップショット出力の宛先となるバッファーの参照。

   ```
   db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
   ```

9. バッファーのオーバーフローを処理するためのロジックを組み込みます。スナップショットが取られた後、バッファー・オーバーフローについて sqlcode がチェックされます。バッファー・オーバーフローが発生する場合には、バッファーがクリアされて再初期化され、スナップショットが再度取られます。

```
while (sqlca.sqlcode == 1606)
{
  free(snapshotBuffer);
  snapshotBufferSize += SNAPSHOT_BUFFER_UNIT_SZ;
  snapshotBuffer = (char *)malloc(snapshotBufferSize);
  if (snapshotBuffer == NULL)
  {
    printf("¥nMemory allocation error.¥n");
    return;
  }

  getSnapshotParam.iBufferSize = snapshotBufferSize;
  getSnapshotParam.poBuffer = snapshotBuffer;
  db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
}
```

10. スナップショット・モニターのデータ・ストリームを処理します。スナップショット・モニターのデータ・ストリームを参照するには、これらのステップの後の図を参照してください。
11. バッファーをクリアします。

```
free(snapshotBuffer);
free(pRequestedDataGroups);
```

タスクの結果

db2GetSnapshot API でヘルス・スナップショットをキャプチャーした後、ヘルス・スナップショット出力が自己記述型データ・ストリームとして戻されます。以下は、データ・ストリーム構造の例です。

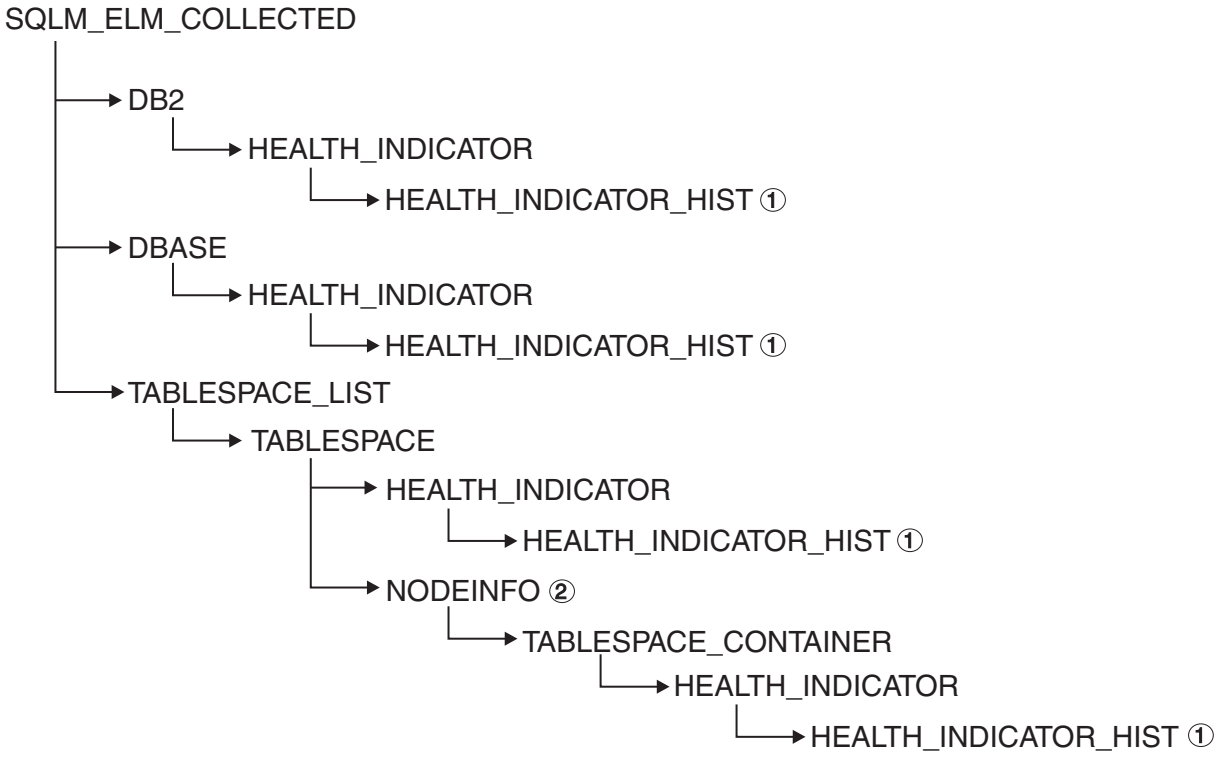


図 10. ヘルス・スナップショット自己記述型データ・ストリーム

凡例:

1. SQLM_CLASS_HEALTH_WITH_DETAIL スナップショット・クラスが使用される場合のみ使用可能。
2. DB2 Enterprise Server Edition でのみ使用可能。それ以外の場合は、表スペース・コンテナー・ストリームになります。

次の階層は、ヘルス・スナップショット自己記述型データ・ストリーム中の特定のエレメントを示しています。

SQLM_ELM_HI の下のエレメントの階層:

```
SQLM_ELM_HI
   SQLM_ELM_HI_ID
   SQLM_ELM_HI_VALUE
      SQLM_ELM_HI_TIMESTAMP
         SQLM_ELM_SECONDS
         SQLM_ELM_MICROSEC
   SQLM_ELM_HI_ALERT_STATE
```

SQLM_ELM_HI_HIST の下のエレメントの階層
(SQLM_CLASS_HEALTH_WITH_DETAIL スナップショット・クラス使用時のみ使用可能):

```
SQLM_ELM_HI_HIST
   SQLM_ELM_HI_FORMULA
   SQLM_ELM_HI_ADDITIONAL_INFO
      SQLM_ELM_HEALTH_INDICATOR_HIST
         SQLM_ELM_HI_ID
         SQLM_ELM_HI_VALUE
            SQLM_ELM_HI_TIMESTAMP
               SQLM_ELM_SECONDS
```

```
      SQLM_ELM_MICROSEC
SQLM_ELM_HI_ALERT_STATE
SQLM_ELM_HI_FORMULA
SQLM_ELM_HI_ADDITIONAL_INFO
```

SQLM_ELM_OBJ_LIST の下のエレメントの階層:

```
SQLM_ELM_HI_OBJ_LIST
   SQLM_ELM_HI_OBJ_NAME
   SQLM_ELM_HI_OBJ_DETAIL
   SQLM_ELM_HI_OBJ_STATE
      SQLM_ELM_HI_TIMESTAMP
         SQLM_ELM_SECONDS
         SQLM_ELM_MICROSEC
```

SQLM_ELM_OBJ_LIST_HIST の下のエレメントの階層 (SQLM_CLASS_HEALTH_WITH_DETAIL スナップショット・クラス使用時のみ使用可能):

```
SQLM_ELM_HI_OBJ_LIST_HIST
   SQLM_ELM_HI_OBJ_NAME
   SQLM_ELM_HI_OBJ_STATE
      SQLM_ELM_HI_TIMESTAMP
         SQLM_ELM_SECONDS
         SQLM_ELM_MICROSEC
```

ヘルス・モニターの出力例

ヘルス・モニターでの作業中、サンプル出力を参照すると役立つ場合があります。

次の例は、CLP を使用して取ったヘルス・スナップショットとそれらに対応する出力を示し、ヘルス・モニターの性質を図示します。この例の目的は、データベース・マネージャーの開始直後に全体の正常性に関する状況を検査することです。

1. 次のように GET HEALTH SNAPSHOT コマンドを使用して、データベース・マネージャーのスナップショットを取ります。

   ```
   db2 get health snapshot for dbm
   ```

   CLP から GET HEALTH SNAPSHOT コマンドが発行された後、スナップショット出力が画面に送られます。

   ```
   Node name                                    =
   Node type                                    = Database Server with local
                                                  and remote clients
   Instance name                                = DB2
   Snapshot timestamp                           = 11-07-2002 12:43:23.613425

   Number of database partitions in DB2 instance  = 1
   Start Database Manager timestamp             = 11-07-2002 12:43:18.000108
   Instance highest severity alert state        = Not yet evaluated

   Health Indicators:

       Not yet evaluated
   ```

2. この出力を分析します。このヘルス・スナップショットから、インスタンスの最大重大度アラート状態が「Not yet evaluated」であることが分かります。ヘルス・モニターが開始されたばかりで、まだヘルス・インディケーターを評価していないので、インスタンスはこの状態です。

   インスタンスの最大重大度アラート状態が変わらない場合、次のようにします。

- HEALTH_MON データベース・マネージャー構成パラメーターの値を検査して、ヘルス・モニターが ON かどうか判別する。
- HEALTH_MON=OFF の場合は、ヘルス・モニターが開始されていない。ヘルス・モニターを開始するには、 UPDATE DBM CFG USING HEALTH_MON ON コマンドを発行します。
- HEALTH_MON=ON の場合は、インスタンスにアタッチしてヘルス・モニターを活動化する。インスタンス接続がある場合は、ヘルス・モニターをメモリー中にロードできなかった可能性があります。

CLP を使用してデータベースのヘルス・スナップショットをとる別の例を以下に概略します。

1. 始める前に、データベース接続が存在し、データベースが静止していることを確認してください。
2. 次のように GET HEALTH SNAPSHOT コマンドを使用して、データベース・マネージャーのスナップショットを取ります。

   ```
   db2 get health snapshot for db on sample
   ```

3. CLP から GET HEALTH SNAPSHOT コマンドが発行された後、スナップショット出力が画面に送られます。

   ```
                Database Health Snapshot

   Snapshot timestamp                          = 12-09-2002 11:44:37.793184

   Database name                               = SAMPLE
   Database path                               = E:¥DB2¥NODE0000¥SQL00002¥
   Input database alias                        = SAMPLE
   Operating system running at database server= NT
   Location of the database                    = Local
   Database highest severity alert state       = Attention

   Health Indicators:

    ...
      Indicator Name                              = db.log_util
         Value                                    = 60
         Unit                                     = %
         Evaluation timestamp                     = 12-09-2002 11:44:00.095000
         Alert state                              = Normal

      Indicator Name                              = db.db_op_status
         Value                                    = 2
         Evaluation timestamp                     = 12-09-2002 11:44:00.095000
         Alert state                              = Attention
   ```

4. この出力を分析します。

   このヘルス・スナップショットは、 *db.db_op_status* ヘルス・インディケーター上にアテンション・アラートがあることを示しています。値 2 は、データベースが静止状態であることを示しています。

グローバル・ヘルス・スナップショット

パーティション・データベース・システムでは、現行パーティション、指定したパーティション、またはすべてのパーティションのヘルス・スナップショットをとることができます。パーティション・データベースのすべてのパーティションに渡ってグローバル・ヘルス・スナップショットをとる場合は、可能であれば、データが集約されてから、結果が戻されます。

ヘルス・インディケーターの集約されたアラート状態は、すべてのデータベース・パーティション間の最大重大度アラート状態と等しくなります。追加情報と履歴データは、データベース・パーティション間で集約できないので使用できません。ヘルス・インディケーターの残りのデータは、以下の表に詳述されているように集約されます。

表 143. ヘルス・インディケーターの値、タイム・スタンプ、および公式データの集約

ヘルス・インディケーター	集約の詳細
• db2.db2_op_status • db2.sort_privmem_util • db2.mon_heap_util • db.db_op_status • db.sort_shrmem_util • db.spilled_sorts • db.log_util • db.log_fs_util • db.locklist_util • db.apps_waiting_locks • db.db_heap_util • db.db_backup_req • ts.ts_util	ヘルス・インディケーターの値は、最大値を含むパーティションから取得される。 評価タイム・スタンプと公式は、同じパーティションから取得される。
• db.max_sort_shrmem_util • db.pkgcache_hitratio • db.catcache_hitratio • db.shrworkspace_hitratio	ヘルス・インディケーターの値は、最小値を含むパーティションから取得される。 評価タイム・スタンプと公式は、同じパーティションから取得される。
• db.deadlock_rate • db.lock_escal_rate	ヘルス・インディケーターの値は、すべてのデータベース・パーティション間の値の合計になる。 評価タイム・スタンプと公式は集約できないので使用できない。
• ts.ts_op_status • tsc.tscont_op_status • tsc.tscont_util	ヘルス・インディケーターは集約されない。
• db.hadr_op_status • db.hadr_log_delay	これらのヘルス・インディケーターは、複数のパーティション・データベースではサポートされない。
• db.tb_reorg_req • db.tb_runstats_req • db.fed_nicknames_op_status • db.fed_servers_op_status	このヘルス・インディケーターは、1 つのパーティションのみに対して評価されるので、集約の必要はない。ヘルス・インディケーターを評価するパーティションからデータが戻されます。

注: 1 つのパーティション・オブジェクトに関するグローバル・スナップショットをとると、集約するパーティションがないので、出力にはすべての属性が含まれます。

正常性の推奨事項の取得

正常性を保つための推奨事項の SQL による照会:

SYSPROC.HEALTH_HI_REC ストアード・プロシージャーを使用して、 SQL で推奨事項を照会できます。

SYSPROC.HEALTH_HI_REC ストアード・プロシージャーを使用すると、推奨事項は次のような XML 文書で戻されます。

- sqllib¥misc ディレクトリー中にある正常性の推奨事項の XML スキーマ DB2RecommendationSchema.xsd に従ってフォーマットされている。
- UTF-8 でエンコードされ、クライアントの言語のテキストが含まれている。
- 推奨事項の集合を収集したものとして編成されている。個々の推奨事項の集合には、解決しようとしている問題 (ヘルス・インディケーター) が記述され、そのヘルス・インディケーターを解決する際の 1 つ以上の推奨事項が含まれます。この文書から検索できる情報に関する特定の詳細情報については、スキーマ定義を参照してください。

CLP を使用して入手できる情報はすべて、 SQL を使用して照会すると戻される XML 推奨文書でも入手できます。

SYSPROC.HEALTH_HI_REC ストアード・プロシージャーは、以下の引数を取ります。

- ヘルス・インディケーター
- ヘルス・インディケーターがアラート状態になっているオブジェクトの定義

出力の推奨文書は BLOB として戻されます。したがって、CLP の場合は表示される出力の量が制限されるので、コマンド行からこのストアード・プロシージャーを処理しても効果的ではありません。戻された XML 文書を適切に解析してご希望のエレメントや属性を検索できる高水準言語 (C や Java など) を使用して、このストアード・プロシージャーを呼び出すことをお勧めします。

正常性を保つための推奨事項のコマンド・ライン・プロセッサーによる検索:

コマンド・ライン・プロセッサー (CLP) から GET RECOMMENDATIONS コマンドを使用して推奨事項を検索できます。このコマンド構文は、推奨事項を照会して、特定のオブジェクト上でアラート状態になっているヘルス・インディケーターなどの特定のヘルス・アラートを解決することをサポートしています。

始める前に

ヘルス・モニターから推奨事項を検索するには、インスタンス接続がなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンス上のヘルス・モニターから推奨事項を取得するには、まずそのインスタンスにアタッチする必要があります。ヘルス・モニターから推奨事項を検索するには、特殊権限は必要ありません。

このタスクについて

またこのコマンド構文は、特定のヘルス・インディケーターに関する推奨事項の完全集合の検索もサポートしています。このヘルス・インディケーターは、コマンド

の実行時にアラート状態になっている必要はありません。特定のヘルス・インディケーターに関するアラートを解決する際の推奨事項は、単一パーティション・レベルかグローバル・レベルのいずれかで照会できます。

特定のオブジェクト上のヘルス・アラートに関する推奨事項を照会すると、ヘルス・モニターは特定のアラートを解決しようとし、出力の問題のセクション中に解決しようとしたアラートに関する詳細情報を示すことができます。

またヘルス・モニターは、推奨事項のランキングを示したり、場合によってはアラートを解決するために実行できるスクリプトを生成することもできます。さらに、一部の推奨事項が特定の問題状態に該当しない場合に、ヘルス・モニターはそれらの推奨事項をリジェクトして非表示にできます。また、下記の最初の例のように、推奨事項がヘルス・インディケーター名のみで照会されると常に、考えられる一連の推奨事項がすべて戻されます。この場合、単に CLP コマンドは、アラートが示された場合に考慮すべきアクションに関する情報を示します。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

手順

GET RECOMMENDATIONS コマンドを使用して、推奨事項を取得します。

- 次のコマンドを実行して、 **db.db_op_status** ヘルス・インディケーターに関するアラートを解決する場合に推奨できるアクションの全集合を参照できます。

  ```
  db2 get recommendations for health indicator db.db_op_status
  ```

  この例では、 **db.db_op_status** ヘルス・インディケーターに関する推奨事項の全集合が戻されます。このコマンドを実行する際に、このヘルス・インディケーターはアラート状態になっている必要はありません。

  この出力は、このヘルス・インディケーターに関する推奨事項が 2 つ考えられることを示しています。 1 つはデータベースの活動化で、もう 1 つはデータベースに関するロールフォワード進行状況の調査です。このコマンドを使用すると、特定のアラートの解決方法が要求されるのではなく、考えられる推奨事項がすべて照会されるので、ヘルス・モニターはこの事例の最善の推奨事項を識別できません。その結果、推奨事項の全集合が戻されます。

  ```
  Recommendations:

  Recommendation: Investigate rollforward progress.

  A rollforward is in progress on the database due to an explicit
  request from the administrator. You have to wait for the rollforward
  to complete for the instance to return to active state.

  From the Command Line Processor, issue the commands shown in the
  following example to view the progress of the rollforward
  utility:

  LIST UTILITIES SHOW DETAIL
  ```

```
Recommendation: Unquiesce the database.

The database has been put into QUIESCE PENDING or QUIESCE state by
an explicit request from the administrator. If you have QUIESCE_CONNECT
authority, or are DBADM or SYSADM, you will still have access to the
database and will be able to use it normally. For all other users, new
connections to the database are not permitted and new units of work
cannot be started. Also, depending on the quiesce request, active
units of work will be allowed to complete or will be rolled back
immediately. You can issue an unquiesce to return to active state.

From the Command Line Processor, issue the commands shown in the
following example:

CONNECT TO DATABASE database-alias
UNQUIESCE DATABASE
```

- データベース SAMPLE のヘルス・インディケーター **db.db_heap_util** がアラート状態になっていることに気付き、アラートの解決方法を判別することにしたとします。この場合、特定の問題を解決することを望んでいるので、次の方法で GET RECOMMENDATIONS コマンドを発行できます。

  ```
  db2 get recommendations for health indicator db.db_heap_util
      for database on sample
  ```

  この出力は、問題のサマリーと、問題を解決するための推奨事項の集合を示しています。ヘルス・モニターが、優先順位に従って推奨事項をランク付けしました。個々の推奨事項には、説明とアクションの集合が含まれていて、推奨アクションを実行する方法が示されています。

  ```
  Problem:

      Indicator Name                          = db.db_heap_util
         Value                                = 42
         Evaluation timestamp                 = 11/25/2003 19:04:54
         Alert state                          = Alarm
         Additional information               =

  Recommendations:

  Recommendation: Increase the database heap size.
  Rank: 1

  Increase the database configuration parameter dbheap sufficiently
  to move utilization to normal operating levels. To increase the
  value, set the new value of dbheap to be equal to
  (pool_cur_size / (4096*U)) where U is the desired utilization rate.
  For example, if your desired utilization rate is 60% of the warning
  threshold level, which you have set at 75%, then
  U = 0.6 * 0.75 = 0.45 (or 45%).
  Execute the following commands at the DB2 server (this can be done using
  the EXEC_DB2_CMD stored procedure):

  CONNECT TO DATABASE SAMPLE;
  UPDATE DB CFG USING DBHEAP 149333;
  CONNECT_RESET;
  Recommendation: Investigate memory usage of database heap.
  Rank: 2

  There is one database heap per database and the database manager uses
  it on behalf of all applications connected to the database. The data
  area is expanded as needed up to the maximum specified by dbheap.

  For more information on the database heap, refer to the DB2
  ```

```
Information Center.

Investigate the amount of memory that was used for the database heap
over time to determine the most appropriate value for the database
heap configuration parameter. The database system monitor tracks the
highest amount of memory that was used for the database heap.
```

- パーティション・データベース・システムの場合、特定のパーティション上のアラート状態になったヘルス・インディケーターに関する推奨事項を照会することも、すべてのパーティションについてグローバルに照会することもできます。推奨事項をグローバルに照会すると、すべてのパーティション上のヘルス・インディケーターに適用される推奨事項の集合が戻されます。例えば、パーティション 1 と 3 上でヘルス・インディケーターがアラート状態になっている場合は、 2 つのスクリプトを収集したものが戻され、それぞれのスクリプトは別のパーティションに適用されます。

  次の例は、特定のパーティション (この例ではパーティション番号 2) 上のヘルス・インディケーターに関する推奨事項を照会する方法を示しています。

  ```
  db2 get recommendations for health indicator db.db_heap_util
    for database on sample at dbpartitionnum 2
  ```

  次の例は、複数のパーティション上でアラート状態になっているヘルス・インディケーターを解決するための推奨事項の集合を検索する方法を示しています。

  ```
  db2 get recommendations for health indicator db.db_heap_util
    for database on sample global
  ```

システムの正常性を保つための推奨事項をクライアント・アプリケーションを使用して検索:

C または C++ アプリケーションで db2GetRecommendations API を使用して、推奨事項を照会できます。

始める前に

ヘルス・スナップショットをキャプチャーするには、インスタンス接続がなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスに関する推奨事項を照会するには、まずそのインスタンスにアタッチする必要があります。

このタスクについて

db2GetRecommendations API を使用すると、推奨事項は次のような XML 文書で戻されます。

- SQLLIB ディレクトリー中の MISC サブディレクトリー中にある正常性の推奨事項の XML スキーマ DB2RecommendationSchema.xsd に従ってフォーマットされている。
- UTF-8 でエンコードされ、クライアントの言語のテキストが含まれている。
- 推奨事項の集合を収集したものとして編成されている。個々の推奨事項の集合には、解決しようとしている問題 (ヘルス・インディケーター) が記述され、そのヘルス・インディケーターを解決する際の 1 つ以上の推奨事項が含まれます。この文書から検索できる情報に関する特定の詳細情報については、スキーマ定義を参照してください。

CLP を使用して入手できる情報はすべて、戻される XML 推奨文書でも入手できます。

クライアント・アプリケーションを使用して正常性の推奨事項を検索するには、次のようにします。

手順

1. sqlmon.h および db2ApiDf.h DB2 ヘッダー・ファイルを組み込みます。これらのファイルは、 sqllib¥include ディレクトリーにあります。

   ```
   #include <db2ApiDf.h>
   #include <sqlmon.h>
   ```

2. sqlca および db2GetRecommendationsData 構造体を宣言します。

   ```
   struct sqlca sqlca ;
   db2GetRecommendationsData recData ;

   memset( &sqlca, '¥0', sizeof( struct sqlca ) ) ;
   memset( &recData, '¥0', sizeof( db2GetRecommendationsData ) ) ;
   ```

3. 推奨事項を検索するアラートに関する情報を、 db2GetRecommendationsData 構造体に取り込みます。次のコードの抜粋では、 Sample データベース上の **db2.db_heap_util** ヘルス・インディケーターに関する推奨事項が照会されます。

   ```
   recData.iSchemaVersion = DB2HEALTH_RECSCHEMA_VERSION8_2 ;
   recData.iNodeNumber = SQLM_CURRENT_NODE ;
   recData.iIndicatorID = SQLM_HI_DATABASE_HEAP_UTILIZATION  ;
   recData.iObjType = DB2HEALTH_OBJTYPE_DATABASE ;
   recData.piDbName = "SAMPLE" ;
   ```

4. db2GetRecommendations API を呼び出して、指定データベース上のこのヘルス・インディケーターのアラートに関する推奨事項を検索します。

   ```
   db2GetRecommendations(  db2Version820, &recData, &sqlca ) ;
   ```

5. sqlca 中に戻された sqlcode を検査して、発生したエラーをチェックします。 API 呼び出しが正常に実行された場合は、 db2GetRecommendationsData 構造体の poRecommendation フィールド中に戻された推奨事項の XML 文書を処理してください。 XML パーサーの選択項目を使用して、ご希望のエレメントまたは属性を抽出してください。 XML 文書から取り出せる情報に関する詳細は、 sqllib¥misc ディレクトリー中の DB2RecommendationSchema.xsd XML スキーマを参照してください。

6. db2GetRecommendations API によって割り振られたメモリーを解放します。こうすると、 db2GetRecommendationsData 構造体の poRecommendation フィールド中に戻された推奨事項の文書が解放されます。

   ```
   db2GetRecommendationsFree( db2Version820, &recData, &sqlca );
   ```

タスクの結果

普通は、ヘルス・インディケーターがアラート状態になっているのを検出した時点で推奨事項を照会するので、通常は上記のコードとスナップショット API に対する呼び出しを組み合わせて、ヘルス・スナップショットを取ります。

ヘルス・インディケーターの構成

デフォルトのヘルス・モニター構成は、インストール時に備えられます。この構成によって、DB2 の開始直後に、ヘルス・モニターがデータベース環境の正常性を評価できることが保証されます。

しかし、特定のユーザーの環境用の構成を使用して、ヘルス・モニターがヘルス・インディケーターを評価したりアラート状態に対応したりする動作を微調整できます。

ヘルス・モニター SQL 表関数は、推奨されなくなりました。

重要: ヘルス・モニター、ヘルス・インディケーター、および関連コンポーネントは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。ヘルス・モニターは、DB2 pureScale 環境ではサポートされていません。 詳しくは、『ヘルス・モニターが推奨されなくなった』のトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html) を参照してください。

構成を定義できるレベルは複数あります。 DB2 のインストール時に、ヘルス・インディケーターごとに工場出荷時設定のデフォルト構成が備えられます。初めてヘルス・モニターを開始する際に、工場出荷時設定のコピーにより、デフォルトのインスタンス設定とグローバル設定が備えられます。

インスタンス設定は、インスタンスに適用されます。グローバル設定は、インスタンス中のカスタマイズ設定が定義されていないデータベース、表スペース、および表スペース・コンテナーなどのオブジェクトに適用されます。

特定のデータベース、表スペース、または表スペース・コンテナーに関するヘルス・インディケーター設定を更新すると、更新されたヘルス・インディケーターのオブジェクト設定が作成されます。オブジェクト設定のデフォルトは、グローバル設定です。

ヘルス・モニターは、特定のデータベース、表スペース、表スペース・コンテナーに関するヘルス・インディケーターを処理する際に、オブジェクト設定を検査します。特定のヘルス・インディケーターの設定が一度も更新されていない場合は、デフォルトのグローバル設定を使用してヘルス・インディケーターが処理されます。ヘルス・モニターがインスタンスに関するヘルス・インディケーターを処理する際には、インスタンス設定が使用されます。

ヘルス・インディケーターごとに構成できる複数の属性を使用して、ヘルス・モニターの動作を変更できます。最初のパラメーターの集合 (評価フラグ、しきい値、感度) は、ヘルス・モニターがヘルス・インディケーターに関するアラートを生成する時点を定義します。 2 番目のパラメーターの集合 (アクション・フラグ、アクション) は、アラート生成時のヘルス・モニターの実行内容を定義します。

評価フラグ

個々のヘルス・インディケーターには、アラート状態の評価を使用可能にしたり使用不可にしたりする評価フラグがあります。

警告およびアラームのしきい値

しきい値ベースのヘルス・インディケーターには、ヘルス・インディケータ

一値の警告およびアラーム領域を定義する設定があります。特定のデータベース環境に合わせて、警告およびアラームのしきい値に変更を加えることができます。

感度パラメーター

感度パラメーターは、アラートが生成される前に、ヘルス・インディケーター値がアラート状態になっていなければならない期間の最小値を秒単位で定義します。感度値に関連した待機時間は、ヘルス・インディケーター値がアラート状態になった最初のリフレッシュ・インターバルの際に開始されます。この値を使用すると、リソースが一時的に使用できないだけでアラートが誤って生成されてしまうことを防止できます。

ログ使用率 (*db.log_util*) ヘルス・インディケーターを使用する例を考えてみましょう。週単位で DB2 通知ログを検討するとします。第 1 週に、*db.log_util* の項目はアラーム状態になっています。この状態に関する通知を受け取ったことを思い出しましたが、 CLP からアラート状態を検査すると、ヘルス・インディケーターは正常な状態に戻っていました。第 2 週の後に、週の同じ時点で同じヘルス・インディケーターに関する 2 度目のアラーム通知項目を受け取りました。アラートが生成された 2 つの事例に関してデータベース環境のアクティビティーを調査して、コミットに長時間を要するアプリケーションが毎週実行されていたことが判明しました。このアプリケーションは、アプリケーションがコミットするまでの短時間 (約 8 分から 9 分)、 ログ使用率の急上昇の原因となっています。通知ログ中のアラーム通知レコード中の履歴項目から、 *db.log_util* ヘルス・インディケーターが 10 分ごとに評価されていたことを判別できます。アラートが生成されているので、アプリケーション時間はこのリフレッシュ・インターバルをまたいでいるはずです。そこで、 *db.log_util* パラメーターの感度を 10 分に設定します。これで、 *db.log_util* の値が初めて警告またはアラームのしきい値の領域に入るたびに、アラートが生成されるにはその前にこの値が 10 分以上その領域に入り続けていなければなりません。アプリケーション時間は 8、9 分のみなので、この状態に関する通知項目が通知ログに記録されることはなくなります。

アクション・フラグ

アラート生成に関するアクションの実行は、アクション・フラグによって制御されます。アクション・フラグが有効になっている場合にのみ、構成済みのアラート・アクションが実行されます。

アクション

スクリプト・アクションかタスク・アクションを構成して、アラートの発生時に実行できます。しきい値ベースのヘルス・インディケーターの場合、警告またはアラームのしきい値に応じて実行するようにアクションを構成できます。状態ベースのヘルス・インディケーターの場合、通常以外のすべての条件に応じて実行するようにアクションを構成できます。アクションを実行するには、DB2 Administration Server を実行していなければなりません。

次の入力パラメーターが、すべてのオペレーティング・システムのコマンド・スクリプトに渡されます。

- <health indicator shortname>
- <object name>
- <value | state>

- <alert type>

  スクリプト・アクションは、オペレーティング・システム上のデフォルトのインタープリターを使用します。デフォルト以外のインタープリターを使用する場合は、 ADMIN_TASK_ADD プロシージャーをスクリプトの内容と一緒に使用してタスクを作成してください。複数のパーティションがある環境では、スクリプト・アクション中に定義されたスクリプトは、すべてのパーティションからアクセス可能でなければなりません。

ヘルス・モニターが個々のヘルス・インディケーターを検査するリフレッシュ・インターバルは構成できません。ヘルス・モニターによって考慮される推奨アクションも構成できません。

ヘルス・モニターの構成は、バイナリー・ファイル HealthRules.reg に保管されます。

- Windows では、HealthRules.reg は `x:¥<SQLLIB_PATH>¥<INSTANCE_NAME>` に保管されます。例えば `d:¥sqllib¥DB2` のようになります。
- UNIX では、 HealthRules.reg は `~/<SQLLIB_PATH>/cfg` 中に保管される。例えば `~/home/sqllib/cfg` のようになります。

ヘルス・モニターの構成を、Linux、UNIX、または Windows サーバー上の他の DB2 インスタンスに複製できます。このレプリケーションを行うには、このバイナリー構成ファイルを、ターゲット・インスタンス上の該当するディレクトリーにコピーします。

CLP を使用したヘルス・インディケーター構成の検索:

GET ALERT CONFIGURATION コマンドを使用すると、工場出荷時設定、インスタンス設定、グローバル設定、およびオブジェクト設定を表示できます。

手順

1. データベース・レベルのヘルス・インディケーターのグローバル設定を表示するには、次のコマンドを発行します。この設定は、ヘルス・インディケーターのカスタマイズ設定のないすべてのデータベースに適用されます。

   ```
   DB2 GET ALERT CONFIGURATION FOR DATABASES
   ```

2. データベース・レベルのヘルス・インディケーターのグローバル設定を表示するには、次のコマンドを発行します。この設定は、ヘルス・インディケーターのカスタマイズ設定のないすべてのデータベースに適用されます。

   ```
   DB2 GET ALERT CONFIGURATION FOR DATABASES
   ```

   個々のヘルス・インディケーターの設定の出力は、デフォルトから変更されたかどうかを示します。以下の出力では、グローバル設定は更新されていません。したがって、デフォルトの工場出荷時設定と同じです。データベース・レベルのヘルス・インディケーターの工場出荷時設定を表示するには、上記の例と同じコマンドに DEFAULT キーワードを指定して発行してください。

   ```
            Alert Configuration

   Indicator Name                   = db.db_op_status
       Default                      = Yes
       Type                         = State-based
       Sensitivity                  = 0
       Formula                      = db.db_status;
       Actions                      = Disabled
   ```

```
    Threshold or State checking    = Enabled

Indicator Name                     = db.sort_shrmem_util
    Default                        = Yes
    Type                           = Threshold-based
    Warning                        = 70
    Alarm                          = 85
    Unit                           = %
    Sensitivity                    = 0
    Formula                        = ((db.sort_shrheap_allocated/sheapthres_shr)
                                      *100);
    Actions                        = Disabled
    Threshold or State checking    = Enabled
...
```

3. SAMPLE データベースのカスタム設定を表示するには、次のコマンドを発行してください。

   ```
   DB2 GET ALERT CONFIGURATION FOR DATABASE ON SAMPLE
   ```

   指定したオブジェクト上に特定のヘルス・インディケーターに関する固有の設定がない場合は、すべてのデータベースのグローバル設定が表示されます。特定のヘルス・インディケーターの設定を表示するには、前述の例に USING *health-indicator-name* 節を追加してください。

CLP を使用したヘルス・インディケーター構成の更新:

特定のヘルス・インディケーターに関するヘルス・インディケーター構成中で、グローバル設定または特定のオブジェクトのオブジェクト設定を更新できます。

UPDATE ALERT CONFIGURATION コマンドには、さまざまな更新オプションに対応した 4 つの副節があります。個々の UPDATE ALERT CONFIGURATION コマンド中で使用できる副節は 1 つのみです。複数のオプションを使用するには、複数の UPDATE ALERT CONFIGURATION コマンドを発行しなければなりません。

1 つ目の副節 SET *parameter-name value* は、次のものの更新をサポートしています。

- 評価フラグ
- 警告およびアラームしきい値 (該当する場合)
- 感度フラグ
- アクション・フラグ

これらの設定に対応するパラメーター名は、以下のとおりです。

- THRESHOLDSCHECKED
- WARNING および ALARM
- SENSITIVITY
- ACTIONSENABLED

他の 3 つの副節は、スクリプト・アクションやタスク・アクションの追加、更新、および削除をサポートしています。

次のコマンドは、 SAMPLE データベース上の *db.spilled_sorts* ヘルス・インディケーターに関するしきい値ベースのヘルス・インディケーター構成を更新します。この更新により、警告しきい値が 25 に変更され、アクションが使用可能になり、スクリプト・アクションが追加されます。

```
DB2 UPDATE ALERT CONFIGURATION FOR DATABASE ON SAMPLE USING DB.SPILLED_SORTS
    SET WARNING 25, ACTIONSENABLED YES
DB2 UPDATE ALERT CONFIGURATION FOR DATABASE ON SAMPLE USING DB.SPILLED_SORTS
    ADD ACTION SCRIPT c:¥myscript TYPE OS COMMAND LINE PARAMETERS 'space'
    WORKING DIRECTORY c:¥ ON ALARM USER dba1 PASSWORD dba1
```

次のコマンドは、 *ts.ts_util* ヘルス・インディケーターに関する状態ベースのヘルス・インディケーター構成中のグローバル設定を更新します。この更新により、表スペースがバックアップ・ペンディング状態にある際に実行するアクションが定義されます。

```
DB2 UPDATE ALERT CONFIGURATION FOR TABLESPACES USING TS.TS_UTIL
    SET ACTIONSENABLED YES
DB2 UPDATE ALERT CONFIGURATION FOR TABLESPACES USING TS.TS_UTIL
    ADD ACTION TASK 0.1 ON ATTENTION 32 ON localhost USER dba1 PASSWORD dba1
```

この更新は、このヘルス・インディケーターのカスタマイズ設定のないインスタンスの表スペースすべてに適用されます。

ヘルス・インディケーター構成にアクションを追加する場合、 ON *condition* 節のオプションは、ヘルス・インディケーターのタイプに基づいて異なります。

- しきい値ベースのヘルス・インディケーターの場合、 WARNING および ALARM が有効な条件。
- 状態ベースのヘルス・インディケーターの場合、 ON ATTENTION *state* オプションを使用する必要があります。定義済みの、ヘルス・インディケーターにとって有効な数値の状態を使用する必要があります。データベース・マネージャーとデータベースの操作可能状態の値は、sqllib¥include¥sqlmon.h 中にあります。表スペースと表スペース・コンテナーの操作可能値は、sqllib¥include¥sqlutil.h 中にリストされています。データベース・マネージャーが停止状態にある場合にはアクションを実行することができません。詳しくは、db2.db2_op_status ヘルス・インディケーターの説明を参照してください。

CLP を使用したヘルス・インディケーター構成のリセット:

CLP は、グローバル設定を工場出荷時設定にリセットすることをサポートしています。特定のオブジェクトのオブジェクト設定を、そのオブジェクト・タイプのカスタム設定にリセットすることもできます。

手順

- SAMPLE データベースのオブジェクト設定をデータベースの現行のグローバル設定にリセットするには、以下のようにします。

  ```
  DB2 RESET ALERT CONFIGURATION FOR DATABASE ON SAMPLE
  ```

- データベースのグローバル設定を工場出荷時設定にリセットするには、次のコマンドを発行してください。

  ```
  DB2 RESET ALERT CONFIGURATION FOR DATABASES
  ```

- 特定のヘルス・インディケーターの構成をリセットするには、前述の例に USING *health-indicator-name* 節を追加してください。

クライアント・アプリケーションを使用したヘルス・インディケーターの構成:

ヘルス・モニターの構成は、C または C++ アプリケーション中の db2GetAlertCfg、 db2UpdateAlertCfg、および db2ResetAlertCfg API によってアクセ

スできます。これらの各 API は、工場出荷時設定、インスタンス設定、グローバル設定、およびオブジェクト設定にアクセスできます。

始める前に

ヘルス・モニター構成にアクセスするには、インスタンス接続がなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのヘルス・モニター構成にアクセスするには、まずそのインスタンスにアタッチする必要があります。

このタスクについて

db2GetAlertCfgData 構造中の **objType** パラメーターと **defaultType** パラメーターを組み合わせて使用すると、さまざまなレベルのヘルス・インディケーター構成にアクセスできます。

表 144. objType および defaultType が構成レベルにアクセスする際の設定

設定	objType および defaultType
工場出荷時設定	objType = DB2ALERTCFG_OBJTYPE_{DBM | DATABASES |TABLESPACES | CONTAINERS} および defaultType = DB2ALERTCFG_DEFAULT
グローバル設定	objType = DB2ALERTCFG_OBJTYPE_{DBM | DATABASES | TABLESPACES | CONTAINERS} and defaultType = DB2ALERTCFG_NOT_DEFAULT または objType = DB2ALERTCFG_OBJTYPE_{DATABASE | TABLESPACE | CONTAINER} および defaultType = DB2ALERTCFG_DEFAULT
オブジェクト設定	objType = DB2ALERTCFG_OBJTYPE_{DATABASE | TABLESPACE | CONTAINER} および defaultType = DB2ALERTCFG_NOT_DEFAULT

手順

1. SAMPLE データベース上のヘルス・インディケーターに関する特定のオブジェクト設定を取得するには、次のようにします。
   a. sqllib¥include ディレクトリー中にある db2ApiDf.h DB2 ヘッダー・ファイルを組み込みます。

      ```
      #include <db2ApiDf.h>
      ```

   b. sqlca および db2GetAlertCfgData 構造体を宣言して初期化します。

      ```
      struct sqlca ca;
      memset (&sqlca, '¥0', sizeof(struct sqlca));

      char* objName = NULL;
      char* dbName = "SAMPLE";
      db2Uint32 objType = DB2ALERTCFG_OBJTYPE_DATABASE;
      db2Uint32 defaultType = DB2ALERTCFG_NOT_DEFAULT;

      db2GetAlertCfgData data = {objType, objName, defaultType, dbName, 0, NULL} ;
      ```

   c. db2GetAlertCfg API を呼び出します。

      ```
      rc = db2GetAlertCfg (db2Version810, &data, &ca);
      ```

d. 戻された構成を処理し、API によって割り当てられたバッファーを解放します。

```
if (rc >= SQLO_OK) {
  if ((data.ioNumIndicators > 0) && (data.pioIndicators != NULL)) {
    db2GetAlertCfgInd *pIndicators = data.pioIndicators;

    for (db2Uint32 i=0; i  data.ioNumIndicators; i++) {
 //process the entry as necessary using fields defined in db2ApiDf.h
    }
  }

  db2GetAlertCfgFree (db2Version810, &data, &ca);
}
```

2. 次のステップは、 **db.sort_shrmem_util** ヘルス・インディケーターのアラート構成中で、データベース・オブジェクトのグローバル設定を更新し、警告しきい値を 80 に設定してタスク・アクション 1.1 を追加する手順を詳述しています。

   a. sqllib¥include ディレクトリー中にある db2ApiDf.h DB2 ヘッダー・ファイルを組み込みます。

   ```
   #include <db2ApiDf.h>
   ```

   b. sqlca および db2AlertTaskAction 構造体を宣言して初期化します。

   ```
   struct sqlca ca;
   memset (&sqlca, '¥0', sizeof(struct sqlca));

   db2Uint32 objType = DB2ALERTCFG_OBJTYPE_DATABASES;

   db2Uint32 taskCondition = DB2ALERTCFG_CONDITION_WARNING;
   char* taskname = "1.1";
   char* hostname = NULL;
   char* userid = "nobody";
   char* password = "nothing";

   db2AlertTaskAction newTask={taskname,taskCondition,userid,password,hostname};
   ```

   c. db2UpdateAlertCfgData 構造体を宣言して初期化します。

   ```
   struct db2UpdateAlertCfgData setData;

   setData.iObjType = objType;
   setData.piObjName = NULL;
   setData.piDbName = NULL;

   setData.iIndicatorID = 1002;

   setData.iNumIndAttribUpdates = 1;
   setData.piIndAttribUpdates[0].iAttribID = DB2ALERTCFG_WARNING;
   setData.piIndAttribUpdates[0].piAttribValue == 80;

   setData.iNumActionUpdates = 0;
   setData.piActionUpdates = NULL;

   setData.iNumActionDeletes = 0;
   setData.piActionDeletes = NULL;

   setData.iNumNewActions = 1;
   setData.piNewActions[0].iActionType = DB2ALERTCFG_ACTIONTYPE_TASK;
   setData.piNewActions[0].piScriptAttribs = NULL;
   setData.piNewActions[0].piTaskAttribs = &newTask;
   ```

   d. db2UpdateAlertCfg API を呼び出します。

   ```
   rc = db2UpdateAlertCfg(db2Version810, &setData, &ca);
   ```

3. 次のステップは、 SAMPLE データベース中の MYTS 表スペースのカスタム設定をリセットする手順を詳述しています。

   a. sqllib¥include ディレクトリー中にある db2ApiDf.h DB2 ヘッダー・ファイルを組み込みます。

   ```
   #include <db2ApiDf.h>
   ```

b. sqlca および db2ResetAlertCfgData 構造体を宣言して初期化します。

```
struct sqlca ca;
memset (&sqlca, '¥0', sizeof(struct sqlca));

char* objName = "MYTS";
char* dbName = "SAMPLE";
db2Uint32 objType = DB2ALERTCFG_OBJTYPE_TABLESPACE;

db2ResetAlertCfgData data = {objType, objName, dbName};
```

c. db2ResetAlertCfg を呼び出します。

```
rc = db2ResetAlertCfg (db2Version810, &data, &ca);
```

組み合わせの状態に対するヘルス・モニターのアラート・アクション:

アラート・アクションは、ヘルス・インディケーターがアラート状態に入るときに実行されるタスクまたはスクリプトです。

DB2 V9.1 以降、ヘルス・インディケーター **ts.ts_op_status** に定義されるヘルス・モニターのアラート・アクションが 1 つのアラート状態にある場合、その他の組み合わせの状態に関係なく、表スペースにこの状態が設定されるといつでも実行されます。これにより、他の状態と一緒に設定されていても、特定の表スペースの状態に対してアラート・アクションを実行することが可能になります。

以下の例の場合、アテンション状態 QUIESCED:share に対して定義されているアラート・アクション script1 は、表スペース状態が同時に QUIESCED:share と QUIESCE:update になる場合でも実行されます。

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    set actionsenabled yes')

db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    add action script /home/guest001/script1 type operating system command line
    parameters userParam working directory /home/guest001/
    on attention QUIESCED_SHARE on aix1 user guest001 using passw0rd')
```

以下の例の場合、状態の組み合わせ (QUIESCED:share + QUIESCED:update = 3) によって定義されているアラート・アクションは、表スペースの状態が QUIESCED:share と QUIESCED:update の両方になっている場合に限って実行されます。

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    set actionsenabled yes')

db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    add action script /home/guest001/script1 type operating system command line
    parameters userParam working directory /home/guest001/
    on attention 3 on aix1 user guest001 using passw0rd')
```

DB2 V9.1 以降、同じアクション属性 (名前、作業ディレクトリー、コマンド行パラメーター、ホスト、ユーザーおよびパスワード) を持つオブジェクトに対して定義されたヘルス・モニターのアラート・アクションは、複数のアラート状態に対して定義された場合でも一度しか実行されません。

以下の例では、2 つの別々のアラート状態に対して同じアクションが定義されています。このアクションは、表スペース状態が QUIESCED:share と QUIESCED:update の両方になっている場合でも、特定の表スペースに対して一度しか実行されません。

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    add action script /home/guest001/script1 type operating system command line
    parameters userParam working directory /home/guest001/
    on attention QUIESCED_SHARE on aix1 user guest001 using passw0rd')

db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    add action script /home/guest001/script1 type operating system command line
    parameters userParam working directory /home/guest001/
    on attention QUIESCED_UPDATE on aix1 user guest001 using passw0rd')
```

Windows Management Instrumentation (WMI) の紹介

管理インフラストラクチャーの規格を制定し、各種のハードウェアおよびソフトウェア管理システムの情報を結合するための方法を提供する、業界イニシアチブが存在します。このイニシアチブは、Web-Based Enterprise Management (WBEM) と呼ばれています。

WBEM は、Desktop Management Task Force (DMTF) 主導の業界標準である Common Information Model (CIM) スキーマを基にしています。

Microsoft Windows Management Instrumentation (WMI) は、サポートされる Windows プラットフォーム用に WBEM イニシアチブを実現したものです。 WMI は、Windows エンタープライズ・ネットワークで役立ちます。これにより、エンタープライズ・ネットワーク・コンポーネントの保守と管理費用が削減されます。 WMI は以下を提供します。

- Windows の操作、構成、および状況の一貫性のあるモデル。
- 管理情報にアクセスできるようにする COM API。
- 他の Windows 管理サービスと協働する機能。
- 柔軟で拡張可能なアーキテクチャー。これにより、ベンダーは、新しい装置、アプリケーション、その他の機能強化をサポートするための他の WMI プロバイダーを作成できます。
- 情報の詳細な照会を作成するための WMI Query Language (WQL)。
- 管理アプリケーション開発者が Visual Basic または Windows Scripting Host (WSH) スクリプトを作成するための API。

WMI アーキテクチャーには、以下の 2 つの部分があります。

1. 管理インフラストラクチャー。これには、CIM Object Manager (CIMOM) と、CIMOM オブジェクト・リポジトリーと呼ばれる管理データ用の中央ストレージ域が含まれています。 CIMOM を使用すると、アプリケーションは一様な方法で管理データにアクセスできるようになります。
2. WMI プロバイダー。 WMI プロバイダーは、CIMOM と管理下のオブジェクトを仲介するコンポーネントです。 WMI API を使用することにより、WMI プロバイダーは、管理下のオブジェクトのデータを CIMOM に提供し、管理アプリケーションの代わりに要求を処理し、イベント通知を生成します。

Windows Management Instrumentation (WMI) プロバイダーは、管理下のオブジェクトと CIM Object Manager (CIMOM) の仲介機能としての役割を果たす、標準の COM または DCOM サーバーです。 CIMOM が、CIMOM オブジェクト・リポジトリーからは使用できないデータ (またはイベント) に対する要求を管理アプリケー

ションから受け取ると、 CIMOM はその要求を WMI プロバイダーに転送します。WMI プロバイダーは、管理対象オブジェクトに対して、それぞれのドメインに固有のデータとイベント通知を提供します。

DB2 データベース・システムと Windows Management Instrumentation の統合

Windows Management Instrumentation (WMI) は、 DB2 パフォーマンス・カウンターを使って、また組み込み PerfMon プロバイダーを使用してスナップショット・モニターにアクセスできます。

WMI は、組み込みレジストリー・プロバイダーを使用して、 DB2 プロファイル・レジストリー変数にアクセスできます。

WMI Software Development Kit (WMI SDK) には、以下の複数の組み込みプロバイダーが含まれています。

- PerfMon プロバイダー
- レジストリー・イベント・プロバイダー
- レジストリー・プロバイダー
- Windows イベント・ログ・プロバイダー
- Win32 プロバイダー
- WDM プロバイダー

WMI は、組み込み Windows イベント・ログ・プロバイダーを使用して、イベント・ログ内の DB2 エラーにアクセスできます。

DB2 データベース・システムには、以下の管理対象オブジェクトにアクセスするための、 DB2 WMI 管理プロバイダーとサンプルの WMI スクリプト・ファイルがあります。

1. 分散インスタンスを含むデータベース・サーバーのインスタンス。以下の操作を実行できます。
   - インスタンスの列挙
   - データベース・マネージャー・パラメーターの構成
   - DB2 サーバー・サービスの開始/停止/状況照会
   - 通信のセットアップまたは確立
2. データベース。以下の操作を実行できます。
   - データベースの列挙
   - データベース・パラメーターの構成
   - データベースの作成/ドロップ
   - データベースのバックアップ/リストア/ロールフォワード

WMI アプリケーションを実行する前に、 DB2 WMI プロバイダーをシステムに登録する必要があります。以下のコマンドを入力して登録を行います。

- mofcomp %DB2PATH%¥bin¥db2wmi.mof

  このコマンドは、DB2 WMI スキーマの定義をシステムにロードします。

- regsvr %DB2PATH%¥bin¥db2wmi.dll

  このコマンドは、DB2 WMI プロバイダー COM DLL を Windows に登録します。

両方のコマンドで、%DB2PATH% は DB2 のインストール先のパスです。また、db2wmi.mof は DB2 WMI スキーマ定義が入っている .MOF ファイルです。

WMI インフラストラクチャーを統合することには、以下のような複数の利点があります。

1. WMI 提供のツールを使用して、 Windows ベースの環境で DB2 サーバーを管理するためのスクリプトを簡単に作成できます。インスタンスのリスト、データベースの作成とドロップ、構成パラメーターの更新などの単純なタスクを実行するための、サンプルの Visual Basic (VBS) スクリプトが提供されています。サンプル・スクリプトは、DB2 Application Development for Windows 製品に組み込まれています。
2. WMI を使用して多くのタスクを実行する強力な管理アプリケーションを作成できます。タスクには以下のものがあります。
   - システム情報の表示
   - DB2 パフォーマンスのモニター
   - DB2 システム・リソース使用量のモニター

   このタイプの管理アプリケーションを使って、システム・イベントと DB2 イベントの両方をモニターすることにより、データベースをよりよく管理できます。
3. 既存の COM および Visual Basic プログラミングの知識とスキルを活用できます。 COM または Visual Basic インターフェースにより、プログラマーは、エンタープライズ管理アプリケーションの開発時間を短縮できます。

Windows プラットフォームでのパフォーマンス・モニター

Windows 用の DB2 データベース・マネージャーを使用する場合は、パフォーマンスのモニターに使用できるツールがあります。

重要: Windows パフォーマンス・モニターは推奨されていません。今後のリリースで廃止される可能性があります。バージョン 10.1 では、IBM InfoSphere Optim Performance Manager を代わりに使用してください。

- **Windows Performance Monitor**

  Windows パフォーマンス・モニターを使用すると、データベースとシステム・パフォーマンスの両方をモニターでき、そのシステムに登録されている任意のパフォーマンス・データ提供元から情報を取り出すことができます。さらに Windows は、以下を含めたコンピューター操作のすべてについて、パフォーマンス情報データを提供します。

  – CPU 使用率
  – メモリー使用率
  – ディスクの活動状況
  – ネットワークの活動状況

Windows パフォーマンス・モニターへの DB2 の登録

セットアップ・プログラムは、 DB2 を自動的に Windows パフォーマンス・モニターへ登録します。 Windows パフォーマンス・モニターを使用し、 DB2 データベースおよび DB2 Connect™ のパフォーマンス情報にアクセスできるようにするには、DB2 (Windows 版) のパフォーマンス・カウンター用の DLL を登録する必要があります。

このタスクについて

また、このプロセスによって、Win32 パフォーマンス API を使用する Windows アプリケーションが他にあれば、そのアプリケーションもパフォーマンス・データを入手できるようになります。 DB2 パフォーマンス・カウンターの DLL (DB2Perf.DLL) を、Windows パフォーマンス・モニターにインストールして登録するには、次のように入力します。

```
db2perfi -i
```

DLL を登録するならば、レジストリーのサービス・オプションで、新しいキーを作成することもできます。 1 つの項目には DLL の名前が示され、カウンターをサポートします。他の 3 つの項目には、その DLL に備えられている機能の名前が示されます。それらの機能は、以下のとおりです。

オープン
処理中に、DLL がシステムによって初めてロードされるときに呼び出されます。

収集 DLL からのパフォーマンス情報を要求するときに呼び出されます。

クローズ
DLL をアンロードするときに呼び出されます。

DB2 パフォーマンス情報へのリモート・アクセスを使用可能にする

別の DB2 (Windows 版) コンピューターから Windows パフォーマンス・オブジェクトを見るには、 DB2 データベース・マネージャーに管理者のユーザー名とパスワードを登録しなければなりません。デフォルトの Windows パフォーマンス・モニターのユーザー名である SYSTEM は、 DB2 データベースの予約語なので使用できません。

このタスクについて

ご使用の DB2 (Windows 版) ワークステーションが、別の Windows コンピューターにネットワークで接続されている場合、この節で説明されているフィーチャーを使用できます。

名前を登録するには、次のように入力します。

```
db2perfr -r username password
```

注: 使用する username は、 DB2 データベース命名規則に適合しなければなりません。

ユーザー名とパスワードのデータは、レジストリー内のキーに置かれます。このときのセキュリティーは、管理者および SYSTEM アカウントだけがアクセスできる

というものです。管理者のパスワードをレジストリーに格納する際のセキュリティー上の問題を防ぐため、データはエンコードされます。

注:

1. いったんユーザー名とパスワードの組み合わせを DB2 データベース・システムに登録すれば、パフォーマンス・モニターのローカル・インスタンスであっても、そのユーザー名とパスワードを使って明示的にログオンします。つまり、DB2 データベース・システムに登録されたユーザー名情報が一致しなければ、パフォーマンス・モニターのローカル・セッションには、 DB2 データベースのパフォーマンス情報が示されないことになります。
2. ユーザー名とパスワードの組み合わせは、 Windows のセキュリティー・データベースに格納されているユーザー名とパスワードと常に一致させる必要があります。Windows セキュリティー・データベース内のユーザー名かパスワードを変更した場合、リモートのパフォーマンス・モニターに使うユーザー名とパスワードの組み合わせを再設定しなければなりません。
3. 登録するには、次のように入力します。

```
db2perfr -u <username> <password>
```

DB2 データベースと DB2 Connect のパフォーマンス値を表示する

保守を実施している、またはデータベース全体のパフォーマンスを改善しようとしている場合、現在のパフォーマンス値を確認すると役立ちます。

このタスクについて

パフォーマンス・モニターを使って DB2 データベースおよび DB2 Connect のパフォーマンス値を表示するには、「**追加先**」ボックスから、表示させる値を示すパフォーマンス・カウンターを選択します。このボックスには、パフォーマンス・データを提示するパフォーマンス・オブジェクトのリストが示されます。提供されているカウンターのリストを見るには、特定のオブジェクトを選択してください。

1 つのパフォーマンス・オブジェクトに、複数のインスタンスが存在することもあります。例えば、LogicalDisk オブジェクトには、「% Disk Read Time」や「Disk Bytes/sec」などのカウンターが備えられています。さらに、コンピューター内の論理ドライブ (「C:」や「D:」など) ごとに、1 つのインスタンスがあります。

Windows パフォーマンス・オブジェクト

DB2 が Windows オペレーティング・システムにインストールされると、提供されるパフォーマンス・モニター・オブジェクトが多数あります。

Windows には、以下のパフォーマンス・オブジェクトがあります。

- **DB2 データベース・マネージャー**

  このオブジェクトは、1 つの Windows インスタンスのための、一般的な情報を提供します。モニターされる DB2 データベース・インスタンスは、オブジェクト・インスタンスとして表されます。

  実用上ならびにパフォーマンス上の理由のため、パフォーマンス情報は、一度に 1 つの DB2 データベース・インスタンスだけから入手されます。パフォーマンス・モニターが示す DB2 データベース・インスタンスは、パフォーマンス・モ

ニターの処理では、 db2instance レジストリー変数によって管理されます。同時に複数の DB2 データベース・インスタンスを実行していて、 2 つ以上のパフォーマンス情報を確認する場合、パフォーマンス・モニターのセッションを個別に開始する必要があります。このとき、db2instance には、モニターする DB2 データベース・インスタンスごとに対応する値を設定します。

パーティション・データベース環境を実行している場合は、一度に 1 つのデータベース・パーティション・サーバーからのみ、パフォーマンス情報を入手できます。デフォルトでは、デフォルト・データベース・パーティション (論理ポートが 0 のデータベース・パーティション) のパフォーマンス情報が表示されます。他のデータベース・パーティションのパフォーマンス情報を表示するには、DB2NODE 環境変数を、モニターするデータベース・パーティションのデータベース・パーティション番号に設定して、パフォーマンス・モニターの他のセッションを開始する必要があります。

- **DB2 データベース**

  このオブジェクトは、特定のデータベースの情報を提供します。現在アクティブなデータベースごとに、情報を利用できます。

- **DB2 アプリケーション**

  このオブジェクトは、特定の DB2 データベース・アプリケーションの情報を提供します。現在アクティブな DB2 データベース・アプリケーションごとに、情報を利用できます。

- **DB2 DCS データベース**

  このオブジェクトは、特定の DCS データベースの情報を提供します。現在アクティブなデータベースごとに、情報を利用できます。

- **DB2 DCS アプリケーション**

  このオブジェクトは、特定の DB2 DCS アプリケーションの情報を提供します。現在アクティブな DB2 DCS アプリケーションごとに、情報を利用できます。

Windows パフォーマンス・モニターによってリストされるオブジェクトは、Windows コンピューターに何がインストールされているか、およびどのアプリケーションがアクティブかによって異なります。例えば、 DB2 データベース・マネージャーをインストールし開始していれば、 DB2 データベース・マネージャー・オブジェクトがリストされます。さらに、そのコンピューターで現在アクティブな DB2 データベースおよびアプリケーションがあれば、 DB2 データベースおよび DB2 アプリケーション・オブジェクトもリストされます。Windows システムを DB2 Connect のゲートウェイとして使っていて、現在アクティブな DCS データベースおよびアプリケーションがいくつかある場合、 DB2 DCS データベースおよび DB2 DCS アプリケーション・オブジェクトがリストされます。

リモートの DB2 データベースのパフォーマンス情報へのアクセス

データベースがリモート・ロケーションにある場合は、別のロケーションからパフォーマンス情報にアクセスして分析することもできます。

このタスクについて

DB2 パフォーマンス情報にリモートでアクセスできるようにする方法については、すでに説明しました。「**追加先**」ボックスで、モニターする別のコンピューターを選択してください。これにより、そのコンピューター上で使用できるすべてのパフォーマンス・オブジェクトのリストが表示されます。

リモート・コンピューターで DB2 パフォーマンス・オブジェクトをモニターできるようにするには、そのコンピューターにインストールされている DB2 データベースまたは DB2 Connect コードのレベルが、バージョン 6 以上でなければなりません。

DB2 パフォーマンス値をリセットする

DB2 データベース・システムに格納されているパフォーマンス値をリセットすることが必要な状況が幾つかあります。

このタスクについて

アプリケーションで DB2 モニター API を呼び出すと、戻される情報は、通常は DB2 データベース・サーバー開始以降の累積値になります。これは、以下のような場合に役立ちます。

- パフォーマンス値をリセットする
- テストを実行する
- その値をもう一度リセットする
- テストを再実行する

データベースのパフォーマンス値をリセットするには、 **db2perfc** プログラムを使用します。次のように入力してください。

```
db2perfc
```

デフォルトでは、これによりアクティブな DB2 データベースすべてのパフォーマンス値がリセットされます。ただし、リセットするデータベースのリストを指定することも可能です。また -d オプションを使用して、 DCS データベースのパフォーマンス値をリセットするよう指定することもできます。例えば、

```
db2perfc
    db2perfc dbalias1 dbalias2 ... dbaliasn

    db2perfc  -d
    db2perfc  -d  dbalias1 dbalias2 ... dbaliasn
```

最初の例では、すべてのアクティブな DB2 データベースのパフォーマンス値がリセットされます。次の例では、特定の DB2 データベースの値がリセットされます。3 番目の例では、すべてのアクティブな DB2 DCS データベースのパフォーマンス値がリセットされます。最後の例では、特定の DB2 DCS データベースの値がリセットされます。

db2perfc プログラムは、関連する DB2 データベース・サーバー・インスタンス (つまり、 **db2perfc** 実行時のセッションの DB2INSTANCE に保持されるインスタンス) に関するデータベース・パフォーマンス情報に現在アクセスしているすべてのプログラムの値をリセットします。

db2perfc を呼び出すと、 **db2perfc** コマンドの実行中に DB2 データベースのパフォーマンス情報にリモートでアクセスしているユーザーがいる場合、そのユーザーに表示される値もリセットされます。

注: sqlmrset という DB2 データベース API を使用すれば、アプリケーションでグローバルではなくローカルに表示される特定のデータベースの値を、アプリケーション側でリセットできます。

第 2 部 モニター・エレメント

モニター・エレメントは、データベース・システムの状態の特定の側面を示す情報を保管するために使用されるデータ構造です。例えば、モニター・エレメント **direct_reads** は、発生した読み取り操作のうち、バッファー・プールからではなくディスクから直接実行された読み取り操作の数を示します。

各モニター・エレメントは、次のいずれかのタイプのデータを示します。

カウンター
カウンターは、何かの発生回数を追跡します。例えば、 **deadlocks** モニター・エレメントは、発生したデッドロックの総数を記録します。カウンターの他の例には、**commit_sql_stmts (試行された COMMIT ステートメント数)**、**rows_deleted**、**total_sorts** などがあります。

ゲージ ゲージは、何かの発生量または使用量の測定値を示します。例えば、**total_section_proc_time** や **total_sort_time** などの消費時間モニター・エレメントは、さまざまな処理フェーズに所要された時間の測定です。ゲージの他の例には、**locks_held**、**num_extent_moved**、 **sort_heap_allocated** などがあります。時間の経過とともに増加する一方のカウンターと比べて、ゲージの値は、データベースで何が発生するかによって、増加したり減少したりすることがあります。

水準点 水準点は、特定の測定において達した最高値を示します。例えば、**uow_total_time_top** は、データベースの活動化以降、最も長く実行された作業単位の存続時間を示します。水準点の他の例には、**pkg_cache_size_top** や **sort_heap_top** などがあります。

テキスト
多くのモニター・エレメントはテキスト値をレポートします。例えば、**stmt_text** には、 SQL ステートメントのテキストが入っています。テキスト・モニター・エレメントの他の例には、 **table_name**、**tablespace_type**、**db_storage_path_state** などがあります。

タイム・スタンプ
タイム・スタンプ・モニター・エレメントは、何かの発生時刻を示します。例えば、**conn_time** は、データベースに接続が確立された時刻を示します。タイム・スタンプ・モニター・エレメントの他の例には、**lock_wait_start_time**、**stmt_first_use_time**、 **uow_stop_time** などがあります。経過時間を測定するゲージと比べて、タイム・スタンプは、何かが開始または終了した時点を正確に測定します。

モニター・エレメントは、DB2 製品に用意されている表関数やイベント・モニターなどのさまざまなモニター・インターフェースを 1 つ以上使用して調べることができます。

第 6 章 要求モニター・エレメント

要求モニター・エレメントは、*要求メトリック* とも呼ばれ、さまざまなタイプの要求を処理するために、データベース・サーバーにより費やされた作業量または処理動作を測定します。これにはシステム全体の処理、特定のタイプの処理に関係した要求、および特定のデータ・サーバー環境に関係した要求が含まれます。

データベース・システム (特に、アプリケーション要求を処理するためにデータ・サーバーが費やした作業量および労力) をモニターするには、要求モニター・エレメントを使用します。

要求 とは、データベース・リソースを消費する作業を実行するための、データベース・エージェントに対する命令です。要求のソースとしては、以下のようなものがあります。

- 外部アプリケーションにより直接出される命令。OPEN または EXECUTE 命令など。これらはアプリケーション要求と呼ばれます。
- コーディネーター・エージェントによって、同じまたは異なるデータベース・メンバーのサブエージェントに出される命令。
- 異なるデータベース・メンバーのエージェントにより出される命令。

システム全体の処理情報を測定するためのいくつかの代表的なモニター・エレメントには、以下のものがあります。

- **rqsts_completed_total** モニター・エレメントは、システムにより完了された数を測定します。
- **total_rqst_time** モニター・エレメントは、待機時間と処理時間を含む、データ・サーバーの要求により費やされた時間を測定します。
- **total_wait_time** モニター・エレメントは、待機時間全体を測定します。
- **total_cpu_time** モニター・エレメントは、CPU 使用時間を測定します。

クライアント・サーバーの処理情報を測定するためのいくつかの代表的なモニター・エレメントには、以下のものがあります。

- **client_idle_wait_time** モニター・エレメントは、オープン接続からの次の要求を待機するために費やされた時間を測定します。
- **tcpip_recv_volume** モニター・エレメントは、データ・サーバーがクライアントから TCP/IP 経由で受信したデータの量を測定します。

共通データ・サーバー処理操作を測定するためのいくつかの代表的なモニター・エレメントには、以下のものがあります。

- **pool_data_l_reads** は、バッファー・プール・リソース使用に関する情報を提供するモニター・エレメントの 1 つです。
- **pool_read_time** は、入出力処理に関する情報を提供するモニター・エレメントの 1 つです。
- **lock_wait_time** は、ロックおよび現在進行中のロックに関する情報を提供するモニター・エレメントの 1 つです。

- **total_section_sorts** は、ソートに関する情報を提供するモニター・エレメントの 1 つです。

選択したタイプのデータ・サーバー環境に関係したモニター処理のためのいくつかの代表的なモニター・エレメントには、以下のものがあります。

- **fcm_recv_wait_time** は、高速コミュニケーション・マネージャー (FCM) 処理を測定するモニター・エレメントの 1 つです。
- **wlm_queue_time_total** は、ワークロード管理制御アクションを測定するモニター・エレメントの 1 つです。

表関数を使用した要求メトリックへのアクセス

要求メトリックにアクセスするために、以下の表関数を使用することができます。

- MON_GET_SERVICE_SUBCLASS および MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_WORKLOAD および MON_GET_WORKLOAD_DETAILS
- MON_GET_CONNECTION および MON_GET_CONNECTION_DETAILS
- MON_GET_UNIT_OF_WORK および MON_GET_UNIT_OF_WORK_DETAILS

モニター表関数のこの一連の各表関数には 2 つの形式があり、その 1 つは名前の末尾に「DETAILS」が付きます。「DETAILS」が末尾に付かない関数は、最も一般的に必要とされるデータを返す SQL リレーショナル・インターフェースを提供します。それ以外の関数は、モニター・データへの XML ベースのアクセスを提供し、さらに総合的なデータ集合を返します。

この一連の表関数により、特定の集約レベルの要求メトリックに焦点を当てることができます。特定の状況で、関心対象のシステム・ワークロードのサブセット (または集約) に焦点を当てることのできる表関数を選択できます。これらのすべての表関数には、要求メトリック・モニター・エレメントの共通セットが組み込まれています。それぞれの表関数は、いくつかの追加の詳細を返す場合があり、それらはすべての表関数に共通というわけではありません。

ユーザー定義のワークロードまたはサービス・クラスがないデータベースでは、データベース・マネージャーによって実行されるすべてのユーザーの作業は、デフォルトのユーザー・ワークロードおよびユーザー・サービス・クラスで実行されます。各サービス・クラス (またはワークロード) にデータを返す表関数は、データベース全体のユーザー・ワークロードの処理を表す、単一のサービス・クラス (またはワークロード) のデータを返します。

ユーザー定義ワークロードおよびサービス・クラスがあるデータベースでは、各サービス・クラス (またはワークロード) のデータを返す表関数により、サービス・クラス (またはワークロード) 単位で処理を比較することができます。 SQL を使用すると、サービス・クラス (またはワークロード) 全体の値を合計して、データベース全体のユーザー・ワークロードの処理を表すモニター・エレメントの値を取得することができます。

イベント・モニターを使用した要求メトリックへのアクセス

要求メトリックは、以下のイベント・モニターにより報告されます。

- 統計イベント・モニター - 要求メトリックは、このイベント・モニターにより報告されるいくつかのタイプの情報の 1 つです。
- UoW イベント・モニター - このイベント・モニターは、MON_GET_UNIT_OF_WORK 表関数と類似のまたは同じフィールドを報告します。

アクティビティー・モニター・エレメント

アクティビティー・モニター・エレメント (別名アクティビティー・メトリック) は、要求モニター・エレメントのサブセットです。アクティビティー・メトリックを使用して、アクティビティーの実行 (特に SQL ステートメント・セクションの実行のための処理) に関係した、データ・サーバー処理のサブセットをモニターします。

要求モニター・エレメントは、アプリケーション要求を処理するためにデータ・サーバーにより費やされた作業量および労力全体をモニターします。アクティビティー・モニター・エレメントは、ロック、ソート、および行処理を含む SQL ステートメント・セクションを実行するために行われる作業をモニターします。

アクティビティー・モニター・エレメントの現行値にアクセスするには、以下の表関数を使用します。

MON_GET_ACTIVITY_DETAILS
進行中の 1 つ以上のアクティビティーに関する詳細を返します。入力パラメーターに関心対象のアクティビティーを指定します。返されるデータには、アクティビティー・メトリック・モニター・エレメント、他の多くのモニター・エレメント、およびステートメント・テキストが含まれます。データは XML 形式で返されます。

MON_GET_PKG_CACHE_STMT
データベース・パッケージ・キャッシュ内の一部またはすべての SQL ステートメント・セクションの詳細を返します。これには静的および動的 SQL ステートメントの両方が含まれます。返されるデータには、パッケージ・キャッシュに追加されてからの、セクションのすべての実行を対象として集約されたアクティビティー・メトリック・モニター・エレメントが含まれます。データはリレーショナル形式で返されます。

アクティビティー・イベント・モニターを使用して、アクティビティーに関する履歴データにアクセスします。このモニターは、それぞれのアクティビティーの各実行データをキャプチャーします。アクティビティー・イベント・モニターは、MON_GET_ACTIVITY_DETAILS 表関数と同じアクティビティー・モニター・エレメントをキャプチャーします。これは、一部の追加情報もキャプチャーします。

第 7 章 データ・オブジェクトに関するモニター・エレメント

データ・オブジェクト・モニター・エレメントは、表、索引、バッファー・プール、表スペース、およびコンテナーを含む、特定のデータ・オブジェクトに対して実行された操作に関する情報を提供します。

すべてのデータ・オブジェクト・タイプには、モニター可能な一連のモニター・エレメントがあります。例えば、バッファー・プールには、バッファー・プール・ヒット率を計算するために使用できるエレメントがあります。

以下の表関数を使用して、データ・オブジェクト・モニター・エレメントの現行値にアクセスします。これらのモニター表関数は、リレーショナル形式でデータを返します。

- MON_GET_BUFFERPOOL
- MON_GET_TABLESPACE
- MON_GET_CONTAINER
- MON_GET_TABLE
- MON_GET_INDEX

第 8 章 モニター・エレメントの収集レベル

モニター・エレメントの収集レベル とは、そのエレメントのデータを収集するためにアクティブでなければならない設定がもしあれば、そのような設定を指します。

多くのモニター・エレメントでは、構成パラメーター、ワークロード管理オブジェクトの定義に使用する DDL 内の節、またはその両方の組み合わせによって、データ収集を制御します。

収集レベルの設定

データベース構成パラメーターを使用して収集レベルを設定すると、特定のクラスのモニター・エレメントに関するデフォルトの収集レベルが、データベース全体に対して設定されます。例えば、構成パラメーター **mon_req_metrics** を BASE に設定すると、データベースで実行されるすべてのエージェントに関して、要求メトリックが収集されることになります。要求メトリックとアクティビティー・メトリック、およびセクション実行時統計モニター・エレメントについては、データベース全体に対して使用するレベルとは別の収集レベルを、特定の WLM オブジェクトに対して指定することもできます。このように、特定のモニター・エレメントの*有効収集レベル* は、エレメントを収集する各範囲において、そのエレメントに指定されている最大範囲 (最高位) の収集レベルを適用することによって決定されます。複数の収集範囲がある場合の処理動作の例については、654 ページの『例』のセクションを参照してください。

ほとんどのモニター・エレメントを説明するトピックに、収集レベルを記載しています。 (スナップショット・インターフェースから戻されるモニター・エレメントの場合は、収集レベルではなくモニター・スイッチを記載しています。) 例として、表 145 に、モニター・エレメント **skipped_prefetch_data_p_reads** を戻すインターフェースと、このエレメントのデータを収集するためにアクティブである必要がある「モニター・エレメントの収集レベル」を一緒に示す表を示します。

ほとんどのトピックには、そのトピックで説明するモニター・エレメントに関して、そのモニター・エレメントを戻すインターフェースと、そのデータ収集に必要な最低収集レベルを示す、このような表が含まれています。

表 145. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

モニター・エレメントのリファレンス・トピックに記載している収集レベルは、以下のとおりです。

常に収集される
このモニター・エレメントのデータは常に収集されます。この情報の収集を制御するための構成パラメーターも SQL ステートメント・オプションも存在しません。

DATA OBJECT METRICS BASE、DATA OBJECT METRICS EXTENDED
この収集レベルのモニター・エレメントは、データベース構成パラメーター **mon_obj_metrics** が BASE または EXTENDED に設定された場合に収集されます。 **mon_obj_metrics** が NONE に設定されると、データは収集されません。

REQUEST METRICS BASE、REQUEST METRICS EXTENDED
この収集レベルのモニター・エレメントは、有効収集レベルが BASE または EXTENDED に設定された場合に収集されます。要求メトリックの有効収集レベルは、データベース構成パラメーター **mon_req_metrics** の現行設定値と、WLM サービス・スーパークラスの COLLECT REQUEST METRICS 節に指定された設定値を検査した上で決定されます。

ACTIVITY METRICS BASE、ACTIVITY METRICS EXTENDED
この収集レベルのモニター・エレメントは、有効収集レベルが BASE または EXTENDED に設定された場合に収集されます。アクティビティー・メトリックの有効収集レベルは、データベース構成パラメーター **mon_act_metrics** の現行設定値と、WLM ワークロードの COLLECT ACTIVITY METRICS 節に指定された設定値を検査した上で決定されます。

SECTION ACTUALS BASE
この収集レベルのモニター・エレメントは、有効収集レベルが BASE に設定された場合に収集されます。セクション実行時統計の有効収集レベルは、データベース構成パラメーター **section_actuals** の現行設定値と、以下の項目の設定値を検査した上で決定されます。

- CREATE または ALTER WORKLOAD、CREATE または ALTER WORK ACTION SET、あるいは CREATE または ALTER SERVICE CLASS ステートメントの一部としての INCLUDE ACTUALS 節。
- WLM_SET_CONN_ENV ルーチンの <collectsectionactuals> の設定値。

COLLECT AGGREGATE ACTIVITY DATA、COLLECT AGGREGATE REQUEST DATA
この収集レベルのモニター・エレメントが収集されるのは、節 COLLECT AGGREGATE ACTIVITY DATA または COLLECT AGGREGATE REQUEST DATA が、特定のタイプの WLM オブジェクトの CREATE または ALTER ステートメントの一部として含まれている場合です。

一部のエレメントには、「適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する」という記述があります。この場合は、既に収集済みのイベント・データの出力を単にフォーマットする関数から、エレメントが戻されます。

例

例 1: データベース全体に対してはデフォルトの収集レベル NONE を、特定のサービス・スーパークラスに対しては EXTENDED を指定する。
この例では、データベース全体に対しては (ここでは要求メトリック・モニ

ター・エレメントについて) 収集レベル NONE を指定しますが、あるサービス・スーパークラスで実行されるエージェントに関しては EXTENDED メトリックを収集する方法を示します。

データベースで実行されるエージェント全体に関して要求メトリックの収集を無効にするが、特定のサービス・スーパークラスに関しては拡張メトリックを収集する場合、以下の手順を実行します。

1. 以下のコマンドを実行して、データベース全体に対して要求メトリックの収集レベルを NONE に設定します。

   ```
   DB2 UPDATE DB CFG FOR database-name USING MON_REQ_METRICS NONE
   ```

2. 以下のステートメントを実行して、要求メトリックを収集するサービス・スーパークラスへの変更を行います。

   ```
   ALTER SERVICE CLASS service-class-name COLLECT REQUEST METRICS EXTENDED
   ```

結果: *service-class-name* という名前のサービス・スーパークラスで実行されるすべてのエージェントに関して、要求メトリックが収集されます。このサービス・スーパークラス以外で実行されるエージェントに関しては、要求メトリックは収集されません。

例 2: データベース全体に対してデフォルトの収集レベル EXTENDED を指定する。 この例では、最大範囲の収集レベルが、モニター・エレメント・データの収集に適用されたために、意図したであろう結果とは異なる結果となるケースを示します。

- 以下のコマンドを使用して、データベースで実行されるすべてのアクティビティーに関して、アクティビティー・メトリックをキャプチャーするように指定します。

  ```
  DB2 UPDATE DB CFG FOR database-name USING MON_ACT_METRICS EXTENDED
  ```

- アクティビティー・メトリックを収集しないように、ある WLM ワークロードを変更します。

  ```
  ALTER WORKLOAD workload-name COLLECT ACTIVITY METRICS NONE
  ```

結果: ワークロード *workload-name* の一部として実行されるエージェントを含め、データベースで実行されるすべてのエージェントに関してアクティビティー・メトリックが収集されます。この場合、有効収集レベルは、**mon_act_metrics** 構成パラメーターによって、データベース全体に対して指定された広い方の範囲の収集レベル (EXTENDED) に決定されます。

第 9 章 消費時間モニター・エレメント

消費時間モニター・エレメントは、システム内で時間がどのように費やされているかを追跡します。これらを照会すると、待機や、さまざまな種類の処理の実行にどのように時間が使われているかを確認できます。また、特定のシステム・コンポーネントで費やされた経過時間を表示することもできます。

図 11 は、要求の処理時間と比較して、待機に費やされた時間を表示する 1 つの方法を例示しています。

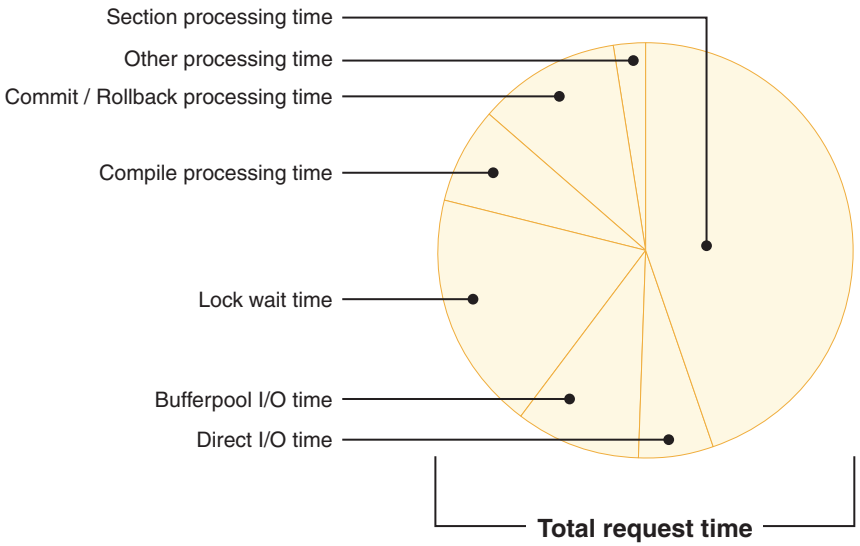


図 11. システムでどのように時間が費やされたかの概要を示す消費時間メトリック： 時間は、待機に費やされた時間 (ロック待機時間、バッファー・プール入出力時間、直接入出力時間)、および実際の処理時間に分割されます。

システムで費やされる時間をデータベース・マネージャーがモニターする方法には、以下の 3 つがあります。

- 待機時間
- コンポーネント処理時間
- コンポーネント経過時間

待機時間

待機時間モニター・エレメントは、データベース・マネージャーが処理を進める前に何らかの待機に費やした時間を反映します。例えば、次のサービスの待機に費やした時間が含まれます。

- クライアント要求の着信
- オブジェクトのロックの解除
- 診断ログへの書き込み
- バッファー・プールとの間の読み取り/書き込み

待機時間を追跡するモニター・エレメントには、例えば **lock_wait_time**、**pool_read_time** があります。

コンポーネント処理時間

これらの時間は、データベースの特定の論理コンポーネントにおいて実際の処理に費やされた時間を表します。例えば、次のようなサービスの実行に費やした時間が含まれます。

- トランザクションのコミットまたはロールバック
- データベース再編成の実行
- SQL のコンパイル
- データのロード
- RUNSTATS 操作の実行

コンポーネント処理時間を追跡するモニター・エレメントには、例えば **total_compile_proc_time**、**total_commit_proc_time** があります。

コンポーネント経過時間

コンポーネント経過時間は、データベースの 1 つの論理コンポーネントで費やされた経過時間の合計を反映します。これには、処理時間と、その処理段階で全般的に発生する可能性のあるさまざまな種類の待機時間の*両方*が含まれます。例えば、コミットの実行に費やされる時間の合計には、実際のコミット処理が含まれるのに加えて、さまざまな種類の待機時間 (入出力操作やログ・ファイル操作の完了を待つために発生した時間など) も含まれる可能性があります。

注: 経過時間は、時計で測定される経過時間と同じではありません。費やされた合計時間が複数のスレッドに分割された場合、各スレッドで費やされた時間がこの数値に表されます。

例えば、次のようにしてコンポーネント時間を活用できます。

- 特定のワークロードにおいて、比較的高い処理コストが発生している部分を把握する (例えば、照会実行と比較した SQL コンパイル)
- 特定のコンポーネント領域のコストは実際の処理に起因しているのか、スループット低下の大きな要因は待機時間であるのか、などを判別する
- システムでの全体的な消費時間のコンテキストにおいて、特定のコンポーネント領域 (例えばロールバック処理) のコストを把握する

全体的なコンポーネント時間を追跡するモニター・エレメントには、例えば **total_compile_time**、**total_commit_time** があります。

処理時間と比べた特定の待機時間を詳しく調べるために、コンポーネント処理時間と待機時間を照会することができます。 657 ページの図 11 は、この 2 種類の消費時間メトリックを互いに比較して表示する方法を例示しています。

特定の種類の待機時間 (例えばロック待機、入出力関連の待機など) を詳しく把握するためにコンポーネント経過時間を使用することはできませんが、この代替的なビューを使用すると、特定の論理データベース・コンポーネントで消費された合計時間と処理時間を比べて確認できます。例えば、表または索引の再編成に関する実際の処理時間 (**total_reorg_proc_time**) と、再編成の実行に費やされた経過時間全体

(**total_reorg_time**) を対比して調べることができます。後者には、再編成そのものには直接関係のない他のさまざまな処理/待機に費やされた時間が含まれる可能性があります。

消費時間モニター・エレメントの階層

多くの消費時間モニター・エレメントの情報は、より包括的なモニター・エレメントにロールアップされます。

例えば、表キューから次のバッファーの情報を受け取るための待機に費やした時間を表すエレメント (**fcm_tq_recv_wait_time**) や、 FCM 応答メッセージの待機に費やした時間を表すエレメント (**fcm_message_recv_wait_time**) などの個々の待機時間エレメントは、両方とも包括的な **fcm_recv_wait_time** エレメントの中に含まれます。消費時間モニター・エレメントの階層は、ユーザーの必要を満たす最も適切な詳細度レベルでエレメントを選択できるように編成されています。

消費時間モニター・エレメントを表示するためのディメンションとパースペクティブ

さまざまな方法で、消費時間モニター・エレメントの階層を表示することができます。 1 つの方法として、全体的なシステムの視点から調べることができます。さらに、システム内の特定のアクティビティーのコンテキストで調べることもできます。

システム・レベルのビュー (またはシステム・ディメンション) に含まれるエレメントは、システムの全体的な動作を確認するのに役立ちます。また、システム・ディメンション内のエレメントを使って、特定のワークロードの消費時間情報を確認することもできます。

アクティビティー・レベルのビューまたはアクティビティー・ディメンションに含まれるエレメントは、 SQL ステートメントの実行など、システム時間が費やされている特定のアクティビティーを確認するのに役立ちます。アクティビティー・ディメンション内のすべてのモニター・エレメントは、上位のシステム・ディメンションに含まれています。

この 2 つの各ディメンションに含まれる以下のような 2 つの異なるパースペクティブ を使用すると、消費時間モニター・エレメントを調べることができます。

- 待機時間と比較したコンポーネント処理時間
- コンポーネント処理時間と比較したコンポーネント経過時間

最初のパースペクティブでは、待機時間エレメントの値はコンポーネント処理時間エレメントの値から独立しており、補完的です。報告されるすべての待機時間の合計とすべてのコンポーネント処理時間の合計を合わせた値は、**total_rqst_time** モニター・エレメントで報告される値に極めて近くなります。この 2 つの値の間にわずかな差異が存在する場合、それはどのモニター・エレメントによっても追跡されていない他のコンポーネント処理時間が少しだけ存在するためです。

2 番目のパースペクティブでは、コンポーネント経過時間はコンポーネント処理時間のスーパーセットです。例えば、コミットを実行するコンポーネントのようなデータベース論理コンポーネントの場合、 **total_commit_proc_time** モニター・エレ

メントで報告されるコミットの合計処理時間は、 **total_commit_time** モニター・エレメントで報告されるコミットの合計経過時間に含まれます。この合計経過時間と合計処理時間の差異は、コンポーネント経過時間モニター・エレメントによって個別に追跡されることがない他の待機時間または処理時間の合計です。

待機時間と比較したコンポーネント経過時間を表示することには意味がありません。その理由は、システムのその部分の経過時間の一部として発生した待機時間は、コンポーネント経過時間の中に既に含まれるためです。仮にコンポーネント経過時間と待機時間の両方から成る円グラフを作成した場合、さまざまな種類の待機時間を二重にカウントしていることになるため、システムで費やされた時間を正確に表すものではありません。

以下のセクションでは、消費時間モニター・エレメントの情報を確認できるさまざまなディメンション (システムおよびアクティビティー) およびパースペクティブ (コンポーネント処理時間/待機時間、コンポーネント経過時間/コンポーネント処理時間) について説明します。

ヒント: すべてのインターフェースですべての消費時間エレメントの情報が報告されるわけではありません。例えば、**client_idle_wait_time** モニター・エレメントは、MON_GET_SERVICE_SUBCLASS 表関数などのシステム・レベルのインターフェースにのみ当てはまります。各モニター・エレメントを報告するインターフェースのリストについては、そのエレメントに関する参照トピックをご覧ください。

- 『システムのディメンション』
- 663 ページの『アクティビティー・ディメンション』

システムのディメンション

661 ページの図 12 は、システム・ディメンションから見た、待機時間およびコンポーネント処理時間モニター・エレメントの相互関連の概要を示しています。

- 897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』
- 1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』
  - 1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』
    - 829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』
    - 1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』
    - 1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』
    - 1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』
    - 1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』
    - 1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』
    - 1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』
    - 1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』
    - 1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』
    - 1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』[1]
      - 1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』[1]
      - 1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』[1]
    - 1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』[1]
      - 1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』[1]
      - 1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』[1]
    - 862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』
    - 858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』
    - 984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』
    - 1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』
    - 1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』
    - 989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』
    - 996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』
    - 1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』
    - 1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』
    - 1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』
    - 905 ページの『comm_exit_wait_time - 通信バッファー出口待機時間のモニター・エレメント』
    - 1092 ページの『ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント』
    - 1087 ページの『ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント』
  - 1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』
    - 1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』
    - 1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』
    - *他のモニター・エレメント*[2]
  - 1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』
  - 1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』
  - 1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』
    - 1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[2]
  - 1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』
  - 1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』
  - 1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』
  - 1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』
  - 1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』
  - 1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』
    - 1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』
    - *他のモニター・エレメント*[2]
  - *他のモニター・エレメント*[3]

[1]FCM に関連するこれらの待ち時間を複数のメンバーで集計した場合、意味のある情報は生成されません。詳しくは、『FCM 通信の待ち時間』を参照してください。

[2]これらの処理時間エレメントは、このコンポーネントに特に関連していない、その他の処理時間と待機時間を含みます。 例えば、 **total_section_proc_time** エレメントが報告する時間には、 **total_section_proc_sort_time** エレメントが報告する時間が含まれます。 **total_section_sort_proc_time** エレメントは、表スキャン、索引スキャン、結合などのアクティビティーの実行に費やした時間を報告します。

[3] これらのモニター・エレメントは、現在モニターされていない、その他の種類の消費時間 (処理時間と待機時間) を少数含みます。

図 12. 待機の消費時間およびコンポーネント処理の消費時間モニター・エレメント - システム・ディメンション： 字下げで示されているモニター・エレメントの値は、そのエレメントの前にある、階層内のそれより 1 つ高いレベルのエレメントに含まれています。

663 ページの図 13 は、さまざまなコンポーネント領域での消費時間に関するモニター・エレメントを詳しく示しています。各コンポーネント時間は、次のような 2 つの異なるモニター・エレメントで表されます。

- 1 つのコンポーネントまたは処理段階での処理時間の合計を報告するモニター・エレメント。
- そのコンポーネントで費やされた経過時間の合計を報告するモニター・エレメント。この合計時間には、コンポーネントの処理時間に加えて、関連するその他の処理時間または待機時間が含まれます (存在する場合)。

- 1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』
  - 1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』
    - 1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』
      - 1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』
    - 1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』
      - 1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』
    - *他のモニター・エレメント*[1]
  - 1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』
    - 1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[1]
  - 1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』
    - 1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』
  - 1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』
    - 1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』
      - 1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』
      - *他のモニター・エレメント*[1]
    - *他のモニター・エレメント*[1]
  - 1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』
    - 1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[1]
  - 1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』
    - 1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[1]
  - 1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』
    - 1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[1]
  - 1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』
    - 1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[1]
  - 1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』
    - 1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[1]
  - 1623 ページの『total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント』
    - 1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』
    - *他のモニター・エレメント*[1]
  - 1620 ページの『total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント』
    - 1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』
    - *他のモニター・エレメント*[1]
  - *他のモニター・エレメント*[2]

[1]このモニター・エレメントは、1 つ以上の種類の待機時間を含みます。

[2]これらのモニター・エレメントは、現在モニターされていない、その他の種類の消費時間 (処理時間と待機時間) を少数含みます。

図 13. コンポーネント処理の消費時間モニター・エレメント - システム・ディメンション： 字下げで示されているモニター・エレメントの値は、そのエレメントの前にある、階層内のそれより 1 つ高いレベルのエレメントに含まれています。

アクティビティー・ディメンション

664 ページの図 14 は、コンポーネント処理時間と比べた待機時間のパースペクティブから見た、アクティビティーに関する表示可能なモニター・エレメントを示して

います。

- 1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』
  - 1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』 [1]
    - 1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』
    - 1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』
    - 1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』
    - 1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』[2]
      - 1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』[2]
      - 1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』[2]
    - 1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』[2]
      - 1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』[2]
      - 1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』[2]
    - 862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』
    - 1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』
    - 858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』
    - 984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』
    - 1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』
    - 1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』
    - 989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』
    - 996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』
    - 1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』
    - 1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』
    - 1092 ページの『ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント』
    - 1087 ページの『ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント』
  - 1657 ページの『total_routine_non_sect_proc_time - 非セクション処理時間 : モニター・エレメント』
    - 1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[3]
  - 1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』
    - 1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』
    - *他のモニター・エレメント*[3]
  - *他のモニター・エレメント*[4]

[1]このモニター・エレメントに、ステートメントによって実行されたネスト (子) アクティビティーにより発生した待機時間は含まれません。

[2]FCM に関連するこれらの待ち時間を複数のメンバーで集計した場合、意味のある情報は生成されません。詳しくは、『FCM 通信の待ち時間』を参照してください。

[3]このコンポーネントに特に関連していない、その他の処理時間を含みます。

[4]現在モニターされていない、少数のその他の種類の消費時間 (処理時間と待機時間) を含みます。また、この時間には子アクティビティーによって発生した処理時間と待機時間も含まれます。

図 14. 待機の消費時間およびコンポーネント処理の消費時間モニター・エレメント - アクティビティー・ディメンション： 字下げで示されているモニター・エレメントの値は、そのエレメントの前にある、階層内のそれより 1 つ高いレベルのエレメントに含まれています。

図 15 は、(コンポーネント処理時間を含む) コンポーネント経過時間のパースペクティブから見た、アクティビティーに関する表示可能なモニター・エレメントを示しています。

- 1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』
  - 1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』
    - 1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』
      - 1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』
      - *その他*[1]
    - *その他*[2]
  - 1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』
    - 1658 ページの『total_routine_non_sect_time - 非セクション・ルーチンの実行時間 : モニター・エレメント』
      - 1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』
      - *その他*[2]
    - *その他*[2]

[1]このモニター・エレメントは、1 つ以上の種類の待機時間を含みます。

[2]これらの処理時間エレメントは、このコンポーネントに特に関連していない、その他の処理時間と待機時間を含みます。

図 15. コンポーネント経過時間およびコンポーネント処理時間のモニター・エレメント - アクティビティー・ディメンション : 字下げで示されているモニター・エレメントの値は、そのエレメントの前にある、階層内のそれより 1 つ高いレベルのエレメントに含まれています。

FCM 通信の待ち時間

複数パーティション・データベース、またはパーティション内並列処理がある環境では、同じステートメントを処理するさまざまなエージェント間の通信は高速コミュニケーション・マネージャー (FCM) によって管理されます (これらのエージェントが同じメンバー上にある場合も、異なるメンバー上にある場合も)。

どんな FCM 通信についても、エージェントが別のエージェントによる処理の完了を待機したり、エージェント間でのデータ転送の完了を待機したりするなどの場合に、待ち時間が発生する可能性があります。

FCM 関連の待ち時間は、メンバー間で処理がブロックされたことを必ずしも示すものではありません。あるステートメントに関して、複数メンバーにわたるサブエージェントで並列的または逐次的に処理が進められる可能性があります。FCM 関連の待ち時間は、1 つのメンバー上のエージェントが別のメンバーを待機してブロックされる時間を示します。しかし、相手のメンバー上では処理が順調に進行している可能性があります。

例えば、メンバー 0 上のエージェント A が、メンバー 1 上のエージェント B にデータが送られてそこで読み取られるのを待機し、ブロックされているとします。仮にエージェント B がビジー状態で、表キューからデータを直ちに受信できない場合、エージェント A は限定的な量のデータだけを送信した後、エージェント B から確認応答が届くのを強制的に待機させられます (確認応答の後で残りのデータを送信します)。この待ち時間はエージェント A では **fcm_tq_send_wait_time** として

カウントされます。

図 16. FCM 通信での待ち時間

別のシナリオとして、あるメンバー上のエージェントが、別のメンバー上のエージェントに要求を送っているとします。以下のいずれかの状況では、**fcm_message_recv_wait_time** が発生します。

- エージェント A が長い要求をエージェント B に送り、エージェント B は要求全体が受信されるまで強制的に待機させられる。この場合、エージェント B では **fcm_message_recv_wait_time** が発生します。
- エージェント A がエージェント B に要求を送り、エージェント B からの応答を待機する。この場合、エージェント A で **fcm_message_recv_wait_time** が発生します。

以下のいずれかの状況では、**fcm_message_send_wait_time** が発生します。

- エージェント A が長い要求をエージェント B に送り、何らかの理由でブロックされる。例えば、エージェント A は、送られる要求の最初の部分がローカル FCM デーモンによって処理されるのを待つ必要があるとします。この場合、エージェント A では **fcm_message_send_wait_time** が発生します。
- エージェント B が、エージェント A からの要求の応答を送る。メッセージ全体が送信される前に何らかの理由でエージェント B がブロックされた場合、エージェント B で **fcm_message_send_wait_time** が発生します。

計測する対象によっては、複数のパーティションにわたる消費時間メトリックを集計するときに、消費された合計時間から FCM 待ち時間を差し引くのが適切な場合があります。

消費時間モニター・エレメント・データの取得と操作

消費時間モニター・エレメントのデータを活用する方法は、ほとんど無限にあります。例えば、システムで時間がどのように費やされているかを一目で分かるように示す図表の生成を自動化することができます。

または、ある期間にわたってシステム内の特定の種類の待機時間を追跡するためにデータを活用することもできます。

この後のトピックでは、消費時間モニター・エレメントの使用法を示すいくつかの基本的な例と、それらに含まれるデータを利用するための表関数について説明します。

システム全体のどこで時間が費やされているかを調べる

消費時間モニター・エレメントを使用して、システムのどこで時間が費やされているかを可視化することができます。消費時間モニター・エレメントを使用して、特定の作業単位、サービス・サブクラス、ワークロード、または接続について報告できます。

このタスクについて

システムのどこで時間が費やされているかを報告する各種モニター・エレメントを検索したら、それらをさまざまに表示することができます。最も基本的なレベルでは、報告された値をリストとして表示できます。例えばロック待機時間の合計要求

時間に対する比率など、値を使用して比率を算出することもできます。あるいは、検索した値を使用してグラフを作成することにより、消費時間モニター・エレメント間の相対比較を視覚化できます。

注:

- 照会の出力で示される値は例示のみを目的としており、ご使用のシステムで表示される可能性のあるものを代表していると解釈すべきではありません。
- このタスクでは、特定の消費時間モニター・エレメントを検索する方法を示します。バージョン 9.7 フィックスパック 1 で導入された新しいフォーマット関数を使用することにより、特定の基準を満たす消費時間モニター・エレメントを検索することもできます。例えば、値がゼロ以外のもの、指定した値の一定範囲に入っているもの、または上位 *n* 個のモニター・エレメント (例えば上位 5 つの待機時間) などを検索できます。これらの関数の機能を、例 4 で示しています。

手順

1. まず、関心のある消費時間エレメントを決定します。 例えば、システムのすべての接続を対象に、合計待機時間を合計要求時間と比較して調べるとします。
2. 関心のあるエレメントを検索するモニター表関数の 1 つを使用して、SQL 照会を作成します。 この場合は、次のように MON_GET_CONNECTION 表関数を使用することによって、接続の **total_request_time** および **total_wait_time** モニター・エレメントを検索できます。

```
SELECT APPLICATION_HANDLE,
       TOTAL_WAIT_TIME,
       TOTAL_RQST_TIME
FROM TABLE(MON_GET_CONNECTION(NULL,NULL))
```

   上記の照会は、次のような出力を戻します (時間はミリ秒単位で報告されます)。

```
APPLICATION_HANDLE   TOTAL_WAIT_TIME      TOTAL_RQST_TIME
-------------------- -------------------- --------------------
                  39                  179                  269
                  78                    0                    0
                  51                  207                  316
                  77                    0                   21
                  50                 1014                 1408
                  40                  109                  351
                  79                   89                  167

  7 record(s) selected.
```

3. この場合は、7 つのアプリケーション接続があります。2 番目と 3 番目の列の結果を使用すれば、待機に費やした時間のパーセンテージをアプリケーションごとに割り出すことができます。 例えばアプリケーション 50 の場合、合計要求時間に対するこのアプリケーションが費やした待機時間のパーセンテージは、(1014 ÷ 1408) × 100 ≈ 72% になります。

例

例 1: 全体要求時間に対する待機に費やした時間の全接続平均比率を割り出す。

この例は前のものと似ていますが、待機時間の平均パーセンテージの計算が以下の SQL 内で行われます。

```
WITH PCTWAIT AS (
     SELECT SUM(TOTAL_WAIT_TIME)AS WAIT_TIME,
            SUM(TOTAL_RQST_TIME)AS RQST_TIME
     FROM TABLE(MON_GET_CONNECTION(NULL,NULL)) AS METRICS)
     SELECT WAIT_TIME,
            RQST_TIME,
     CASE WHEN RQST_TIME > 0
     THEN DEC((FLOAT(WAIT_TIME))/FLOAT(RQST_TIME) * 100,5,2)
     ELSE NULL END AS WAIT_PCT FROM PCTWAIT
```

この照会の実行結果は、次のようなものになります。

```
WAIT_TIME            RQST_TIME            WAIT_PCT
-------------------- -------------------- --------
                1515                 2439    62.11

  1 record(s) selected.
```

例 2: 特定のサービス・サブクラスについて、合計待機時間と選択したコンポーネント処理時間を比較する。

この例は、特定のタイプのコンポーネント処理で費やされた時間と待機に費やされた時間を比較する方法を示しています。

```
SELECT SUM(TOTAL_WAIT_TIME) AS WAIT,
       SUM(TOTAL_COMPILE_PROC_TIME) AS COMPILE,
       SUM(TOTAL_IMPLICIT_COMPILE_PROC_TIME) AS IMP_COMPILE,
       SUM(TOTAL_SECTION_PROC_TIME) AS SECTION,
       SUM(TOTAL_COMMIT_PROC_TIME) AS COMMIT,
       SUM(TOTAL_REORG_PROC_TIME) AS REORG,
       SUM(TOTAL_RUNSTATS_PROC_TIME) AS RUNSTATS,
       SUM(TOTAL_ROLLBACK_PROC_TIME) AS ROLLBACK,
       SUM(TOTAL_LOAD_PROC_TIME) AS LOAD
 FROM TABLE(MON_GET_SERVICE_SUBCLASS( 'SYSDEFAULTUSERCLASS','SYSDEFAULTSUBCLASS',NULL))
```

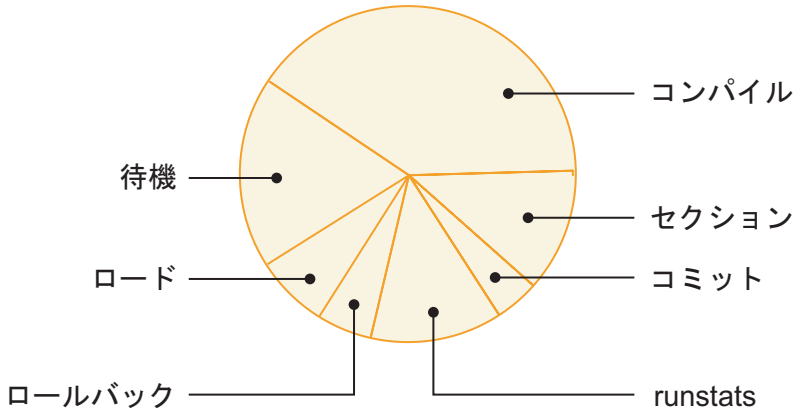
この照会の結果は、次のようなものになります (表示形式上、照会の出力行が分割されています)。

```
WAIT                 COMPILE              IMP_COMPILE          SECTION              COMMIT
-------------------- -------------------- -------------------- -------------------- --------------------
                 611                 1931                    0                  395                   15

REORG                RUNSTATS             ROLLBACK             LOAD
-------------------- -------------------- -------------------- --------------------
                   0                  432                   18                    0

  1 record(s) selected.
```

報告された数値を使用して、待機に費やされた時間とさまざまな処理ステージで費やされた時間の相対比較を示す円グラフを構成することもできます (コンポーネント時間 0 は含まれません)。



例 3: 合計消費時間と各コンポーネントでの処理時間の比率を表示する。

この例は、処理の各ステージ (コンポーネント) での作業に費やされた時間の概要を、そのコンポーネントで費やされた合計時間に対する相対比較で知る方法を示しています。次の照会は、特定のコンポーネントで費やされた合計経過時間に対する実際の処理で費やされた時間の比率 (パーセンテージとして表される) を計算します。

```
WITH PCTPROC AS (
     SELECT SUM(TOTAL_SECTION_TIME) AS SECT_TIME, SUM(TOTAL_SECTION_PROC_TIME) AS SECT_PROC_TIME,
            SUM(TOTAL_COMPILE_TIME) AS COMP_TIME, SUM(TOTAL_COMPILE_PROC_TIME) AS COMP_PROC_TIME,
            SUM(TOTAL_IMPLICIT_COMPILE_TIME) AS IMP_C_TIME, SUM(TOTAL_IMPLICIT_COMPILE_PROC_TIME) AS IMP_C_PROC_TIME,
            SUM(TOTAL_COMMIT_TIME) AS COMMIT_TIME, SUM(TOTAL_COMMIT_PROC_TIME) AS COMMIT_PROC_TIME,
            SUM(TOTAL_ROLLBACK_TIME) AS ROLLBACK_TIME, SUM(TOTAL_ROLLBACK_PROC_TIME) AS ROLLBACK_PROC_TIME,
            SUM(TOTAL_RUNSTATS_TIME) AS RUNSTATS_TIME, SUM(TOTAL_RUNSTATS_PROC_TIME)AS RUNSTATS_PROC_TIME,
            SUM(TOTAL_REORG_TIME) AS REORG_TIME, SUM(TOTAL_REORG_PROC_TIME) AS REORG_PROC_TIME,
            SUM(TOTAL_LOAD_TIME) AS LOAD_TIME, SUM(TOTAL_LOAD_PROC_TIME) AS LOAD_PROC_TIME
     FROM TABLE(MON_GET_CONNECTION(NULL, -2)) AS METRICS)
     SELECT CASE WHEN SECT_TIME > 0
                 THEN DEC((FLOAT(SECT_PROC_TIME) / FLOAT(SECT_TIME)) * 100,5,1)
                 ELSE NULL END AS SECT_PROC_PCT,
            CASE WHEN COMP_TIME > 0
                 THEN DEC((FLOAT(COMP_PROC_TIME) / FLOAT(COMP_TIME)) * 100,5,1)
                 ELSE NULL END AS COMPILE_PROC_PCT,
            CASE WHEN IMP_C_TIME > 0
                 THEN DEC((FLOAT(IMP_C_PROC_TIME) / FLOAT(IMP_C_TIME)) * 100,5,1)
                 ELSE NULL END AS IMPL_COMPILE_PROC_PCT,
              CASE WHEN ROLLBACK_TIME > 0
                 THEN DEC((FLOAT(ROLLBACK_PROC_TIME) / FLOAT(ROLLBACK_TIME)) * 100,5,1)
                 ELSE NULL END AS ROLLBACK_PROC_PCT,
              CASE WHEN COMMIT_TIME > 0
                 THEN DEC((FLOAT(COMMIT_PROC_TIME) / FLOAT(COMMIT_TIME)) * 100,5,1)
                 ELSE NULL END AS COMMIT_PROC_PCT,
              CASE WHEN RUNSTATS_TIME > 0
                 THEN DEC((FLOAT(RUNSTATS_PROC_TIME) / FLOAT(RUNSTATS_TIME)) * 100,5,1)
                 ELSE NULL END AS RUNSTATS_PROC_PCT,
            CASE WHEN REORG_TIME > 0
                 THEN DEC((FLOAT(REORG_PROC_TIME) / FLOAT(REORG_TIME)) * 100,5,1)
                 ELSE NULL END AS REORG_PROC_PCT,
            CASE WHEN LOAD_TIME > 0
                 THEN DEC((FLOAT(LOAD_PROC_TIME) / FLOAT(LOAD_TIME)) * 100,5,1)
                 ELSE NULL END AS LOAD_PROC_PCT
       FROM PCTPROC
```

この照会は、以下の出力を生成します。

```
SECT_PROC_PCT COMPILE_PROC_PCT IMPL_COMPILE_PROC_PCT ROLLBACK_PROC_PCT COMMIT_PROC_PCT RUNSTATS_PROC_PCT REORG_PROC_PCT LOAD_PROC_PCT
------------- ---------------- ---------------------- ----------------- ---------------- ------------------ ------------- -------------
         57.6              0.1                      -              96.9             95.6                0.0          71.1          84.6
  1 record(s) selected.
```

このデータのグラフィカル表現は、671 ページの図 17 に示すようなものになります。

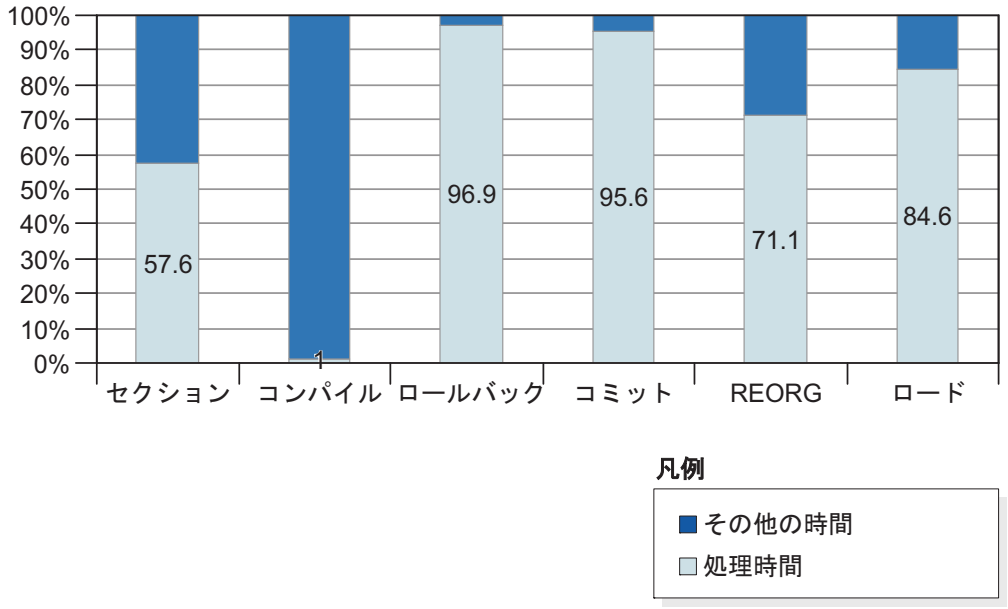


図 17. 全体消費時間に対するパーセンテージで表されたコンポーネント処理時間

例 4: 消費時間モニター・エレメントのランキングを表示する。

これまでの例では、表示されるモニター・エレメントはすべて照会 SQL で明示的に指定されており、照会結果ではそれぞれが独自の列に表示されます。しかし、調べる消費時間モニター・エレメントが分かっていないときもあります。例えば、上位 10 の待機時間モニター・エレメントを知りたい場合や、ゼロ以外の消費時間モニター・エレメントのみを知りたい場合があります。

各エレメントが単独の行で表示される行指向フォーマットでモニター・エレメントを表示するために使用できるいくつかの表関数が、DB2 バージョン 9.7 フィックスパック 1 で追加されました。この処理に使用できる表関数には、MON_FORMAT_XML_* _BY_ROW 形式の名前が付けられています。これらの関数は、特定のモニター・インターフェースから返される XML 文書からメトリックを抽出します。 (詳しくは、 20 ページの『モニター・データを XML 文書で返すインターフェース』を参照してください。) MON_FORMAT_XML_* _BY_ROW 関数は、表示したいエレメントが分かっていないときに便利です。例えば、CLPWORKLOAD というワークロードの待機時間モニター・エレメントの上位 10 エレメントを調べることもできます。この情報を収集するには、DBSTATS (event_wlstats 論理データ・グループ) という統計イベント・モニターを作成します。このイベント・モニターを表に書き込むようにセットアップしたとすると、メトリックは DETAILS_XML という列に記録されます。以下のように、イベント・モニターからの出力表にモニター・データを設定したら、MON_FORMAT_XML_WAIT_TIMES_BY_ROW 関数を使用する照会を構成することによって、調べたいモニター・エレメントを抽出できます。

```
SELECT SUBSTR(STATS.WORKLOAD_NAME,1,15) AS WORKLOAD_NAME,
       SUBSTR(METRICS.METRIC_NAME,1,30) AS METRIC_NAME,
       SUM(METRICS.TOTAL_TIME_VALUE) AS TOTAL_TIME_VALUE
FROM   WLSTATS_DBSTATS AS STATS,
       TABLE(MON_FORMAT_XML_WAIT_TIMES_BY_ROW(STATS.DETAILS_XML)) AS METRICS
```

```
WHERE  WORKLOAD_NAME='CLPWORKLOAD' AND (PARENT_METRIC_NAME='TOTAL_WAIT_TIME')
GROUP  BY WORKLOAD_NAME,METRIC_NAME
ORDER  BY TOTAL_TIME_VALUE DESC
FETCH FIRST 10 ROWS ONLY
```

要確認: 消費時間モニター・エレメントは、階層的に編成されています。この例では、待機時間の二重カウントを避けるために、**total_wait_time** に累積されるモニター・エレメントのみを含めています (上記 SQL ステートメントの WHERE 節を参照)。そうしないと、個別の待機時間をいくつも含んでいる **total_wait_time** 自体が結果に入ってしまうことになります。
この照会の結果は、例えば以下のような出力になります。

```
WORKLOAD_NAME   METRIC_NAME                    TOTAL_TIME_VALUE
--------------- ------------------------------ --------------------
CLPWORKLOAD     LOCK_WAIT_TIME                        15138541
CLPWORKLOAD     DIRECT_READ_TIME                       6116231
CLPWORKLOAD     POOL_READ_TIME                         6079458
CLPWORKLOAD     DIRECT_WRITE_TIME                       452627
CLPWORKLOAD     POOL_WRITE_TIME                         386208
CLPWORKLOAD     IPC_SEND_WAIT_TIME                      283172
CLPWORKLOAD     LOG_DISK_WAIT_TIME                      103888
CLPWORKLOAD     DIAGLOG_WRITE_WAIT_TIME                  78198
CLPWORKLOAD     IPC_RECV_WAIT_TIME                       15612
CLPWORKLOAD     TCPIP_SEND_WAIT_TIME                      3291

  10 record(s) selected.
```

SQL ステートメント実行時にどこで時間を費やしたかを判別する

アクティビティー・レベルの消費時間情報を検索する 1 つの例として、特定の SQL ステートメントの消費時間モニター・エレメントを表示する場合が挙げられます。この情報を検索するには、MON_GET_PKG_CACHE_STMT 表関数を使用できます。

このタスクについて

このタスクでは、パッケージ・キャッシュ内の SQL ステートメントに関する選択した消費時間の詳細を検索する方法の例を示します。

注:

- パッケージ・キャッシュ内のある特定のステートメントについて報告される消費時間メトリックは、そのステートメントのすべての実行についての消費時間メトリックの総計です。
- 照会の出力で示される値は例示のみを目的としており、ご使用のシステムで表示される可能性のあるものを代表していると解釈すべきではありません。

手順

1. パッケージ・キャッシュ内のステートメントに関する情報を検索する SQL ステートメントを、MON_GET_PKG_CACHE_STMT 表関数を使用して作成します。例えば、ステートメントの合計実行時間に対する合計待機時間を確認するとします。この情報を検索するための照会は、例えば次のようになります。

```
SELECT SUM(STMT_EXEC_TIME) AS TOTAL_EXEC_TIME,
       SUM(TOTAL_ACT_WAIT_TIME) AS TOTAL_WAIT_TIME,
       EXECUTABLE_ID
FROM TABLE(MON_GET_PKG_CACHE_STMT ( NULL, NULL, NULL, -2)) AS T
WHERE STMT_EXEC_TIME <> 0
GROUP BY EXECUTABLE_ID
ORDER BY TOTAL_EXEC_TIME DESC
```

2. この照会を実行します。 結果は、例えば次の出力のようになります。

```
TOTAL_EXEC_TIME      TOTAL_WAIT_TIME      EXECUTABLE_ID
-------------------- -------------------- --------------------------------------------------------------------------------
                9021                 9021 x'0100000000000000320000000000000000000000020020091111120320140000'
                3017                  372 x'0100000000000000030000000000000000000000020020091111115438062000'
                 591                    0 x'0100000000000000010000000000000000000000020020091111115252265000'
                 203                  192 x'0100000000000000270000000000000000000000020020091111115936750000'
                 142                    0 x'01000000000000002B0000000000000000000000020020091111115944000000'
                 111                   48 x'0100000000000000070000000000000000000000020020091111115441359002'
                 108                   35 x'01000000000000000B0000000000000000000000020020091111115441750000'
                  55                    0 x'01000000000000000D0000000000000000000000020020091111115442062000'
                  50                    0 x'01000000000000000C0000000000000000000000020020091111115441921000'
                  38                    0 x'0100000000000000260000000000000000000000020020091111115936609003'
                  35                    2 x'01000000000000000A0000000000000000000000020020091111115441609000'
                  35                   35 x'0100000000000000130000000000000000000000020020091111115442593001'
                  33                    0 x'0100000000000000120000000000000000000000020020091111115442531000'
                  32                    0 x'0100000000000000240000000000000000000000020020091111115936578000'
                  29                    0 x'01000000000000000E0000000000000000000000020020091111115442203000'
                  24                   23 x'0100000000000000040000000000000000000000020020091111115440640000'
                  24                    0 x'0100000000000000110000000000000000000000020020091111115442484003'
                  20                    0 x'0100000000000000300000000000000000000000020020091111120241828000'
                  15                    0 x'0100000000000000050000000000000000000000020020091111115440984000'
                  14                    0 x'0100000000000000080000000000000000000000020020091111115441437000'
                  13                   13 x'010000000000000F0000000000000000000000020020091111115442406001'
                   4                    0 x'0100000000000000100000000000000000000000020020091111115442484001'
                   3                    0 x'0100000000000000180000000000000000000000020020091111115442828000'
                   3                    3 x'010000000000000001F0000000000000000000000020020091111115936515000'
                   3                    0 x'0100000000000000290000000000000000000000020020091111115943968001'
                   2                    0 x'0100000000000000150000000000000000000000020020091111115442656001'
                   2                    0 x'0100000000000000170000000000000000000000020020091111115442750000'
                   1                    0 x'0100000000000000160000000000000000000000020020091111115442734000'
                   1                    0 x'0100000000000000280000000000000000000000020020091111115937000001'
                   1                    0 x'01000000000000002A0000000000000000000000020020091111115943984000'

  30 record(s) selected.
```

タスクの結果

この時点で、もう一度 MON_GET_PKG_CACHE_STMT 表関数を使用して、特に関心のあるステートメントのステートメント・テキストを検索することもできます。例えば、次の照会を使用して、上記の待機時間が最も長いステートメントを確認できます。

```
SELECT VARCHAR(STMT_TEXT, 80) AS STMT_TEXT
FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL,x'0100000000000000320000000000000000000000020020091111120320140000',NULL,-2))
     AS T
```

この照会の出力は、次のようなものになります。

```
STMT_TEXT
--------------------------------------------------------------------------------
UPDATE EMPLOYEE SET BONUS=10000 WHERE PERF_RATING=1
  1 record(s) selected.
```

第 10 章 論理データ・グループの概要

モニター・エレメントのデータを調べるときは、関連するエレメントを同時に複数参照すると有用である場合がよくあります。*論理データ・グループ* は、相互に補完するエレメントをグループ化したものです。

例えば、uow 論理データ・グループには、 **appl_id** (アプリケーション ID) や **appl_name** (アプリケーション名) などのエレメントが含まれています。これらの 2 つのエレメントが一緒に確認するエレメントであることは容易に予測できます。

論理データ・グループは、スナップショット・モニター・インターフェースで使用されますが、特にイベント・モニターで使用されます。DB2 製品には 1,000 個を超えるモニター・エレメントがあります。モニター・エレメントを調べる場合は必ず、どのエレメントを参照する必要があるのか検討し、明確に指定しなければならないため、非常に面倒に感じることがあるでしょう。例えば、イベント・モニターを作成する場合、どのエレメントについてデータをキャプチャーするか検討して指定する作業に、時間がかかると考えられます。これを避けるため、DB2 製品は、各イベント・モニターにデフォルトの論理データ・グループのセットを関連付けています。これは、CREATE EVENT MONITOR ステートメントに何も指定せずに、キャプチャーするイベントに関連するモニター・エレメントのみが含まれた、有効なモニター・エレメントのセットをキャプチャーできることを意味します。通常の表に書き込むイベント・モニターの場合は、モニター・エレメント・データをキャプチャーする論理データ・グループを、さらに柔軟に指定できます。

未フォーマット・イベント (UE) 表に書き込むイベント・モニターの場合も、デフォルトのモニター・エレメントのセットをキャプチャーします。EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用してリレーショナル表を生成する際には、論理データ・グループを使用して、関連するエレメントが別々の表にグループ化されます。例えば、 `lock` 論理データ・グループには LOCK_EVENT 表に使用されるエレメントが入っており、`participant` 論理データ・グループには LOCK_PARTICIPANT 表に使用されるエレメントが入っています。

イベント・モニターの論理データ・グループおよびモニター・エレメント

一緒に調べると有用であることが多いモニター・エレメントは、*論理データ・グループ* にグループ化されます。

すべてのイベント・モニターは、何らかの形で論理データ・グループを使用します。一部のイベント・モニター・タイプでは、情報を記録する論理データ・グループを指定することによって、収集する情報を指定できます。論理データ・グループは、イベント・モニターが生成する出力の、データをグループ化するときにも使用されます。例えば、表に書き込むイベント・モニターは、一般的にはモニター・エレメントの論理データ・グループごとに 1 つの表を作成します。

次の表は、論理データ・グループと、イベント・モニターによって戻されるモニター・エレメントの一覧表です。

- 53 ページの『changesummary 論理データ・グループ』
- 54 ページの『dbdbmcfg 論理データ・グループ』
- 54 ページの『ddlstmtexec 論理データ・グループ』
- 54 ページの『dllock 論理データ・グループ』
- 55 ページの『event_activity 論理データ・グループ』
- 58 ページの『event_activitymetrics 論理データ・グループ』
- 62 ページの『event_activitystmt 論理データ・グループ』
- 63 ページの『event_activityvals 論理データ・グループ』
- 63 ページの『event_bufferpool 論理データ・グループ』
- 65 ページの『event_conn 論理データ・グループ』
- 68 ページの『event_connheader 論理データ・グループ』
- 69 ページの『event_connmemuse 論理データ・グループ』
- 69 ページの『event_data_value 論理データ・グループ』
- 70 ページの『event_db 論理データ・グループ』
- 74 ページの『event_dbheader 論理データ・グループ』
- 75 ページの『event_dbmemuse 論理データ・グループ』
- 75 ページの『event_deadlock 論理データ・グループ』
- 75 ページの『event_detailed_dlconn 論理データ・グループ』
- 77 ページの『event_dlconn 論理データ・グループ』
- 78 ページの『event_histogrambin 論理データ・グループ』
- 78 ページの『event_log_header 論理データ・グループ』
- 78 ページの『event_overflow 論理データ・グループ』
- 79 ページの『event_qstats 論理データ・グループ』
- 79 ページの『event_scmetrics 論理データ・グループ』
- 89 ページの『event_scstats 論理データ・グループ』
- 91 ページの『event_start 論理データ・グループ』
- 91 ページの『event_stmt 論理データ・グループ』
- 93 ページの『event_stmt_history 論理データ・グループ』
- 93 ページの『event_subsection 論理データ・グループ』
- 94 ページの『event_table 論理データ・グループ』
- 95 ページの『event_tablespace 論理データ・グループ』
- 96 ページの『event_thresholdviolations 論理データ・グループ』
- 97 ページの『event_wcstats 論理データ・グループ』
- 98 ページの『event_wlmetrics 論理データ・グループ』
- 108 ページの『event_wlstats 論理データ・グループ』
- 110 ページの『event_xact 論理データ・グループ』
- 111 ページの『evmonstart 論理データ・グループ』
- 111 ページの『lock 論理データ・グループ』
- 113 ページの『lock_participants 論理データ・グループ』
- 112 ページの『lock_participant_activities 論理データ・グループ』

- 111 ページの『lock_activity_values 論理データ・グループ』
- 115 ページの『pkgcache 論理データ・グループ』
- 116 ページの『pkgcache_metrics 論理データ・グループ』
- 121 ページの『pkgcache_stmt_args 論理データ・グループ』
- 122 ページの『regvar 論理データ・グループ』
- 122 ページの『sqlca 論理データ・グループ』
- 122 ページの『txncompletion 論理データ・グループ』
- 123 ページの『uow 論理データ・グループ』
- 125 ページの『uow_metrics 論理データ・グループ』
- 133 ページの『uow_package_list 論理データ・グループ』
- 124 ページの『uow_executable_list 論理データ・グループ』
- 133 ページの『utillocation 論理データ・グループ』
- 133 ページの『utilphase 論理データ・グループ』
- 134 ページの『utilstart 論理データ・グループ』
- 134 ページの『utilstop 論理データ・グループ』

changesummary 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

939 ページの『coord_member - コーディネーター・メンバー : モニター・エレメント』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

1729 ページの『utility_type ユーティリティー・タイプ』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

853 ページの『application_handle - アプリケーション・ハンドル : モニター・エレメント』

1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

900 ページの『client_port_number - クライアント・ポート番号 : モニター・エレメント』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

896 ページの『client_hostname - クライアント・ホスト名 : モニター・エレメント』

903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』

893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』

902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』

894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』

869 ページの『backup_timestamp - バックアップ・タイム・スタンプ』

dbdbmcfg 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

888 ページの『cfg_name - 構成名』

890 ページの『cfg_value - 構成値』

890 ページの『cfg_value_flags - 構成値フラグ』

889 ページの『cfg_old_value - 古い構成値』

889 ページの『cfg_old_value_flags - 古い構成値のフラグ』

888 ページの『cfg_collection_type - 構成収集タイプ』

980 ページの『deferred - 据え置き』

ddlstmtexec 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1061 ページの『global_transaction_id - グローバル・トランザクション ID のモニター・エレメント』

1126 ページの『local_transaction_id - ローカル・トランザクション ID のモニター・エレメント』

1474 ページの『savepoint_id - セーブポイント ID』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

975 ページの『ddl_classification - DDL 種別』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

dllock 論理データ・グループ

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1129 ページの『lock_count ロック・カウント : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1144 ページの『lock_object_name ロック対象名』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

1148 ページの『lock_status - ロック状況 : モニター・エレメント』

1208 ページの『node_number ノード番号』

1554 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

注: この論理データ・グループの基礎となる実装は、スナップショット・モニターの LOCK 論理データ・グループです。ファイルおよびパイプ出力オプションで使用される自己記述型ストリームで、この論理グループの出力を参照すると、 LOCK グループを使用して出力が生成されていることを確認できます。

event_activity 論理データ・グループ

812 ページの『act_exec_time アクティビティー実行時間 : モニター・エレメント』

817 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

820 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

821 ページの『activity_type アクティビティー・タイプ : モニター・エレメント』

823 ページの『address - 接続の開始元となった IP アドレス』

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

855 ページの『arm_correlator アプリケーション応答測定相関関係子 : モニター・エレメント』

940 ページの『coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント』

971 ページの『db_work_action_set_id データベース作業アクション・セット ID : モニター・エレメント』

972 ページの『db_work_class_id データベース作業クラス ID : モニター・エレメント』

982 ページの『details_xml - 詳細 XML : モニター・エレメント』

1110 ページの『intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント』

1220 ページの『num_remaps 再マップ数 : モニター・エレメント』

1256 ページの『parent_activity_id 親アクティビティー ID : モニター・エレメント』

1256 ページの『parent_uow_id 親作業単位 ID : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』

1474 ページの『sc_work_action_set_id サービス・クラス作業アクション・セット ID : モニター・エレメント』

1475 ページの『sc_work_class_id サービス・クラス作業クラス ID : モニター・エレメント』

1476 ページの『section_actuals - セクション実行時統計 : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1510 ページの『sqlca SQL 連絡域 (SQLCA)』

1599 ページの『time_completed 完了時刻 : モニター・エレメント』

1599 ページの『time_created 作成時刻 : モニター・エレメント』

1600 ページの『time_started 開始時刻 : モニター・エレメント』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1740 ページの『workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1423 ページの『prep_time 準備時間 : モニター・エレメント』

1432 ページの『query_card_estimate 照会行数の見積もり』

1433 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

1463 ページの『rows_fetched フェッチ行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

1553 ページの『system_cpu_time システム CPU 時間 : モニター・エレメント』

1721 ページの『user_cpu_time ユーザー CPU 時間 : モニター・エレメント』

1731 ページの『wl_work_action_set_id - ワークロード作業アクション・セット ID : モニター・エレメント』

1732 ページの『wl_work_class_id - ワークロード作業クラス ID : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』
1193 ページの『member - データベース・メンバー・モニター・エレメント』
1434 ページの『query_data_tag_list - 見積もりの照会データ・タグ・リストのモニター・エレメント』
1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』
1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』
1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』
1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

event_activitymetrics 論理データ・グループ

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』
860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』
862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』
864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』
941 ページの『coord_stmt_exec_time - コーディネーター・エージェントのステートメントの実行時間 : モニター・エレメント』
978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』
984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』
986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』
987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』
989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』
991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』
994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』
996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』
998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』
1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』
1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』
1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1657 ページの『total_routine_non_sect_proc_time - 非セクション処理時間 : モニター・エレメント』

1658 ページの『total_routine_non_sect_time - 非セクション・ルーチンの実行時間 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

event_activitystmt 論理データ・グループ

817 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

820 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

908 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

1005 ページの『eff_stmt_text - 有効なステートメント・テキスト : モニター・エレメント』

1020 ページの『executable_id 実行可能 ID : モニター・エレメント』

1110 ページの『intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1220 ページの『num_routines - ルーチン数のモニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』

1460 ページの『routine_id - ルーチン ID : モニター・エレメント』

1477 ページの『section_env セクション環境 : モニター・エレメント』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1528 ページの『stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ : モニター・エレメント』

1530 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』

1530 ページの『stmt_isolation ステートメント分離』

1531 ページの『stmt_last_use_time - ステートメント最終使用時タイム・スタンプ : モニター・エレメント』

1531 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』

1532 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1535 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』

1537 ページの『stmt_source_id ステートメント・ソース ID』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

1546 ページの『stmtno - ステートメント番号のモニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

event_activityvals 論理データ・グループ

817 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

820 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1110 ページの『intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント』

1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』

1543 ページの『stmt_value_data 値データ』

1543 ページの『stmt_value_index 値索引』

1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

event_bufferpool 論理データ・グループ

874 ページの『bp_id バッファー・プール ID : モニター・エレメント』

874 ページの『bp_name - バッファー・プール名 : モニター・エレメント』

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1012 ページの『event_time イベント時刻』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1057 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1271 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1278 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

1279 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

1280 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

1281 ページの『pool_async_write_time - バッファー・プール非同期書き込み時間 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

871 ページの『block_ios - ブロック入出力要求数 : モニター・エレメント』

1254 ページの『pages_from_block_ios - ブロック入出力によって読み取られたページ数の合計 : モニター・エレメント』

1254 ページの『pages_from_vectored_ios - ベクトル化入出力によって読み取られたページ数の合計 : モニター・エレメント』

1277 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1286 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

1343 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

1730 ページの『vectored_ios - ベクトル化入出力要求数 : モニター・エレメント』

event_conn 論理データ・グループ

808 ページの『acc_curs_blk 受け入れられたブロック・カーソル要求』

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

848 ページの『appl_priority アプリケーション・エージェント優先順位』

849 ページの『appl_priority_type アプリケーション優先順位タイプ』

849 ページの『appl_section_inserts セクション挿入数 : モニター・エレメント』

850 ページの『appl_section_lookups - セクション検索』

866 ページの『authority_bitmap ユーザー許可レベル : モニター・エレメント』

867 ページの『authority_lvl ユーザー許可レベル : モニター・エレメント』

870 ページの『binds_precompiles 試行されたバインド/プリコンパイル』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

883 ページの『cat_cache_overflows カタログ・キャッシュ・オーバーフロー数』

907 ページの『commit_sql_stmts - 試行されたコミット・ステートメント』

976 ページの『ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1002 ページの『disconn_time データベース非アクティブ化タイム・スタンプ』

1004 ページの『dynamic_sql_stmts 試行された動的 SQL ステートメント』

1024 ページの『failed_sql_stmts 失敗したステートメント操作』

1078 ページの『hash_join_overflows ハッシュ結合のオーバーフロー』

1079 ページの『hash_join_small_overflows ハッシュ結合の短精度オーバーフロー』

1102 ページの『int_auto_rebinds 内部自動再バインド』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1105 ページの『int_deadlock_rollbacks デッドロックによる内部ロールバック』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1238 ページの『olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1109 ページの『int_rows_updated 更新された内部行数』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1425 ページの『priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数』

1426 ページの『priv_workspace_section_inserts 専用ワークスペース・セクション挿入』

1427 ページの『priv_workspace_section_lookups 専用ワークスペース・セクション検索』

1427 ページの『priv_workspace_size_top 専用ワークスペースの最大サイズ』

1444 ページの『rej_curs_blk リジェクトされたブロック・カーソル要求』

1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1471 ページの『rows_selected 選択行数』

1472 ページの『rows_written 書き込み行数』

1479 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1489 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

1490 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

1490 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

1491 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1523 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

1553 ページの『system_cpu_time システム CPU 時間 : モニター・エレメント』

1634 ページの『total_hash_joins ハッシュ結合の合計』

1635 ページの『total_hash_loops ハッシュ・ループの合計』

1644 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

1669 ページの『total_sec_cons 2 次接続』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1708 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

1721 ページの『user_cpu_time ユーザー CPU 時間 : モニター・エレメント』

1742 ページの『x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント』

1744 ページの『xquery_stmts - 試行された XQuery ステートメント』

850 ページの『appl_status アプリケーション状況 : モニター・エレメント』

884 ページの『cat_cache_size_top - カタログ・キャッシュの最高水準点 : モニター・エレメント』

940 ページの『coord_node コーディネーター・ノード』

1007 ページの『elapsed_exec_time ステートメント実行経過時間』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1462 ページの『rows_deleted - 削除行数 : モニター・エレメント』

1463 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

1471 ページの『rows_updated - 更新行数 : モニター・エレメント』

879 ページの『cat_cache_heap_full - カタログ・キャッシュ・ヒープ・フルのモニター・エレメント』

event_connheader 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

866 ページの『auth_id 許可 ID』

895 ページの『client_db_alias アプリケーションで使用するデータベース別名』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

904 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

924 ページの『conn_time データベース接続時刻 : モニター・エレメント』

941 ページの『corr_token DRDA 相関トークン』

1023 ページの『execution_id ユーザー・ログイン ID』

1208 ページの『node_number ノード番号』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1592 ページの『territory_code データベース・テリトリー・コード』

898 ページの『client_nname - クライアント名モニター・エレメント』

event_connmemuse 論理データ・グループ

1208 ページの『node_number ノード番号』

1287 ページの『pool_config_size メモリー・プールの構成済みサイズ』

1288 ページの『pool_cur_size メモリー・プールの現行サイズ』

1324 ページの『pool_id メモリー・プール ID』

1375 ページの『pool_secondary_id メモリー・プール 2 次 ID』

1390 ページの『pool_watermark メモリー・プール水準点』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

POOL_LIST_ID

POOL_MAX_SIZE

event_data_value 論理データ・グループ

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1258 ページの『participant_no デッドロック内の参加者』

1529 ページの『stmt_history_id ステートメント履歴 ID』

1543 ページの『stmt_value_data 値データ』

1543 ページの『stmt_value_index 値索引』

1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

event_db 論理データ・グループ

1102 ページの『int_auto_rebinds 内部自動再バインド』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1170 ページの『log_held_by_dirty_pages ダーティー・ページによって占有されるログ・スペースの量』

1171 ページの『log_read_time ログ読み取り時間』

1172 ページの『log_reads 読み取られたログ・ページの数』

1172 ページの『log_to_redo_for_recovery リカバリーの場合に再実行されるログの量』

1173 ページの『log_write_time ログ書き込み時間』

1174 ページの『log_writes 書き込まれたログ・ページの数』

1216 ページの『num_log_read_io ログ読み取り数』

1217 ページの『num_log_write_io ログ書き込み数』

1221 ページの『num_threshold_violations しきい値違反の回数 : モニター・エレメント』

1238 ページの『olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

1268 ページの『pkg_cache_num_overflows パッケージ・キャッシュ・オーバーフロー』

1268 ページの『pkg_cache_size_top - パッケージ・キャッシュの最高水準点』

1271 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1425 ページの『priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数』

1426 ページの『priv_workspace_section_inserts 専用ワークスペース・セクション挿入』

1427 ページの『priv_workspace_section_lookups 専用ワークスペース・セクション検索』

1427 ページの『priv_workspace_size_top 専用ワークスペースの最大サイズ』

1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』

1462 ページの『rows_deleted - 削除行数 : モニター・エレメント』

1463 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1471 ページの『rows_selected 選択行数』

1471 ページの『rows_updated - 更新行数 : モニター・エレメント』

1475 ページの『sec_log_used_top 使用された最大 2 次ログ・スペース』

1479 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

1483 ページの『server_platform サーバーのオペレーティング・システム』

1489 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

1490 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

1490 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

1491 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1523 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

1524 ページの『stats_cache_size - 統計キャッシュのサイズ : モニター・エレメント』

1525 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間 : モニター・エレメント』

1526 ページの『stats_fabrications - 統計作成の合計数 : モニター・エレメント』

1551 ページの『sync_runstats - 同期 RUNSTATS アクティビティーの合計数 : モニター・エレメント』

1552 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント』

1601 ページの『tot_log_used_top 使用された最大合計ログ・スペース』

1617 ページの『total_cons データベース活動化以降の接続』

1634 ページの『total_hash_joins ハッシュ結合の合計』

1635 ページの『total_hash_loops ハッシュ・ループの合計』

1644 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1708 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

1742 ページの『x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント』

1744 ページの『xquery_stmts - 試行された XQuery ステートメント』

1007 ページの『elapsed_exec_time ステートメント実行経過時間』

1216 ページの『num_log_part_page_io 部分ログ・ページ書き込み数』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1286 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1506 ページの『sort_shrheap_top ソート共有ヒープの最高水準点』

879 ページの『cat_cache_heap_full - カタログ・キャッシュ・ヒープ・フルのモニター・エレメント』

LOG_FILE_ARCHIVE

LOG_FILE_NUM_CURR

LOG_FILE_NUM_FIRST

LOG_FILE_NUM_LAST

NUM_LOG_BUFF_FULL

NUM_LOG_DATA_IN_BUFF

event_dbheader 論理データ・グループ

924 ページの『conn_time データベース接続時刻 : モニター・エレメント』

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

event_dbmemuse 論理データ・グループ

1208 ページの『node_number ノード番号』

1287 ページの『pool_config_size メモリー・プールの構成済みサイズ』

1288 ページの『pool_cur_size メモリー・プールの現行サイズ』

1324 ページの『pool_id メモリー・プール ID』

1390 ページの『pool_watermark メモリー・プール水準点』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1375 ページの『pool_secondary_id メモリー・プール 2 次 ID』

POOL_MAX_SIZE

event_deadlock 論理データ・グループ

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1003 ページの『dl_conns - デッドロックに関係している接続 : モニター・エレメント』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1458 ページの『rolled_back_agent_id ロールバックされたエージェント』

1458 ページの『rolled_back_appl_id ロールバック・アプリケーション』

1459 ページの『rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント』

1459 ページの『rolled_back_sequence_no ロールバックされたシーケンス番号』

1523 ページの『start_time イベント開始時刻』

event_detailed_dlconn 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

845 ページの『appl_id_holding_lk ロックを保持しているアプリケーション ID』

872 ページの『blocking_cursor ブロック・カーソル』

926 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

958 ページの『cursor_name カーソル名』

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1142 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

1144 ページの『lock_node ロック・ノード』

1144 ページの『lock_object_name ロック対象名』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

1153 ページの『lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント』

1163 ページの『locks_held - ロック保持数 : モニター・エレメント』

1165 ページの『locks_in_list 報告されたロックの回数』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1258 ページの『participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1481 ページの『sequence_no_holding_lk ロックを保持しているシーケンス番号』

1523 ページの『start_time イベント開始時刻』

1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1129 ページの『lock_count ロック・カウント : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

event_dlconn 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

845 ページの『appl_id_holding_lk ロックを保持しているアプリケーション ID』

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1129 ページの『lock_count ロック・カウント : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1142 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1144 ページの『lock_node ロック・ノード』

1144 ページの『lock_object_name ロック対象名』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

1153 ページの『lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1258 ページの『participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1481 ページの『sequence_no_holding_lk ロックを保持しているシーケンス番号』

1523 ページの『start_time イベント開始時刻』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

event_histogrambin 論理データ・グループ

870 ページの『bin_id ヒストグラム・ビン ID : モニター・エレメント』

873 ページの『bottom ヒストグラム・ビンの最下位 : モニター・エレメント』

1080 ページの『histogram_type ヒストグラム・タイプ : モニター・エレメント』

1223 ページの『number_in_bin ビン内の数 : モニター・エレメント』

1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1600 ページの『top ヒストグラム・ビンの最上位 : モニター・エレメント』

1736 ページの『work_action_set_id 作業アクション・セット ID : モニター・エレメント』

1737 ページの『work_class_id 作業クラス ID : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_log_header 論理データ・グループ

878 ページの『byte_order イベント・データのバイト・オーダー』

904 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

1011 ページの『event_monitor_name イベント・モニター名』

1219 ページの『num_nodes_in_db2_instance パーティション内のノード数』

1482 ページの『server_instance_name サーバー・インスタンス名』

1483 ページの『server_prdid - サーバー製品/バージョン ID』

1592 ページの『territory_code データベース・テリトリー・コード』

1730 ページの『version モニター・データのバージョン』

event_overflow 論理データ・グループ

942 ページの『count - イベント・モニター・オーバーフロー数』

1058 ページの『first_overflow_time 最初のイベント・オーバーフロー時刻』

1120 ページの『last_overflow_time 最後のイベント・オーバーフロー時刻』

1208 ページの『node_number ノード番号』

event_qstats 論理データ・グループ

1123 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

1435 ページの『queue_assignments_total キュー割り当ての合計 : モニター・エレメント』

1436 ページの『queue_size_top キュー・サイズの最上位 : モニター・エレメント』

1436 ページの『queue_time_total キュー時間の合計 : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1595 ページの『threshold_domain しきい値ドメイン : モニター・エレメント』

1596 ページの『threshold_name しきい値名 : モニター・エレメント』

1597 ページの『threshold_predicate しきい値述部 : モニター・エレメント』

1598 ページの『thresholdid しきい値 ID : モニター・エレメント』

1736 ページの『work_action_set_name 作業アクション・セット名 : モニター・エレメント』

1737 ページの『work_class_name 作業クラス名 : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_scmetrics 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

1260 ページの『partition_number パーティション番号』

1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』

831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』

1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』

897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』

1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』

1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』

1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』

1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』

1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』

1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』

860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』

862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』

864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』

986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』

1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』

1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』

841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』

1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』

1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』

1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』

1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』

1664 ページの『total_rqst_mapped_in - マッピングの際の要求の合計 : モニター・エレメント』

1664 ページの『total_rqst_mapped_out - マッピングの際に除外された要求の合計 : モニター・エレメント』

813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』

809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』

1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』

1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』

1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』

1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』

1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』

1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』

1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』

1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』

1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』

1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』

1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』

1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』

1642 ページの『total_loads - ロード合計 : モニター・エレメント』

1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』

1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』

1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』

1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』

1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』

1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』

1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』

1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』

1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』

1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』

1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』

1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』

1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』

1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』

1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』

1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』

1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』

1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』

1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』

1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』

1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』

1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』

1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』

1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』

1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』

1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』

1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』

1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』

1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』

1357 ページの『pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』

1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』

1366 ページの『pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求のモニター・エレメント』

1353 ページの『pool_queued_async_other_reqs - プリフェッチャーが処理したその他の要求モニター・エレメント』

1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』

1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』

1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』

1355 ページの『pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求のモニター・エレメント』

1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』

1364 ページの『pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求のモニター・エレメント』

1308 ページの『pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求のモニター・エレメント』

1310 ページの『pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求のモニター・エレメント』

1322 ページの『pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求のモニター・エレメント』

1314 ページの『pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』

1317 ページの『pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント』

1319 ページの『pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求のモニター・エレメント』

1313 ページの『pool_failed_async_other_reqs - 失敗した非プリフェッチの要求のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

1647 ページの『total_peds - partial early distinct の合計回数のモニター・エレメント』

1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』

1416 ページの『post_threshold_peds - partial early distinct しきい値のモニター・エレメント』

1645 ページの『total_peas - partial early aggregation の合計回数のモニター・エレメント』

1413 ページの『post_threshold_peas - partial early aggregation しきい値のモニター・エレメント』

1703 ページの『tq_sort_heap_requests - 表キュー・ソート・ヒープ要求のモニター・エレメント』

1700 ページの『tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否のモニター・エレメント』

1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』

1622 ページの『total_connect_requests - 接続要求またはユーザー切り替え要求のモニター・エレメント』

1623 ページの『total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント』

1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』

1619 ページの『total_connect_authentications - 実行された接続またはユーザー切り替え認証のモニター・エレメント』

1620 ページの『total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1422 ページの『prefetch_waits - プリフェッチャーの待機カウントのモニター・エレメント』

1288 ページの『pool_data_gbp_indep_pages_found
_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』

1326 ページの『pool_index_gbp_indep_pages
_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』

1393 ページの『pool_xda_gbp_indep_pages
_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』

905 ページの『comm_exit_wait_time - 通信バッファー出口待機時間のモニター・エレメント』

906 ページの『comm_exit_waits - 通信バッファー出口待機回数のモニター・エレメント』

FCM_TQ_RECV_WAITS_TOTAL

FCM_MESSAGE_RECV_WAITS_TOTAL

FCM_TQ_SEND_WAITS_TOTAL

FCM_MESSAGE_SEND_WAITS_TOTAL

FCM_SEND_WAITS_TOTAL

FCM_RECV_WAITS_TOTAL

1092 ページの『ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント』

1093 ページの『ida_sends_total - データ送信回数 : モニター・エレメント』

1090 ページの『ida_send_volume - 送信された合計データ量 : モニター・エレメント』

1087 ページの『ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント』

1088 ページの『ida_recvs_total - データ受信回数 : モニター・エレメント』

1085 ページの『ida_recv_volume - 受信した合計データ量 : モニター・エレメント』

event_scstats 論理データ・グループ

811 ページの『act_cpu_time_top - アクティビティーの CPU 時間の最上位 : モニター・エレメント』

814 ページの『act_remapped_in - 再マッピングするアクティビティー : モニター・エレメント』

814 ページの『act_remapped_out - 再マッピングの際に除外されるアクティビティー : モニター・エレメント』

814 ページの『act_rows_read_top - アクティビティーの読み取り行数の最上位 : モニター・エレメント』

816 ページの『act_throughput - アクティビティー・スループット : モニター・エレメント』

835 ページの『agg_temp_tablespace_top 集約 TEMPORARY 表スペースの最上位 : モニター・エレメント』

911 ページの『concurrent_act_top 並行アクティビティーの最上位 : モニター・エレメント』

913 ページの『concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント』

912 ページの『concurrent_connection_top 並行接続の最上位 : モニター・エレメント』

930 ページの『coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント』

930 ページの『coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント』

931 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

932 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

933 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

934 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

935 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

936 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

937 ページの『coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント』

942 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

951 ページの『cpu_utilization - CPU 使用率 : モニター・エレメント』

982 ページの『details_xml - 詳細 XML : モニター・エレメント』

1123 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

1203 ページの『metrics - メトリック : モニター・エレメント』.)

1455 ページの『request_exec_time_avg 要求の平均実行時間 : モニター・エレメント』

1470 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1592 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1710 ページの『uow_completed_total - 完了済みの合計作業単位 : モニター・エレメント』

1712 ページの『uow_lifetime_avg - 作業単位の平均存続期間 : モニター・エレメント』

1716 ページの『uow_throughput - 作業単位スループット : モニター・エレメント』

1716 ページの『uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_start 論理データ・グループ

1523 ページの『start_time イベント開始時刻』

event_stmt 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

834 ページの『agents_top 作成されたエージェントの数』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

872 ページの『blocking_cursor ブロック・カーソル』

926 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

958 ページの『cursor_name カーソル名』

1056 ページの『fetch_count 成功したフェッチの数』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1509 ページの『sql_req_id SQL ステートメントの要求 ID』

1510 ページの『sqlca SQL 連絡域 (SQLCA)』

1523 ページの『start_time イベント開始時刻』

1525 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間 : モニター・エレメント』

1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

1547 ページの『stop_time イベント停止時刻』

1552 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント』

1553 ページの『system_cpu_time システム CPU 時間 : モニター・エレメント』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1721 ページの『user_cpu_time ユーザー CPU 時間 : モニター・エレメント』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

event_stmt_history 論理データ・グループ

908 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

976 ページの『deadlock_id デッドロック・イベント ID』

977 ページの『deadlock_node デッドロック発生場所のパーティション番号』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1528 ページの『stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ : モニター・エレメント』

1529 ページの『stmt_history_id ステートメント履歴 ID』

1530 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』

1530 ページの『stmt_isolation ステートメント分離』

1531 ページの『stmt_last_use_time - ステートメント最終使用時タイム・スタンプ : モニター・エレメント』

1531 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』

1532 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1535 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』

1537 ページの『stmt_source_id ステートメント・ソース ID』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

event_subsection 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

1208 ページの『num_agents ステートメントで作動しているエージェントの数』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1519 ページの『ss_exec_time サブセクション実行経過時間』

1519 ページの『ss_node_number サブセクション・ノード番号』

1520 ページの『ss_number サブセクション番号 : モニター・エレメント』

1521 ページの『ss_sys_cpu_time サブセクションに使用されたシステム CPU 時間』

1521 ページの『ss_usr_cpu_time サブセクションに使用されたユーザー CPU 時間』

1698 ページの『tq_max_send_spills 表キュー・バッファー・オーバーフローの最大数』

1699 ページの『tq_rows_read 表キューから読み取られた行数』

1700 ページの『tq_rows_written 表キューに書き込まれた行数』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1509 ページの『sql_req_id SQL ステートメントの要求 ID』

event_table 論理データ・グループ

959 ページの『data_object_pages データ・オブジェクト・ページ数』

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

1012 ページの『event_time イベント時刻』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1098 ページの『index_object_pages 索引オブジェクト・ページ数』

1124 ページの『lob_object_pages LOB オブジェクト・ページ数』

1174 ページの『long_object_pages 長いオブジェクト・ページ数』

1246 ページの『overflow_accesses - オーバーフロー・レコードへのアクセス : モニター・エレメント』

1251 ページの『page_reorgs ページ再編成 : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1559 ページの『table_type - 表タイプ : モニター・エレメント』

1563 ページの『tablespace_id - 表スペース ID : モニター・エレメント』

1742 ページの『xda_object_pages XDA オブジェクト・ページ数』

event_tablespace 論理データ・グループ

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1012 ページの『event_time イベント時刻』

1014 ページの『evmon_activates イベント・モニター活動化回数』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

1057 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1271 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1277 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

1278 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

1279 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

1280 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

1281 ページの『pool_async_write_time - バッファー・プール非同期書き込み時間 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1343 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1286 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

TABLESPACE_FS_CACHING

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

event_thresholdviolations 論理データ・グループ

817 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

819 ページの『activity_collected 収集されたアクティビティー : モニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』

894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』

896 ページの『client_hostname - クライアント・ホスト名 : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

900 ページの『client_port_number - クライアント・ポート番号 : モニター・エレメント』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』

903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』

924 ページの『connection_start_time - 接続開始時刻 : モニター・エレメント』

940 ページの『coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント』

982 ページの『destination_service_class_id - 宛先サービス・クラス ID : モニター・エレメント』

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

1507 ページの『source_service_class_id ソース・サービス・クラス ID : モニター・エレメント』

1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』

1595 ページの『threshold_action しきい値アクション : モニター・エレメント』

1596 ページの『threshold_maxvalue しきい値最大値 : モニター・エレメント』

1597 ページの『threshold_predicate しきい値述部 : モニター・エレメント』

1598 ページの『threshold_queuesize しきい値キュー・サイズ : モニター・エレメント』

1598 ページの『thresholdid しきい値 ID : モニター・エレメント』

1599 ページの『time_of_violation 違反時刻 : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

event_wcstats 論理データ・グループ

811 ページの『act_cpu_time_top - アクティビティーの CPU 時間の最上位 : モニター・エレメント』

814 ページの『act_rows_read_top - アクティビティーの読み取り行数の最上位 : モニター・エレメント』

817 ページの『act_total アクティビティーの合計 : モニター・エレメント』

931 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

932 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

933 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

934 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

935 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

936 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

942 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

1123 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

1470 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1592 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

1736 ページの『work_action_set_id 作業アクション・セット ID : モニター・エレメント』

1736 ページの『work_action_set_name 作業アクション・セット名 : モニター・エレメント』

1737 ページの『work_class_id 作業クラス ID : モニター・エレメント』

1737 ページの『work_class_name 作業クラス名 : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_wlmetrics 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

1260 ページの『partition_number パーティション番号』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1739 ページの『workload_name ワークロード名 : モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』

831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』

1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』

897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』

1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』

1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』

1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』

1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』

1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』

1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』

860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』

862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』

864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』

986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』

1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』

1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』

841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』

1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』

1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』

1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』

1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』

813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』

809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』

1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』

1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』

1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』

1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』

1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』

1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』

1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』

1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』

1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』

1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』

1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』

1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』

1642 ページの『total_loads - ロード合計 : モニター・エレメント』

1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』

1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』

1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』

1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』

1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』

1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』

1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』

1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』

1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』

1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』

1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』

1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』

1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』

1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』

1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』

1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』

1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』

1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』

1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』

1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』

1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』

1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』

1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』

1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』

1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』

1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』

1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』

1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』

1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』

1357 ページの『pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』

1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』

1366 ページの『pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求のモニター・エレメント』

1353 ページの『pool_queued_async_other_reqs - プリフェッチャーが処理したその他の要求モニター・エレメント』

1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』

1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』

1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』

1355 ページの『pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求のモニター・エレメント』

1359 ページの『pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント』

1364 ページの『pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求のモニター・エレメント』

1308 ページの『pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求のモニター・エレメント』

1310 ページの『pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求のモニター・エレメント』

1322 ページの『pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求のモニター・エレメント』

1314 ページの『pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント』

1317 ページの『pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント』

1319 ページの『pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求のモニター・エレメント』

1313 ページの『pool_failed_async_other_reqs - 失敗した非プリフェッチの要求のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

1647 ページの『total_peds - partial early distinct の合計回数のモニター・エレメント』

1000 ページの『disabled_peds - 無効化された partial early distinct のモニター・エレメント』

1416 ページの『post_threshold_peds - partial early distinct しきい値のモニター・エレメント』

1645 ページの『total_peas - partial early aggregation の合計回数のモニター・エレメント』

1413 ページの『post_threshold_peas - partial early aggregation しきい値のモニター・エレメント』

1703 ページの『tq_sort_heap_requests - 表キュー・ソート・ヒープ要求のモニター・エレメント』

1700 ページの『tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否のモニター・エレメント』

1621 ページの『total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント』

1622 ページの『total_connect_requests - 接続要求またはユーザー切り替え要求のモニター・エレメント』

1623 ページの『total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント』

1618 ページの『total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント』

1619 ページの『total_connect_authentications - 実行された接続またはユーザー切り替え認証のモニター・エレメント』

1620 ページの『total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1422 ページの『prefetch_waits - プリフェッチャーの待機カウントのモニター・エレメント』

1288 ページの『pool_data_gbp_indep_pages_found
_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』

1326 ページの『pool_index_gbp_indep_pages
_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』

1393 ページの『pool_xda_gbp_indep_pages
_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』

905 ページの『comm_exit_wait_time - 通信バッファー出口待機時間のモニター・エレメント』

906 ページの『comm_exit_waits - 通信バッファー出口待機回数のモニター・エレメント』

FCM_TQ_RECV_WAITS_TOTAL

FCM_MESSAGE_RECV_WAITS_TOTAL

FCM_TQ_SEND_WAITS_TOTAL

FCM_MESSAGE_SEND_WAITS_TOTAL

FCM_SEND_WAITS_TOTAL

FCM_RECV_WAITS_TOTAL

1092 ページの『ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント』

1093 ページの『ida_sends_total - データ送信回数 : モニター・エレメント』

1090 ページの『ida_send_volume - 送信された合計データ量 : モニター・エレメント』

1087 ページの『ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント』

1088 ページの『ida_recvs_total - データ受信回数 : モニター・エレメント』

1085 ページの『ida_recv_volume - 受信した合計データ量 : モニター・エレメント』

event_wlstats 論理データ・グループ

811 ページの『act_cpu_time_top - アクティビティーの CPU 時間の最上位 : モニター・エレメント』

814 ページの『act_rows_read_top - アクティビティーの読み取り行数の最上位 : モニター・エレメント』

816 ページの『act_throughput - アクティビティー・スループット : モニター・エレメント』

913 ページの『concurrent_wlo_act_top 並行 WLO アクティビティーの最上位 : モニター・エレメント』

913 ページの『concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント』

930 ページの『coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント』

930 ページの『coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント』

931 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

932 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

933 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

934 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

935 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

936 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

937 ページの『coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント』

942 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

951 ページの『cpu_utilization - CPU 使用率 : モニター・エレメント』

982 ページの『details_xml - 詳細 XML : モニター・エレメント』

1123 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

1159 ページの『lock_wait_time_top - ロック待機時間の最上位 : モニター・エレメント』

1203 ページの『metrics - メトリック : モニター・エレメント』

1470 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

1524 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

1592 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1710 ページの『uow_completed_total - 完了済みの合計作業単位 : モニター・エレメント』

1712 ページの『uow_lifetime_avg - 作業単位の平均存続期間 : モニター・エレメント』

1716 ページの『uow_throughput - 作業単位スループット : モニター・エレメント』

1716 ページの『uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント』

1735 ページの『wlo_completed_total 完了したワークロード・オカレンスの合計 : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1739 ページの『workload_name ワークロード名 : モニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

1159 ページの『lock_wait_time_global_top - 最長グローバル・ロック待機時間 : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

event_xact 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1164 ページの『locks_held_top - ロック保持最大数 : モニター・エレメント』

1257 ページの『partial_record 部分レコード : モニター・エレメント』

1424 ページの『prev_uow_stop_time 直前の作業単位完了タイム・スタンプ』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1553 ページの『system_cpu_time システム CPU 時間 : モニター・エレメント』

1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

1713 ページの『uow_log_space_used - 使用されている作業単位ログ・スペース: モニター・エレメント』

1714 ページの『uow_start_time - 作業単位開始タイム・スタンプ : モニター・エレメント』

1715 ページの『uow_status 作業単位の状況』

1547 ページの『stop_time イベント停止時刻』

1721 ページの『user_cpu_time ユーザー CPU 時間 : モニター・エレメント』

1742 ページの『x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント』

1022 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

evmonstart 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

965 ページの『db2start_time - データベース・マネージャー開始タイム・スタンプ』

965 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』

lock 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

1003 ページの『dl_conns - デッドロックに関係している接続 : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1459 ページの『rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント』

978 ページの『deadlock_type - デッドロック・タイプのモニター・エレメント』

lock_activity_values 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1543 ページの『stmt_value_index 値索引』

1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1543 ページの『stmt_value_data 値データ』

1010 ページの『event_id - イベント ID モニター・エレメント』

lock_participant_activities 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

819 ページの『activity_id アクティビティー ID : モニター・エレメント』

821 ページの『activity_type アクティビティー・タイプ : モニター・エレメント』

926 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

1006 ページの『effective_isolation - 有効な分離 : モニター・エレメント』

1007 ページの『effective_query_degree - 有効な照会の度合い : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1097 ページの『incremental_bind - 追加バインドのモニター・エレメント』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_schema - パッケージ・スキーマ : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1432 ページの『query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント』

1447 ページの『reopt - REOPT バインド・オプションのモニター・エレメント』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1528 ページの『stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ : モニター・エレメント』

1530 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』

1531 ページの『stmt_last_use_time - ステートメント最終使用時タイム・スタンプ : モニター・エレメント』

1531 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』

1532 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1535 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』

1537 ページの『stmt_source_id ステートメント・ソース ID』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1542 ページの『stmt_unicode - ステートメントのユニコード・フラグのモニター・エレメント』

1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

lock_participants 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

827 ページの『agent_status DCS アプリケーション・エージェント』

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

853 ページの『application_handle - アプリケーション・ハンドル : モニター・エレメント』

866 ページの『auth_id 許可 ID』

893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』

894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』

902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』

903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』

936 ページの『coord_agent_tid - コーディネーター・エージェントのエンジン・ディスパッチ可能単位 ID のモニター・エレメント』

958 ページの『current_request - 現在の操作要求のモニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1129 ページの『lock_count ロック・カウント : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1142 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

LOCK_OBJECT_TYPE_ID

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

LOCK_RRIID

1148 ページの『lock_status - ロック状況 : モニター・エレメント』

1149 ページの『lock_timeout_val ロック・タイムアウト値 : モニター・エレメント』

1153 ページの『lock_wait_end_time - ロック待機終了タイム・スタンプ : モニター・エレメント』

1153 ページの『lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント』

1159 ページの『lock_wait_val - ロック待機値のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1232 ページの『object_requested - 要求されたオブジェクトのモニター・エレメント』

1258 ページの『participant_no デッドロック内の参加者』

1258 ページの『participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者』

1259 ページの『participant_type - 参加者タイプのモニター・エレメント』

1261 ページの『past_activities_wrapped - 過去のアクティビティー・リストの折り返しモニター・エレメント』

1485 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

1486 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

1554 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1598 ページの『thresholdid しきい値 ID : モニター・エレメント』

1596 ページの『threshold_name しきい値名 : モニター・エレメント』

1738 ページの『workload_id ワークロード ID : モニター・エレメント』

1739 ページの『workload_name ワークロード名 : モニター・エレメント』

828 ページの『agent_tid - エージェント・スレッド ID モニター・エレメント』

840 ページの『appl_action - アプリケーション・アクションのモニター・エレメント』

977 ページの『deadlock_member - デッドロック・メンバー : モニター・エレメント』

1435 ページの『queue_start_time - キュー開始タイム・スタンプのモニター・エレメント』

1437 ページの『queued_agents - キューに入れられているしきい値エージェントのモニター・エレメント』

1487 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

1744 ページの『xid トランザクション ID』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

pkgcache 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

908 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

1006 ページの『effective_isolation - 有効な分離 : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1020 ページの『executable_id 実行可能 ID : モニター・エレメント』

1102 ページの『insert_timestamp - 挿入タイムスタンプ : モニター・エレメント』

1120 ページの『last_metrics_update - メトリック最終更新タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1210 ページの『num_coord_exec - コーディネーター・エージェントによる実行数 : モニター・エレメント』

1210 ページの『num_coord_exec_with_metrics - メトリック付きの、コーディネーター・エージェントによる実行数 : モニター・エレメント』

1212 ページの『num_exec_with_metrics メトリックが収集された実行数 : モニター・エレメント』

1211 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』

1220 ページの『num_routines - ルーチン数のモニター・エレメント』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_schema - パッケージ・スキーマ : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1423 ページの『prep_time 準備時間 : モニター・エレメント』

1433 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

1434 ページの『query_data_tag_list - 見積もりの照会データ・タグ・リストのモニター・エレメント』

1460 ページの『routine_id - ルーチン ID : モニター・エレメント』

1477 ページの『section_env セクション環境 : モニター・エレメント』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1479 ページの『section_type - セクション・タイプ標識 : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1541 ページの『stmt_type_id - ステートメント・タイプ ID : モニター・エレメント』

1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』

1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』

1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1546 ページの『stmtno - ステートメント番号のモニター・エレメント』

1176 ページの『max_coord_stmt_exec_time - コーディネーターの最大ステートメント実行時間のモニター・エレメント』

1179 ページの『max_coord_stmt_exec_timestamp - コーディネーターの最大ステートメント実行のタイム・スタンプのモニター・エレメント』

pkgcache_metrics 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』

860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』

862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』

864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』

986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1528 ページの『stmt_exec_time - ステートメント実行時間 : モニター・エレメント』

941 ページの『coord_stmt_exec_time - コーディネーター・エージェントのステートメントの実行時間 : モニター・エレメント』

1657 ページの『total_routine_non_sect_proc_time - 非セクション処理時間 : モニター・エレメント』

1658 ページの『total_routine_non_sect_time - 非セクション・ルーチンの実行時間 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』

1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』

1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』

1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』

1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』

1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』

1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』

1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』

1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』

1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』

1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』

1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』

1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』

1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』

1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』

1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』

1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』

1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』

1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』

1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』

1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』

1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』

pkgcache_stmt_args 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1543 ページの『stmt_value_index 値索引』

1545 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

1544 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

1545 ページの『stmt_value_type 値タイプ : モニター・エレメント』

1543 ページの『stmt_value_data 値データ』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

regvar 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1443 ページの『regvar_name - レジストリー変数名』

1444 ページの『regvar_value - レジストリー変数の値』

1444 ページの『regvar_old_value - レジストリー変数の古い値』

1443 ページの『regvar_level - レジストリー変数のレベル』

1443 ページの『regvar_collection_type - レジストリー変数収集タイプ』

sqlca 論理データ・グループ

sqlcabc

sqlcaid

sqlcode

sqlerrd

sqlerrmc

sqlerrml

sqlerrp

sqlstate

sqlwarn

txncompletion 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1061 ページの『global_transaction_id - グローバル・トランザクション ID のモニター・エレメント』

1126 ページの『local_transaction_id - ローカル・トランザクション ID のモニター・エレメント』

1474 ページの『savepoint_id - セーブポイント ID』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

975 ページの『ddl_classification - DDL 種別』

1707 ページの『txn_completion_status - トランザクション完了状況』

uow 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

853 ページの『application_handle - アプリケーション・ハンドル : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

893 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』

894 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』

896 ページの『client_hostname - クライアント・ホスト名 : モニター・エレメント』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

900 ページの『client_port_number - クライアント・ポート番号 : モニター・エレメント』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

902 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』

903 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』

908 ページの『completion_status 完了状況 : モニター・エレメント』

924 ページの『conn_time データベース接続時刻 : モニター・エレメント』

939 ページの『coord_member - コーディネーター・メンバー : モニター・エレメント』

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1021 ページの『executable_list_size - 実行可能リスト・サイズのモニター・エレメント』

1061 ページの『global_transaction_id - グローバル・トランザクション ID のモニター・エレメント』

1110 ページの『intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント』

1126 ページの『local_transaction_id - ローカル・トランザクション ID のモニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

965 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』

1204 ページの『mon_interval_id - モニター間隔 ID のモニター・エレメント』

1248 ページの『package_list_exceeded - パッケージ・リストの超過 : モニター・エレメント』

1248 ページの『package_list_size - パッケージ・リスト・サイズのモニター・エレメント』

service_class_id サービス・クラス ID

service_subclass_name サービス・サブクラス名

service_superclass_name サービス・スーパークラス名

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

start_time イベント開始時刻

stop_time イベント停止時刻

1553 ページの『system_auth_id - システム許可 ID : モニター・エレメント』

UOW_CLIENT_PLATFORM

UOW_CLIENT_PROTOCOL

uow_id 作業単位 ID

1713 ページの『uow_log_space_used - 使用されている作業単位ログ・スペース: モニター・エレメント』

workload_id ワークロード ID

workload_name - ワークロード名

workload_occurrence_id ワークロード・オカレンス ID

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

1022 ページの『executable_list_truncated - 実行可能リスト切り捨てのモニター・エレメント』

924 ページの『connection_start_time - 接続開始時刻 : モニター・エレメント』

uow_executable_list 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1020 ページの『executable_id 実行可能 ID : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1211 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

uow_id 作業単位 ID

uow_metrics 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1734 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

1732 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』

1048 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

1028 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』

1053 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』

1033 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』

829 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』

831 ページの『agent_waits_total - エージェント待機の合計 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

1166 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1167 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』

1169 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』

1587 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』

1588 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』

897 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』

1112 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』

1113 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』

1115 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』

1116 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』

1590 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』

1591 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

858 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』

860 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』

862 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』

864 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』

984 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』

986 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』

1043 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』

1038 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』

1694 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』

1473 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』

1665 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』

841 ページの『app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント』

1607 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』

1671 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

1675 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』

1673 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1464 ページの『rows_modified 変更行数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1624 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』

810 ページの『act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1468 ページの『rows_returned 戻り行数 : モニター・エレメント』

978 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1044 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』

1040 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』

1041 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』

1036 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』

1035 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』

1030 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』

1031 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』

1026 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』

1054 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』

1049 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』

1051 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』

1046 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1589 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』

1586 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』

1114 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』

1111 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

857 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』

813 ページの『act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント』

809 ページの『act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1658 ページの『total_routine_time - 合計ルーチン時間 : モニター・エレメント』

1615 ページの『total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント』

1614 ページの『total_compilations - 合計コンパイル数 : モニター・エレメント』

1616 ページの『total_compile_time - 合計コンパイル時間 : モニター・エレメント』

1636 ページの『total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント』

1635 ページの『total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント』

1638 ページの『total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント』

1667 ページの『total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント』

1666 ページの『total_runstats - ランタイム統計の合計 : モニター・エレメント』

1668 ページの『total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント』

1649 ページの『total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント』

1652 ページの『total_reorgs - 再編成の合計 : モニター・エレメント』

1651 ページの『total_reorg_time - 合計再編成時間 : モニター・エレメント』

1639 ページの『total_load_proc_time - ロード処理時間の合計 : モニター・エレメント』

1642 ページの『total_loads - ロード合計 : モニター・エレメント』

1640 ページの『total_load_time - 合計ロード時間 : モニター・エレメント』

1670 ページの『total_section_proc_time - セクション処理時間の合計 : モニター・エレメント』

1608 ページの『total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント』

1677 ページの『total_section_time - 合計セクション時間 : モニター・エレメント』

1611 ページの『total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント』

1605 ページの『total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント』

1612 ページの『total_commit_time - 合計コミット時間 : モニター・エレメント』

1653 ページの『total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント』

1606 ページの『total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント』

1654 ページの『total_rollback_time - 合計ロールバック時間 : モニター・エレメント』

1660 ページの『total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント』

1662 ページの『total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント』

1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』

1217 ページの『num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント』

1655 ページの『total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

880 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント』

882 ページの『cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

815 ページの『act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント』

1603 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』

1602 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』

1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』

1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』

1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』

1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』

1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』

1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』

1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』

1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』

1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』

1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』

1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』

1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』

1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』

1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』

1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』

1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

1015 ページの『evmon_wait_time - イベント・モニターの待機時間モニター・エレメント』

1018 ページの『evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント』

1629 ページの『total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント』

1631 ページの『total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント』

1682 ページの『total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント』

1685 ページの『total_stats_fabrications - 統計作成の合計回数のモニター・エレメント』

1683 ページの『total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント』

1688 ページの『total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント』

1689 ページの『total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント』

1686 ページの『total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント』

1627 ページの『total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント』

1346 ページの『pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント』

1351 ページの『pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント』

1370 ページの『pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント』

1344 ページの『pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント』

1349 ページの『pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント』

1368 ページの『pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント』

838 ページの『app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

837 ページの『app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

839 ページの『app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント』

uow_package_list 論理データ・グループ

1259 ページの『partition_key - パーティション・キーのモニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

1110 ページの『invocation_id - 呼び出し ID : モニター・エレメント』

1205 ページの『nesting_level - ネスト・レベル : モニター・エレメント』

1247 ページの『package_elapsed_time - パッケージ経過時間 : モニター・エレメント』

1247 ページの『package_id - パッケージ ID : モニター・エレメント』

1460 ページの『routine_id - ルーチン ID : モニター・エレメント』

1711 ページの『uow_id 作業単位 ID : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

utillocation 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

1729 ページの『utility_type ユーティリティー・タイプ』

983 ページの『device_type - 装置タイプ』

1127 ページの『location_type - ロケーション・タイプ・』

1127 ページの『location - ロケーション』

utilphase 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』

1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』

1193 ページの『member - データベース・メンバー・モニター・エレメント』

1012 ページの『event_type - イベント・タイプ・モニター・エレメント』

1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』

1729 ページの『utility_type ユーティリティー・タイプ』

1726 ページの『utility_phase_type - ユーティリティー・フェーズ・タイプ』

1262 ページの『phase_start_event_id - フェーズ開始イベント ID』

1262 ページの『phase_start_event_timestamp - フェーズ開始イベントのタイム・スタンプ』

1237 ページの『objtype - オブジェクト・タイプ : モニター・エレメント』
1232 ページの『object_schema - オブジェクト・スキーマのモニター・エレメント』
1231 ページの『object_name - オブジェクト名モニター・エレメント』
1726 ページの『utility_phase_detail - ユーティリティー・フェーズ詳細』

utilstart 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』
1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』
1193 ページの『member - データベース・メンバー・モニター・エレメント』
1012 ページの『event_type - イベント・タイプ・モニター・エレメント』
1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』
1729 ページの『utility_type ユーティリティー・タイプ』
1724 ページの『utility_operation_type - ユーティリティー操作のタイプ』
1724 ページの『utility_invoker_type - ユーティリティー呼び出し側タイプ』
1726 ページの『utility_priority ユーティリティー優先度』
1727 ページの『utility_start_type - ユーティリティー開始タイプ』
1237 ページの『objtype - オブジェクト・タイプ : モニター・エレメント』
1232 ページの『object_schema - オブジェクト・スキーマのモニター・エレメント』
1231 ページの『object_name - オブジェクト名モニター・エレメント』
1221 ページの『num_tbsps - 表スペース数のモニター・エレメント』
1584 ページの『tbsp_names - 表スペース名』
1723 ページの『utility_detail - ユーティリティー詳細』

utilstop 論理データ・グループ

1010 ページの『event_id - イベント ID モニター・エレメント』
1012 ページの『event_timestamp - イベント・タイム・スタンプ : モニター・エレメント』
1193 ページの『member - データベース・メンバー・モニター・エレメント』
1012 ページの『event_type - イベント・タイプ・モニター・エレメント』
1723 ページの『utility_invocation_id - ユーティリティー呼び出し ID』
1729 ページの『utility_type ユーティリティー・タイプ』
1728 ページの『utility_stop_type - ユーティリティー停止タイプ』
1522 ページの『start_event_id - 開始イベント ID』
1522 ページの『start_event_timestamp - 開始イベント・タイム・スタンプ』
1510 ページの『sqlca SQL 連絡域 (SQLCA)』

イベント・タイプの論理データ・グループへのマッピング

ファイルおよびパイプのイベント・モニターに関しては、イベント・モニターの出力は、順序付けされた一連の論理データ・グループから成っています。どのイベント・モニター・タイプの場合にも、出力レコードには必ず同じ開始論理データ・グループが含まれています。

論理データ・グループが示すフレームは、イベント・モニターによって記録されるイベント・タイプによって異なります。

ファイルおよびパイプ・イベント・モニターの場合、イベント・レコードはどの接続についても生成されることがあるため、ストリーム中にいろいろな順序で現れる場合があります。すなわち、接続 1 のトランザクション・イベントの直後に接続 2 の接続イベントを取得することがあるということです。しかし、単一の接続または単一のイベントに属するレコードは、論理順序で現れます。例えば、ステートメント・レコード (ステートメントの終わり) は、トランザクション・レコード (UOW の終わり) があれば、必ずその前に来ます。同様に、デッドロック・イベント・レコードは、必ずデッドロックに関係する各接続のデッドロック接続イベント・レコードの前に来ます。**アプリケーション ID** または**アプリケーション・ハンドル (agent_id)** を使って、レコードと接続を一致させることができます。

接続ヘッダー・イベントは通常、データベースへのそれぞれの接続ごとに書き込まれます。詳細付きデッドロック・イベント・モニターの場合は、デッドロックが生じたときにだけ書き込まれます。この場合、接続ヘッダー・イベントは、デッドロックに関係したものについてのみ書き込まれます。データベースへのすべての接続について書き込まれるわけではありません。

論理データ・グループは、4 つの異なるレベルに従って配列されます。すなわちモニター、プロローグ、内容、およびエピローグです。以下は、各レベルの詳細な記述です。対応するイベント・タイプおよび論理データ・グループも示しています。

モニター

モニター・レベルの情報は、すべてのイベント・モニターに対して生成されます。これは、イベント・モニターのメタデータから成っています。

表 146. イベント・モニター・データ・ストリーム : モニター・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
モニター・レベル	event_log_stream_header	イベント・モニターのバージョン・レベルおよびバイトの並び順を識別する。アプリケーションはこのヘッダーを使用して、 evmon 出力ストリームを処理できるかどうかを判別できます。

プロローグ

プロローグ情報は、イベント・モニターが活動化されると生成されます。

表 147. イベント・モニター・データ・ストリーム : プロローグ・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
ログ・ヘッダー	event_log_header	トレースの特性、例えば、サーバーのタイプおよびメモリーのレイアウト。
データベース・ヘッダー	event_db_header	データベース名、パスおよび活動化時間。
イベント・モニター開始	event_start	モニターが開始されるか再始動された時間。
接続ヘッダー	event_connheader	現行接続のヘッダーごとに 1 つ。接続時間とアプリケーション名を含みます。イベント接続ヘッダーが生成されるのは、接続、ステートメント、トランザクション、およびデッドロックの各イベント・モニターに対してのみです。詳細付きデッドロック・イベント・モニターは、デッドロックが生じたときだけ接続ヘッダーを生成します。

内容

イベント・モニターの指定されたイベント・タイプに固有の情報は、内容セクションに表示されます。

表 148. イベント・モニター・データ・ストリーム : 内容セクション

イベント・タイプ	論理データ・グループ	入手できる情報
ステートメント・イベント	event_stmt	ステートメント・レベル・データ。動的ステートメントのテキストを含む。ステートメント・イベント・モニターは、フェッチのログを取りません。
サブセクション・イベント	event_subsection	サブセクション・レベル・データ。
トランザクション・イベント[1]	event_xact	トランザクション・レベル・データ。
接続イベント	event_conn	接続レベル・データ。
デッドロック・イベント	event_deadlock	デッドロック・レベル・データ。
デッドロック接続イベント	event_dlconn	デッドロックに関係している接続ごとに 1 つ。関係するアプリケーションと競合しているロックを含みます。

表 148. イベント・モニター・データ・ストリーム : 内容セクション (続き)

イベント・タイプ	論理データ・グループ	入手できる情報
詳細付きデッドロック接続イベント	event_detailed_dlconn、lock	デッドロックに関係している接続ごとに 1 つ。関係するアプリケーション、競合しているロック、現在のステートメント情報、およびアプリケーション競合によって保持された他のロックを含みます。
オーバーフロー	event_overflow	脱落したレコードの数。書き込み装置が (ブロック化されていない) イベント・モニターに追い付かないときに生成されます。
詳細付きデッドロック履歴[2]	event_stmt_history	デッドロックに関係する作業単位で実行されたステートメントのリスト。
詳細付きデッドロック履歴の値[2]	event_data_value	event_stmt_history リスト内のステートメント用のパラメーター・マーカー。
アクティビティー	event_activity	システムで実行が完了した、または完了前にキャプチャーされたアクティビティーのリスト。
	event_activitystmt	アクティビティー・タイプがステートメントであった際に、そのアクティビティーが実行したステートメントに関する情報。
	event_activityvals	SQL ステートメントである各アクティビティーの入力変数として使用されるデータ値。こうしたデータ値には、LOB データ、LONG データ、または構造化タイプ・データは含まれません。
統計	event_scstats	システム内のそれぞれのサービス・クラス、処理クラス、またはワークロードで実行されたアクティビティーから算出される統計、さらにしきい値キューから算出される統計。
	event_wcstats	
	event_wlstats	
	event_qstats	
	event_histogrambin	
しきい値違反	event_thresholdviolations	違反したしきい値とその時間を示す情報。

1 このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用して、トランザクション・イベントをモニターしてください。

2 このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT

MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

エピローグ

エピローグ情報は、データベースが非アクティブ化状態にあるとき (最後のアプリケーションが切断を終了したとき) に生成されます。

表 149. イベント・モニター・データ・ストリーム : エピローグ・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
データベース・イベント	event_db	データベース・マネージャー・レベル・データ。
バッファー・プール・イベント	event_bufferpool	バッファー・プール・レベル・データ。
表スペース・イベント	event_tablespace	表スペース・レベル・データ。
表イベント	event_table	表レベル・データ。

COLLECT ACTIVITY DATA 設定の影響を受ける論理データ・グループ

以下の表は、サービス・サブクラス、ワークロード、作業クラス (作業アクションを介するもの)、およびしきい値を含むすべてのタイプの WLM オブジェクトにおいて、COLLECT ACTIVITY DATA のさまざまなオプションが指定された場合にどの論理データ・グループが収集されるかを示しています。

表 150. COLLECT ACTIVITY DATA 設定

COLLECT ACTIVITY DATA の設定	収集される論理データ・グループ
NONE	none
WITHOUT DETAILS	event_activity event_activitymetrics
WITH DETAILS	event_activity event_activitymetrics event_activitystmt
WITH DETAILS AND VALUES	event_activity event_activitymetrics event_activitystmt event_activityvals

スナップショット・モニター・インターフェースの論理データ・グループへのマッピング

それぞれの API 要求タイプ、CLP コマンド、および SQL 管理ビューは、すべての論理データ・グループのサブセットからのモニター・データのみをキャプチャーします。

次の表では、スナップショット・モニター・データにアクセスするためのいくつかの方法をリストしています。すべてのスナップショット・モニター・データは、モニター・エレメント内に保管され、論理データ・グループによってカテゴリー化されます。

この表にリストされている、それぞれの API 要求タイプ、CLP コマンド、および SQL 管理ビューは、右端の列にリストされている論理データ・グループからのモニター・エレメントを戻します。

注:

1. 対応する SQL 管理ビューがない、いくつかの API 要求タイプおよび CLP コマンドがあります。その他の API 要求タイプおよび CLP コマンドの場合、それぞれの SQL 管理ビューは、関連した論理データ・グループのサブセットをキャプチャーします。
2. 一部のモニター・エレメントは、関連したモニター・スイッチが ON に設定されている場合にだけ戻されます。スイッチで必須エレメントを制御できるかどうか判別するには、個々のモニター・エレメントを参照してください。

表 151. スナップショット・モニター・インターフェースの論理データ・グループへのマッピング

db2GetSnapshot API 要求タイプ	CLP コマンド	SQL 管理ビュー	論理データ・グループ
SQLMA_APPLINFO_ALL	list applications [show detail]	APPLICATIONS	appl_info
SQLMA_DBASE_APPLINFO	list applications for database *dbname* [show detail]	APPLICATIONS	appl_info
SQLMA_DCS_APPLINFO_ALL	list dcs applications [show detail]		dcs_appl_info
SQLMA_DB2	get snapshot for dbm	SNAPDBM	db2
		SNAPFCM	fcm
		SNAPFCMPART	fcm_node
		SNAPUTIL	utility_info
		SNAPUTIL_PROGRESS	progress、progress_info
		SNAPDBM_MEMORY_POOL	memory_pool
	get dbm monitor switches	SNAPSWITCHES	switch_list
SQLMA_DBASE	get snapshot for database on *dbname*	SNAPDB	dbase
		SNAPDETAILLOG	detail_log
		ADMIN_GET_STORAGE_PATHS	db_storage_group
			rollforward
			db_sto_path_info
		SNAPTBSP	tablespace
		SNAPDB_MEMORY_POOL	memory_pool

表 151. スナップショット・モニター・インターフェースの論理データ・グループへのマッピング (続き)

db2GetSnapshot API 要求タイプ	CLP コマンド	SQL 管理ビュー	論理データ・グループ
SQLMA_DBASE_ALL	get snapshot for all databases	SNAPDB	dbase
		ADMIN_GET_STORAGE_PATHS	db_storage_group
			rollforward
			db_sto_path_info
		SNAPTBSP	tablespace
		SNAPDB_MEMORY_POOL	memory_pool
	list active databases		dbase
SQLMA_DCS_DBASE	get snapshot for dcs database on *dbname*		dcs_dbase、stmt_transmissions
SQLMA_DCS_DBASE_ALL	get snapshot for all dcs databases		dcs_dbase、stmt_transmissions
SQLMA_DBASE_REMOTE	get snapshot for remote database on *dbname*		dbase_remote
SQLMA_DBASE_REMOTE_ALL	get snapshot for all remote databases		dbase_remote
SQLMA_APPL	get snapshot for application applid *appl-id*	SNAPAPPL	appl
		SNAPAGENT	agent
		SNAPAPPL_INFO	appl_info
		SNAPLOCKWAIT	lock_wait
		SNAPSTMT	stmt
		SNAPSUBSECTION	subsection
		SNAPAGENT_MEMORY_POOL	memory_pool
SQLMA_AGENT_ID	get snapshot for application agentid *appl-handle*	SNAPAGENT	appl
		SNAPAGENT	agent
		SNAPAPPL_INFO	appl_info
		SNAPLOCKWAIT	lock_wait
		SNAPSTMT	stmt
		SNAPSUBSECTION	subsection
		SNAPAGENT_MEMORY_POOL	memory pool
SQLMA_DBASE_APPLS	get snapshot for applications on *dbname*	SNAPAPPL	appl
		SNAPAGENT	agent
		SNAPAPPL_INFO	appl_info
		SNAPLOCKWAIT	lock_wait
		SNAPSTMT	stmt
		SNAPSUBSECTION	subsection
		SNAPAGENT_MEMORY_POOL	memory_pool

表 151. スナップショット・モニター・インターフェースの論理データ・グループへのマッピング (続き)

db2GetSnapshot API 要求タイプ	CLP コマンド	SQL 管理ビュー	論理データ・グループ
SQLMA_APPL_ALL	get snapshot for all applications	SNAPAPPL	appl
		SNAPAPPL_INFO	appl_info
		SNAPLOCKWAIT	lock_wait
		SNAPSTATEMENT	stmt
		SNAPAGENT	agent
		SNAPSUBSECTION	subsection
		SNAPAGENT_MEMORY_POOL	memory_pool
SQLMA_DCS_APPL	get snapshot for dcs application applid *appl-id*		dcs_appl、dcs_stmt、dcs_appl_info、stmt_transmissions
SQLMA_DCS_APPL_ALL	get snapshot for all dcs applications		dcs_appl、dcs_stmt、dcs_appl_info、stmt_transmissions
SQLMA_DCS_APPL_HANDLE	get snapshot for dcs application agentid *appl-handle*		dcs_appl、dcs_stmt、dcs_appl_info、stmt_transmissions
SQLMA_DCS_DBASE_APPLS	get snapshot for dcs applications on *dbname*		dcs_appl、dcs_stmt、dcs_appl_info、stmt_transmissions
SQLMA_DBASE_APPLS_REMOTE	get snapshot for remote applications on *dbname*		dbase_appl
SQLMA_APPL_REMOTE_ALL	get snapshot for all remote applications		dbase_appl
SQLMA_DBASE_TABLES	get snapshot for tables on *dbname*	SNAPTAB	table
		SNAPTAB_REORG	table_reorg
			table_list
SQLMA_APPL_LOCKS	get snapshot for locks for application applid *appl-id*	SNAPLOCK、SNAPAPPL、SNAPLOCKWAIT	appl_lock_list、lock_wait、lock
SQLMA_APPL_LOCKS_AGENT_ID	get snapshot for locks for application agentid *appl-handle*	SNAPLOCK、SNAPAPPL、SNAPLOCKWAIT	appl_lock_list、lock_wait、lock
SQLMA_DBASE_LOCKS	get snapshot for locks on *dbname*	SNAPLOCK	appl_lock_list、lock
		SNAPLOCK、SNAPLOCKWAIT	db_lock_list、lock_wait

表 151. スナップショット・モニター・インターフェースの論理データ・グループへのマッピング (続き)

db2GetSnapshot API 要求タイプ	CLP コマンド	SQL 管理ビュー	論理データ・グループ
SQLMA_DBASE_TABLESPACES	get snapshot for tablespaces on *dbname*	SNAPTBSP	tablespace
		SNAPTBSPPART	tablespace、 tablespace_nodeinfo
		SNAPTBSP_QUIESCER	tablespace_quiescer、 tablespace_nodeinfo
		SNAPCONTAINER	tablespace_container、 tablespace_nodeinfo
		SNAPTBSP_RANGE	tablespace_ranges、 tablespace_nodeinfo
			tablespace_list、 tablespace_nodeinfo
SQLMA_BUFFERPOOLS_ALL	get snapshot for all bufferpools	SNAPBP	bufferpool
SQLMA_DBASE_BUFFERPOOLS	get snapshot for bufferpools on *dbname*	SNAPBP	bufferpool
SQLMA_DYNAMIC_SQL	get snapshot for dynamic sql on *dbname*	SNAPDYN_SQL	dynsql
			dynsql_list

以下の図は、論理データ・グループがスナップショット・データ・ストリーム内に現れる順番を示しています。

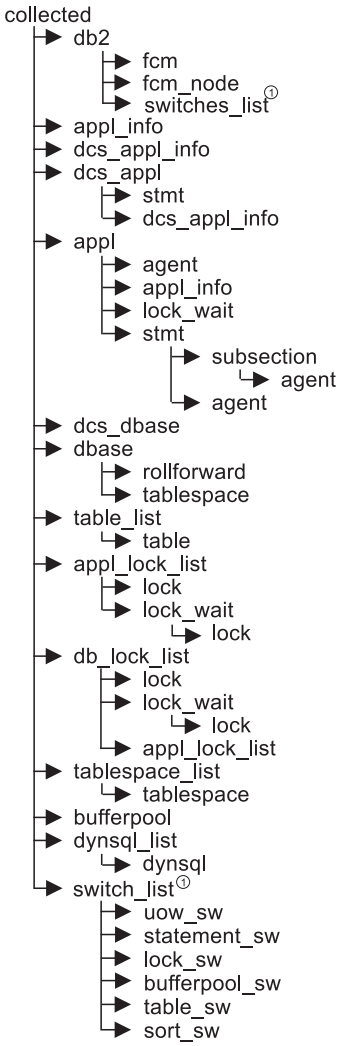


図 18. データ・ストリーム階層

注: 論理データ・グループの一部として、時間が戻されることがあります。

スナップショット・モニターの論理データ・グループおよびモニター・エレメント

次のセクションでは、論理データ・グループと、スナップショット・モニターによって戻されるモニター・エレメントをリストしています。

- 768 ページの『agent 論理データ・グループ』
- 769 ページの『appl 論理データ・グループ』
- 772 ページの『appl_id_info 論理データ・グループ』
- 773 ページの『appl_info 論理データ・グループ』
- 774 ページの『appl_lock_list 論理データ・グループ』
- 774 ページの『appl_remote 論理データ・グループ』
- 775 ページの『bufferpool 論理データ・グループ』

- 777 ページの『bufferpool_nodeinfo 論理データ・グループ』
- 777 ページの『collected 論理データ・グループ』
- 777 ページの『db2 論理データ・グループ』
- 778 ページの『db_lock_list 論理データ・グループ』
- 779 ページの『dbase 論理データ・グループ』
- 784 ページの『dbase_remote 論理データ・グループ』
- 785 ページの『db_storage_group 論理データ・グループ』
- 785 ページの『dcs_appl 論理データ・グループ』
- 787 ページの『dcs_appl_info 論理データ・グループ』
- 788 ページの『dcs_dbase 論理データ・グループ』
- 790 ページの『dcs_stmt 論理データ・グループ』
- 791 ページの『detail_log 論理データ・グループ』
- 791 ページの『dynsql 論理データ・グループ』
- 792 ページの『dynsql_list 論理データ・グループ』
- 792 ページの『fcm 論理データ・グループ』
- 793 ページの『fcm_node 論理データ・グループ』
- 793 ページの『hadr 論理データ・グループ』
- 793 ページの『lock 論理データ・グループ』
- 794 ページの『lock_wait 論理データ・グループ』
- 795 ページの『memory_pool 論理データ・グループ』
- 795 ページの『progress 論理データ・グループ』
- 795 ページの『progress_list 論理データ・グループ』
- 795 ページの『rollforward 論理データ・グループ』
- 795 ページの『stmt 論理データ・グループ』
- 797 ページの『stmt_transmissions 論理データ・グループ』
- 799 ページの『subsection 論理データ・グループ』
- 799 ページの『table 論理データ・グループ』
- 800 ページの『table_list 論理データ・グループ』
- 800 ページの『table_reorg 論理データ・グループ』
- 800 ページの『tablespace 論理データ・グループ』
- 803 ページの『tablespace_container 論理データ・グループ』
- 803 ページの『tablespace_list 論理データ・グループ』
- 803 ページの『tablespace_nodeinfo 論理データ・グループ』
- 804 ページの『tablespace_quiescer 論理データ・グループ』
- 804 ページの『tablespace_range 論理データ・グループ』
- 805 ページの『utility_info 論理データ・グループ』

agent 論理データ・グループ

826 ページの『agent_pid エンジン・ディスパッチ可能単位 (EDU) ID : モニター・エレメント』

appl 論理データ・グループ

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1105 ページの『int_deadlock_rollbacks デッドロックによる内部ロールバック』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1121 ページの『last_reset 最後のリセット・タイム・スタンプ』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1149 ページの『lock_timeout_val ロック・タイムアウト値 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1163 ページの『locks_held - ロック保持数 : モニター・エレメント』

1165 ページの『locks_waiting - ロックで待機中の現行エージェント : モニター・エレメント』

1208 ページの『num_agents ステートメントで作動しているエージェントの数』

1238 ページの『olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント』

1239 ページの『open_loc_curs 開かれているローカル・カーソル』

1239 ページの『open_loc_curs_blk 開かれているローカル・ブロック・カーソル』

1240 ページの『open_rem_curs 開かれているリモート・カーソル』

1240 ページの『open_rem_curs_blk 開かれているリモート・ブロック・カーソル』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1420 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

1424 ページの『prev_uow_stop_time 直前の作業単位完了タイム・スタンプ』

1425 ページの『priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数』

1426 ページの『priv_workspace_section_inserts 専用ワークスペース・セクション挿入』

1427 ページの『priv_workspace_section_lookups 専用ワークスペース・セクション検索』

1427 ページの『priv_workspace_size_top 専用ワークスペースの最大サイズ』

1444 ページの『rej_curs_blk リジェクトされたブロック・カーソル要求』

1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』

1462 ページの『rows_deleted - 削除行数 : モニター・エレメント』

1463 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1471 ページの『rows_selected 選択行数』

1471 ページの『rows_updated - 更新行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1479 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

1489 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

1490 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

1490 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

1491 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1509 ページの『sql_reqs_since_commit 最終コミット後の SQL 要求』

1523 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

1634 ページの『total_hash_joins ハッシュ結合の合計』

1635 ページの『total_hash_loops ハッシュ・ループの合計』

1644 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1708 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

1709 ページの『uow_comp_status 作業単位完了状況』

1710 ページの『uow_elapsed_time 最新の作業単位の経過時間』

1713 ページの『uow_lock_wait_time - ロック待機中の作業単位の合計時間 : モニター・エレメント』

1713 ページの『uow_log_space_used - 使用されている作業単位ログ・スペース: モニター・エレメント』

1714 ページの『uow_start_time - 作業単位開始タイム・スタンプ : モニター・エレメント』

1715 ページの『uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント』

1742 ページの『x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント』

1744 ページの『xquery_stmts - 試行された XQuery ステートメント』

appl_id_info 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

850 ページの『appl_status アプリケーション状況 : モニター・エレメント』

866 ページの『auth_id 許可 ID』

895 ページの『client_db_alias アプリケーションで使用するデータベース別名』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

904 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

1100 ページの『input_db_alias 入力データベース別名』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1527 ページの『status_change_time アプリケーション状況変更時刻』

appl_info 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

849 ページの『appl_section_inserts セクション挿入数 : モニター・エレメント』

850 ページの『appl_section_lookups - セクション検索』

850 ページの『appl_status アプリケーション状況 : モニター・エレメント』

866 ページの『auth_id 許可 ID』

866 ページの『authority_bitmap ユーザー許可レベル : モニター・エレメント』

867 ページの『authority_lvl ユーザー許可レベル : モニター・エレメント』

895 ページの『client_db_alias アプリケーションで使用するデータベース別名』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

904 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

938 ページの『coord_agent_pid コーディネーター・エージェント ID : モニター・エレメント』

940 ページの『coord_node コーディネーター・ノード』

941 ページの『corr_token DRDA 相関トークン』

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

1023 ページの『execution_id ユーザー・ログイン ID』

1100 ページの『input_db_alias 入力データベース別名』

1117 ページの『is_system_appl システム・アプリケーション : モニター・エレメント』

1209 ページの『num_assoc_agents 関連したエージェント数』

1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』

1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』
1527 ページの『status_change_time アプリケーション状況変更時刻』
1592 ページの『territory_code データベース・テリトリー・コード』
1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』
1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』
1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』
1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』
1738 ページの『workload_id ワークロード ID : モニター・エレメント』

appl_lock_list 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』
842 ページの『appl_id - アプリケーション ID : モニター・エレメント』
847 ページの『appl_name アプリケーション名 : モニター・エレメント』
850 ページの『appl_status アプリケーション状況 : モニター・エレメント』
866 ページの『auth_id 許可 ID』
895 ページの『client_db_alias アプリケーションで使用するデータベース別名』
904 ページの『codepage_id アプリケーションで使用するコード・ページ ID』
1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』
1163 ページの『locks_held - ロック保持数 : モニター・エレメント』
1165 ページの『locks_waiting - ロックで待機中の現行エージェント : モニター・エレメント』
1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』
1488 ページの『session_auth_id セッション許可 ID : モニター・エレメント』
1527 ページの『status_change_time アプリケーション状況変更時刻』

appl_remote 論理データ・グループ

907 ページの『commit_sql_stmts - 試行されたコミット・ステートメント』
954 ページの『create_nickname ニックネーム作成回数』
955 ページの『create_nickname_time ニックネーム作成応答時間』
961 ページの『datasource_name データ・ソース名』
967 ページの『db_name データベース名 : モニター・エレメント』
981 ページの『delete_sql_stmts 削除回数』
981 ページの『delete_time 削除応答時間』
1024 ページの『failed_sql_stmts 失敗したステートメント操作』
1100 ページの『insert_sql_stmts 挿入回数』
1101 ページの『insert_time 挿入応答時間』

bufferpool 論理データ・グループ

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1277 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

1278 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

1279 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

1280 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

1281 ページの『pool_async_write_time - バッファー・プール非同期書き込み時間 : モニター・エレメント』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1286 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1343 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1391 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1407 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

1730 ページの『vectored_ios - ベクトル化入出力要求数 : モニター・エレメント』

bufferpool_nodeinfo 論理データ・グループ

873 ページの『bp_cur_buffsz - バッファー・プールの現行サイズ』

875 ページの『bp_new_buffsz 新規バッファー・プール・サイズ』

875 ページの『bp_pages_left_to_remove 除去残ページ数』

875 ページの『bp_tbsp_use_count バッファー・プールにマップされている表スペースの数』

1208 ページの『node_number ノード番号』

collected 論理データ・グループ

1208 ページの『node_number ノード番号』

1482 ページの『server_db2_type モニター対象 (サーバー) ノードのデータベース・マネージャーのタイプ』

1482 ページの『server_instance_name サーバー・インスタンス名』

1483 ページの『server_prdid - サーバー製品/バージョン ID』

1484 ページの『server_version サーバー・バージョン』

1600 ページの『time_stamp スナップショット時刻』

1600 ページの『time_zone_disp 時間帯変位』

db2 論理データ・グループ

832 ページの『agents_created_empty_pool - エージェント・プールが空のために作成されたエージェント』

832 ページの『agents_from_pool - プールから割り当てられたエージェント』

833 ページの『agents_registered 登録済みエージェント』

833 ページの『agents_registered_top - エージェント最大登録数』

833 ページの『agents_stolen スチールされたエージェント』

834 ページの『agents_waiting_on_token - トークン待ちエージェント』

835 ページの『agents_waiting_top - エージェント最大待機数 : モニター・エレメント』

907 ページの『comm_private_mem - コミット済み専用メモリー』

910 ページの『con_local_dbases 現行接続を持つローカル・データベース』

939 ページの『coord_agents_top - コーディネーター・エージェント最大数』

965 ページの『db2start_time - データベース・マネージャー開始タイム・スタンプ』

968 ページの『db_status データベース状況 : モニター・エレメント』

1063 ページの『gw_cons_wait_client クライアントの要求送信を待機している接続の数』

1063 ページの『gw_cons_wait_host ホストの応答を待機している接続の数』

1063 ページの『gw_cur_cons DB2 Connect の現在の接続数』

1064 ページの『gw_total_cons DB2 Connect の接続試行合計回数』

1095 ページの『idle_agents - アイドル・エージェント数』

1121 ページの『last_reset 最後のリセット・タイム・スタンプ』

1125 ページの『local_cons - ローカル接続』

1125 ページの『local_cons_in_exec - データベース・マネージャーで実行中のローカル接続』

1176 ページの『max_agent_overflows 最大エージェント・オーバーフロー回数』

1213 ページの『num_gw_conn_switches - 接続切り替え回数』

1219 ページの『num_nodes_in_db2_instance パーティション内のノード数』

1262 ページの『piped_sorts_accepted 受け入れられたパイプ・ソート』

1263 ページの『piped_sorts_requested 要求されたパイプ・ソート数』

1412 ページの『post_threshold_hash_joins ハッシュ結合のしきい値』

1412 ページの『post_threshold_olap_funcs OLAP 関数のしきい値 : モニター・エレメント』

1418 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

1428 ページの『product_name - 製品名』

1445 ページの『rem_cons_in - データベース・マネージャーへのリモート接続』

1445 ページの『rem_cons_in_exec - データベース・マネージャーで実行中のリモート接続』

1486 ページの『service_level - サービス・レベル・』

1502 ページの『smallest_log_avail_node 使用可能なログ・スペースが最小のノード』

1502 ページの『sort_heap_allocated 割り振られたソート・ヒープの合計』

1503 ページの『sort_heap_top ソート専用ヒープの最高水準点』

db_lock_list 論理データ・グループ

854 ページの『appls_cur_cons - 現在接続されているアプリケーション』

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

1100 ページの『input_db_alias 入力データベース別名』

dbase 論理データ・グループ

987 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

989 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

991 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

994 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

996 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1004 ページの『dynamic_sql_stmts 試行された動的 SQL ステートメント』

1024 ページの『failed_sql_stmts 失敗したステートメント操作』

1057 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

1078 ページの『hash_join_overflows ハッシュ結合のオーバーフロー』

1079 ページの『hash_join_small_overflows ハッシュ結合の短精度オーバーフロー』

1100 ページの『input_db_alias 入力データベース別名』

1102 ページの『int_auto_rebinds 内部自動再バインド』

1103 ページの『int_commits - 内部コミット数 : モニター・エレメント』

1105 ページの『int_deadlock_rollbacks デッドロックによる内部ロールバック』

1106 ページの『int_rollbacks - 内部ロールバック数 : モニター・エレメント』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1118 ページの『last_backup 最終バックアップ・タイム・スタンプ』

1121 ページの『last_reset 最後のリセット・タイム・スタンプ』

1132 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

1140 ページの『lock_list_in_use - 使用中のロック・リスト・メモリーの合計 : モニター・エレメント』

1149 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

1154 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

1159 ページの『lock_waits - ロック待機数 : モニター・エレメント』

1163 ページの『locks_held - ロック保持数 : モニター・エレメント』

1165 ページの『locks_waiting - ロックで待機中の現行エージェント : モニター・エレメント』

1170 ページの『log_held_by_dirty_pages ダーティー・ページによって占有されるログ・スペースの量』

1171 ページの『log_read_time ログ読み取り時間』

1172 ページの『log_reads 読み取られたログ・ページの数』

1172 ページの『log_to_redo_for_recovery リカバリーの場合に再実行されるログの量』

1173 ページの『log_write_time ログ書き込み時間』

1174 ページの『log_writes 書き込まれたログ・ページの数』

1209 ページの『num_assoc_agents 関連したエージェント数』

1210 ページの『num_db_storage_paths 自動ストレージ・パスの数』

1213 ページの『num_indoubt_trans 未確定トランザクション数』

1213 ページの『num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数』

1215 ページの『num_log_data_found_in_buffer ログ・データがバッファーにある回数』

1216 ページの『num_log_part_page_io 部分ログ・ページ書き込み数』

1216 ページの『num_log_read_io ログ読み取り数』

1217 ページの『num_log_write_io ログ書き込み数』

1238 ページの『olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント』

1264 ページの『pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント』

1265 ページの『pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント』

1268 ページの『pkg_cache_num_overflows パッケージ・キャッシュ・オーバーフロー』

1268 ページの『pkg_cache_size_top - パッケージ・キャッシュの最高水準点』

1271 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1277 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

1278 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

1279 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

1280 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

1281 ページの『pool_async_write_time - バッファー・プール非同期書き込み時間 : モニター・エレメント』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1410 ページの『post_shrthreshold_hash_joins ポストしきい値ハッシュ結合』

1410 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

1425 ページの『priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数』

1426 ページの『priv_workspace_section_inserts 専用ワークスペース・セクション挿入』

1427 ページの『priv_workspace_section_lookups 専用ワークスペース・セクション検索』

1427 ページの『priv_workspace_size_top 専用ワークスペースの最大サイズ』

1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』

1462 ページの『rows_deleted - 削除行数 : モニター・エレメント』

1463 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1471 ページの『rows_selected 選択行数』

1471 ページの『rows_updated - 更新行数 : モニター・エレメント』

1475 ページの『sec_log_used_top 使用された最大 2 次ログ・スペース』

1476 ページの『sec_logs_allocated 現在割り振られている 2 次ログ』

1479 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

1483 ページの『server_platform サーバーのオペレーティング・システム』

1489 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

1490 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

1490 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

1491 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

1502 ページの『sort_heap_allocated 割り振られたソート・ヒープの合計』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1506 ページの『sort_shrheap_allocated 現在割り振られているソート共有ヒープ』

1506 ページの『sort_shrheap_top ソート共有ヒープの最高水準点』

1523 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

1524 ページの『stats_cache_size - 統計キャッシュのサイズ : モニター・エレメント』

1525 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間 : モニター・エレメント』

1526 ページの『stats_fabrications - 統計作成の合計数 : モニター・エレメント』

1551 ページの『sync_runstats - 同期 RUNSTATS アクティビティーの合計数 : モニター・エレメント』

1552 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント』

1601 ページの『tot_log_used_top 使用された最大合計ログ・スペース』

1617 ページの『total_cons データベース活動化以降の接続』

1634 ページの『total_hash_joins ハッシュ結合の合計』

1635 ページの『total_hash_loops ハッシュ・ループの合計』

1643 ページの『total_log_available 使用可能なログの合計』

1643 ページの『total_log_used 使用されているログ・スペースの合計』

1644 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

1669 ページの『total_sec_cons 2 次接続』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1680 ページの『total_sorts - ソート合計 : モニター・エレメント』

1708 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

1708 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

1742 ページの『x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント』

1744 ページの『xquery_stmts - 試行された XQuery ステートメント』

dbase_remote 論理データ・グループ

907 ページの『commit_sql_stmts - 試行されたコミット・ステートメント』

954 ページの『create_nickname ニックネーム作成回数』

955 ページの『create_nickname_time ニックネーム作成応答時間』

961 ページの『datasource_name データ・ソース名』

967 ページの『db_name データベース名 : モニター・エレメント』

981 ページの『delete_sql_stmts 削除回数』

981 ページの『delete_time 削除応答時間』

1003 ページの『disconnects 切断回数』

1024 ページの『failed_sql_stmts 失敗したステートメント操作』

1100 ページの『insert_sql_stmts 挿入回数』

1101 ページの『insert_time 挿入応答時間』

1260 ページの『passthru_time パススルー時間』

1261 ページの『passthrus パススルー数』

1446 ページの『remote_lock_time リモート・ロック時間』

1446 ページの『remote_locks リモート・ロック数』

1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』

1462 ページの『rows_deleted - 削除行数 : モニター・エレメント』

1463 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

1471 ページの『rows_selected 選択行数』

1471 ページの『rows_updated - 更新行数 : モニター・エレメント』

1479 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

1480 ページの『select_time 照会応答時間』

1507 ページの『sp_rows_selected ストアード・プロシージャーによって戻された行数』

1549 ページの『stored_proc_time ストアード・プロシージャー時間』

1549 ページの『stored_procs ストアード・プロシージャー数』

1617 ページの『total_cons データベース活動化以降の接続』

1717 ページの『update_sql_stmts 更新回数』

1718 ページの『update_time 更新応答時間』

db_storage_group 論理データ・グループ

1059 ページの『fs_id - 固有のファイル・システム識別番号 : モニター・エレメント』

1060 ページの『fs_total_size - ファイル・システムの合計サイズ : モニター・エレメント』

1060 ページの『fs_used_size - ファイル・システム上で使用されるスペースの量 : モニター・エレメント』

1208 ページの『node_number ノード番号』

1547 ページの『sto_path_free_size 自動ストレージ・パスのフリー・スペース : モニター・エレメント』

dcs_appl 論理データ・グループ

846 ページの『appl_idle_time アプリケーション・アイドル時間』

907 ページの『commit_sql_stmts - 試行されたコミット・ステートメント』

1007 ページの『elapsed_exec_time ステートメント実行経過時間』

1024 ページの『failed_sql_stmts 失敗したステートメント操作』

1062 ページの『gw_con_time DB2 Connect ゲートウェイの最初の接続開始』

1064 ページの『gw_exec_time DB2 Connect ゲートウェイ処理の経過時間』

1084 ページの『host_response_time ホスト応答時間』

1096 ページの『inbound_bytes_received 受信されたインバウンド・バイト数』

1096 ページの『inbound_bytes_sent 送信されたインバウンド・バイト数』

1121 ページの『last_reset 最後のリセット・タイム・スタンプ』

1179 ページの『max_data_received_1024 受信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

1180 ページの『max_data_received_128 受信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数』

1180 ページの『max_data_received_16384 受信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数』

1181 ページの『max_data_received_2048 受信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数』

1181 ページの『max_data_received_256 受信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数』

1192 ページの『max_network_time_gt500_ms ネットワーク時間が 500 ミリ秒を超えるステートメント数』

1205 ページの『network_time_bottom ステートメントの最小ネットワーク時間』

1206 ページの『network_time_top ステートメントの最大ネットワーク時間』

1239 ページの『open_cursors オープン・カーソル数』

1243 ページの『outbound_bytes_received 受信されたアウトバウンド・バイト数』

1244 ページの『outbound_bytes_sent 送信されたアウトバウンド・バイト数』

1424 ページの『prev_uow_stop_time 直前の作業単位完了タイム・スタンプ』

1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』

1471 ページの『rows_selected 選択行数』

1509 ページの『sql_stmts 試行された SQL ステートメントの数』

1695 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

1696 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

1696 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

1697 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

1709 ページの『uow_comp_status 作業単位完了状況』

1710 ページの『uow_elapsed_time 最新の作業単位の経過時間』

1714 ページの『uow_start_time - 作業単位開始タイム・スタンプ : モニター・エレメント』

1715 ページの『uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント』

1744 ページの『xid トランザクション ID』

dcs_appl_info 論理データ・グループ

824 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

827 ページの『agent_status DCS アプリケーション・エージェント』

842 ページの『appl_id - アプリケーション ID : モニター・エレメント』

847 ページの『appl_name アプリケーション名 : モニター・エレメント』

866 ページの『auth_id 許可 ID』

898 ページの『client_pid - クライアント・プロセス ID : モニター・エレメント』

899 ページの『client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント』

900 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

901 ページの『client_protocol - クライアント通信プロトコル : モニター・エレメント』

904 ページの『codepage_id アプリケーションで使用するコード・ページ ID』
974 ページの『dcs_appl_status DCS アプリケーション状況 : モニター・エレメント』
975 ページの『dcs_db_name DCS データベース名』
1023 ページの『execution_id ユーザー・ログイン ID』
1064 ページの『gw_db_alias ゲートウェイでのデータベース別名』
1082 ページの『host_ccsid ホスト・コード化文字セット ID』
1082 ページの『host_db_name ホスト・データベース名』
1083 ページの『host_prdid - ホスト製品/バージョン ID』
1096 ページの『inbound_comm_address - インバウンド通信アドレス』
1242 ページの『outbound_appl_id アウトバウンド・アプリケーション ID』
1245 ページの『outbound_comm_address アウトバウンド通信アドレス』
1245 ページの『outbound_comm_protocol アウトバウンド通信プロトコル』
1246 ページの『outbound_sequence_no アウトバウンド・シーケンス番号』
1480 ページの『sequence_no シーケンス番号 : モニター・エレメント』
1527 ページの『status_change_time アプリケーション状況変更時刻』

dcs_dbase 論理データ・グループ

907 ページの『commit_sql_stmts - 試行されたコミット・ステートメント』
910 ページの『con_elapsed_time 最新の接続経過時間』
911 ページの『con_response_time 接続の最新応答時間』
975 ページの『dcs_db_name DCS データベース名』
1007 ページの『elapsed_exec_time ステートメント実行経過時間』
1024 ページの『failed_sql_stmts 失敗したステートメント操作』
1061 ページの『gw_comm_error_time 通信エラー時刻』
1062 ページの『gw_comm_errors 通信エラー』
1062 ページの『gw_con_time DB2 Connect ゲートウェイの最初の接続開始』
1062 ページの『gw_connections_top ホスト・データベースへの同時接続の最大数』
1063 ページの『gw_cons_wait_client クライアントの要求送信を待機している接続の数』
1063 ページの『gw_cons_wait_host ホストの応答を待機している接続の数』
1063 ページの『gw_cur_cons DB2 Connect の現在の接続数』
1064 ページの『gw_total_cons DB2 Connect の接続試行合計回数』
1082 ページの『host_db_name ホスト・データベース名』
1084 ページの『host_response_time ホスト応答時間』
1096 ページの『inbound_bytes_received 受信されたインバウンド・バイト数』
1121 ページの『last_reset 最後のリセット・タイム・スタンプ』
1179 ページの『max_data_received_1024 受信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

1190 ページの『max_network_time_16_ms ネットワーク時間が 4 から 16 ミリ秒のステートメント数』
1191 ページの『max_network_time_1_ms ネットワーク時間が 1 ミリ秒以下のステートメント数』
1191 ページの『max_network_time_4_ms ネットワーク時間が 1 から 4 ミリ秒のステートメント数』
1192 ページの『max_network_time_500_ms ネットワーク時間が 100 から 500 ミリ秒のステートメント数』
1192 ページの『max_network_time_gt500_ms ネットワーク時間が 500 ミリ秒を超えるステートメント数』
1205 ページの『network_time_bottom ステートメントの最小ネットワーク時間』
1206 ページの『network_time_top ステートメントの最大ネットワーク時間』
1244 ページの『outbound_bytes_sent 送信されたアウトバウンド・バイト数』
1457 ページの『rollback_sql_stmts - 試行されたロールバック・ステートメント』
1471 ページの『rows_selected 選択行数』
1509 ページの『sql_stmts 試行された SQL ステートメントの数』

dcs_stmt 論理データ・グループ

872 ページの『blocking_cursor ブロック・カーソル』
955 ページの『creator アプリケーション作成者』
1007 ページの『elapsed_exec_time ステートメント実行経過時間』
1056 ページの『fetch_count 成功したフェッチの数』
1064 ページの『gw_exec_time DB2 Connect ゲートウェイ処理の経過時間』
1084 ページの『host_response_time ホスト応答時間』
1096 ページの『inbound_bytes_received 受信されたインバウンド・バイト数』
1096 ページの『inbound_bytes_sent 送信されたインバウンド・バイト数』
1222 ページの『num_transmissions 伝送回数』
1222 ページの『num_transmissions_group 伝送グループの回数』
1243 ページの『outbound_bytes_received 受信されたアウトバウンド・バイト数』
1244 ページの『outbound_bytes_sent 送信されたアウトバウンド・バイト数』
1248 ページの『package_name - パッケージ名 : モニター・エレメント』
1432 ページの『query_card_estimate 照会行数の見積もり』
1433 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』
1477 ページの『section_number - セクション番号 : モニター・エレメント』
1527 ページの『stmt_elapsed_time 最新のステートメント経過時間』
1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』
1537 ページの『stmt_start ステートメント操作開始タイム・スタンプ』
1537 ページの『stmt_stop ステートメント操作停止タイム・スタンプ』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

detail_log 論理データ・グループ

957 ページの『current_active_log 現行アクティブ・ログ・ファイル番号』

958 ページの『current_archive_log 現行アーカイブ・ログ・ファイル番号』

1058 ページの『first_active_log 先頭アクティブ・ログ・ファイル番号』

1118 ページの『last_active_log 最終アクティブ・ログ・ファイル番号』

1208 ページの『node_number ノード番号』

dynsql 論理データ・グループ

1056 ページの『fetch_count 成功したフェッチの数』

1102 ページの『insert_timestamp - 挿入タイムスタンプ : モニター・エレメント』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1209 ページの『num_compilations - ステートメント・コンパイル数』

1211 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1424 ページの『prep_time_best ステートメント最短準備時間 : モニター・エレメント』

1424 ページの『prep_time_worst ステートメント最長準備時間 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1525 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間 : モニター・エレメント』

1534 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント』

1536 ページの『stmt_sorts ステートメント・ソート回数』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1552 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント』

1629 ページの『total_exec_time ステートメント実行の経過時間 : モニター・エレメント』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

1690 ページの『total_sys_cpu_time ステートメントのシステム CPU の合計時間 : モニター・エレメント』

1693 ページの『total_usr_cpu_time ステートメントのユーザー CPU の合計時間 : モニター・エレメント』

dynsql_list 論理データ・グループ

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

fcm 論理データ・グループ

876 ページの『buff_free - 現在空いている FCM バッファー』

876 ページの『buff_free_bottom 空き FCM バッファーの最小数』

877 ページの『buff_max - FCM バッファーの最大可能数 : モニター・エレメント』

877 ページの『buff_total - 現在割り振られている FCM バッファー数 : モニター・エレメント』

891 ページの『ch_free - 現在空いているチャネル』

892 ページの『ch_free_bottom 空いているチャネルの最小』

892 ページの『ch_max - FCM チャネルの最大可能数 : モニター・エレメント』

892 ページの『ch_total - 現在割り振られている FCM チャネル数 : モニター・エレメント』

fcm_node 論理データ・グループ

925 ページの『connection_status - 接続状況』

1208 ページの『node_number ノード番号』

1609 ページの『total_buffers_rcvd 受信された FCM バッファーの合計』

1610 ページの『total_buffers_sent 送信された FCM バッファーの合計』

hadr 論理データ・グループ

1065 ページの『hadr_connect_status HADR 接続状況 : モニター・エレメント』

1066 ページの『hadr_connect_time HADR 接続時刻 : モニター・エレメント』

1066 ページの『hadr_heartbeat HADR ハートビート : モニター・エレメント』

1067 ページの『hadr_local_host - HADR ローカル・ホスト : モニター・エレメント』

1068 ページの『hadr_local_service HADR ローカル・サービス : モニター・エレメント』

1069 ページの『hadr_log_gap HADR ログ・ギャップ』

1070 ページの『hadr_primary_log_file HADR 1 次ログ・ファイル : モニター・エレメント』

1071 ページの『hadr_primary_log_lsn HADR 1 次ログ LSN : モニター・エレメント』

1071 ページの『hadr_primary_log_page HADR 1 次ログ・ページ : モニター・エレメント』

1072 ページの『hadr_remote_host HADR リモート・ホスト : モニター・エレメント』

1073 ページの『hadr_remote_instance HADR リモート・インスタンス : モニター・エレメント』

1073 ページの『hadr_remote_service HADR リモート・サービス : モニター・エレメント』

1074 ページの『hadr_role HADR の役割』

1074 ページの『hadr_standby_log_file HADR スタンバイ・ログ・ファイル : モニター・エレメント』

1075 ページの『hadr_standby_log_lsn HADR スタンバイ・ログ LSN : モニター・エレメント』

1075 ページの『hadr_standby_log_page HADR スタンバイ・ログ・ページ : モニター・エレメント』

1076 ページの『hadr_state HADR の状態 : モニター・エレメント』

1077 ページの『hadr_syncmode HADR 同期モード : モニター・エレメント』

1078 ページの『hadr_timeout HADR タイムアウト : モニター・エレメント』

lock 論理データ・グループ

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1129 ページの『lock_count ロック・カウント : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1139 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1144 ページの『lock_object_name ロック対象名』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

1148 ページの『lock_status - ロック状況 : モニター・エレメント』

1208 ページの『node_number ノード番号』

1554 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

lock_wait 論理データ・グループ

826 ページの『agent_id_holding_lock ロックを保持しているエージェント ID』

845 ページの『appl_id_holding_lk ロックを保持しているアプリケーション ID』

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

1128 ページの『lock_attributes ロック属性 : モニター・エレメント』

1130 ページの『lock_current_mode - 変換前の元のロック・モード : モニター・エレメント』

1131 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

1140 ページの『lock_mode - ロック・モード : モニター・エレメント』

1142 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

1143 ページの『lock_name ロック名 : モニター・エレメント』

1145 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

1147 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

1153 ページの『lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント』

1208 ページの『node_number ノード番号』

1520 ページの『ss_number サブセクション番号 : モニター・エレメント』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1566 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

memory_pool 論理データ・グループ

1208 ページの『node_number ノード番号』

1287 ページの『pool_config_size メモリー・プールの構成済みサイズ』

1288 ページの『pool_cur_size メモリー・プールの現行サイズ』

1324 ページの『pool_id メモリー・プール ID』

1375 ページの『pool_secondary_id メモリー・プール 2 次 ID』

1390 ページの『pool_watermark メモリー・プール水準点』

progress 論理データ・グループ

1428 ページの『progress_completed_units 完了した進行作業単位』

1429 ページの『progress_description 進行の記述』

1430 ページの『progress_seq_num 進行シーケンス番号』

1430 ページの『progress_start_time 進行開始時刻』

1431 ページの『progress_total_units 合計進行作業単位』

1431 ページの『progress_work_metric 進行作業メトリック』

progress_list 論理データ・グループ

1429 ページの『progress_list_attr 現在の進行リストの属性』

1430 ページの『progress_list_cur_seq_num 現行の進行リストのシーケンス番号』

rollforward 論理データ・グループ

1208 ページの『node_number ノード番号』

1456 ページの『rf_log_num - ロールフォワードされているログ : モニター・エレメント』

1456 ページの『rf_status ログ・フェーズ』

1456 ページの『rf_timestamp ロールフォワード・タイム・スタンプ』

1457 ページの『rf_type ロールフォワード・タイプ』

1707 ページの『ts_name - ロールフォワードされている表スペース : モニター・エレメント』

stmt 論理データ・グループ

834 ページの『agents_top 作成されたエージェントの数』

872 ページの『blocking_cursor ブロック・カーソル』

926 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

955 ページの『creator アプリケーション作成者』

958 ページの『cursor_name カーソル名』

980 ページの『degree_parallelism 並列処理の度合い』

1056 ページの『fetch_count 成功したフェッチの数』

1108 ページの『int_rows_deleted 削除された内部行数』

1108 ページの『int_rows_inserted 挿入された内部行数』

1109 ページの『int_rows_updated 更新された内部行数』

1208 ページの『num_agents ステートメントで作動しているエージェントの数』

1248 ページの『package_name - パッケージ名 : モニター・エレメント』

1249 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1377 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

1379 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

1381 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

1383 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

1386 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

1388 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

1400 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

1432 ページの『query_card_estimate 照会行数の見積もり』

1433 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1477 ページの『section_number - セクション番号 : モニター・エレメント』

1504 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

1527 ページの『stmt_elapsed_time 最新のステートメント経過時間』

1533 ページの『stmt_node_number ステートメント・ノード』

1533 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

1536 ページの『stmt_sorts ステートメント・ソート回数』

1537 ページの『stmt_start ステートメント操作開始タイム・スタンプ』

1537 ページの『stmt_stop ステートメント操作停止タイム・スタンプ』

1538 ページの『stmt_sys_cpu_time ステートメントが使用したシステム CPU 時間』

1539 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

1540 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

1542 ページの『stmt_usr_cpu_time ステートメントに使用されたユーザー CPU 時間』

1679 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

stmt_transmissions 論理データ・グループ

1007 ページの『elapsed_exec_time ステートメント実行経過時間』

1084 ページの『host_response_time ホスト応答時間』

1179 ページの『max_data_received_1024 受信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

1180 ページの『max_data_received_128 受信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数』

1180 ページの『max_data_received_16384 受信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数』

1181 ページの『max_data_received_2048 受信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数』

1181 ページの『max_data_received_256 受信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数』

1182 ページの『max_data_received_31999 受信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数 : モニター・エレメント』

1182 ページの『max_data_received_4096 受信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数』

1183 ページの『max_data_received_512 受信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数』

1183 ページの『max_data_received_64000 受信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数 : モニター・エレメント』

1184 ページの『max_data_received_8192 受信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数』

1184 ページの『max_data_received_gt64000 受信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数』

1185 ページの『max_data_sent_1024 送信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

1185 ページの『max_data_sent_128 送信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数』

1186 ページの『max_data_sent_16384 送信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数』

1186 ページの『max_data_sent_2048 送信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数』

1187 ページの『max_data_sent_256 送信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数』

1187 ページの『max_data_sent_31999 送信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数』

1188 ページの『max_data_sent_4096 送信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数』

1188 ページの『max_data_sent_512 送信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数』

1189 ページの『max_data_sent_64000 送信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数』

1189 ページの『max_data_sent_8192 送信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数』

1190 ページの『max_data_sent_gt64000 送信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数』

1190 ページの『max_network_time_100_ms ネットワーク時間が 16 から 100 ミリ秒のステートメント数』

1190 ページの『max_network_time_16_ms ネットワーク時間が 4 から 16 ミリ秒のステートメント数』

1191 ページの『max_network_time_1_ms ネットワーク時間が 1 ミリ秒以下のステートメント数』

1191 ページの『max_network_time_4_ms ネットワーク時間が 1 から 4 ミリ秒のステートメント数』

1192 ページの『max_network_time_500_ms ネットワーク時間が 100 から 500 ミリ秒のステートメント数』

1192 ページの『max_network_time_gt500_ms ネットワーク時間が 500 ミリ秒を超えるステートメント数』

1205 ページの『network_time_bottom ステートメントの最小ネットワーク時間』

1206 ページの『network_time_top ステートメントの最大ネットワーク時間』

1243 ページの『outbound_bytes_received 受信されたアウトバウンド・バイト数』

1243 ページの『outbound_bytes_received_bottom 受信された最小アウトバウンド・バイト数』

1243 ページの『outbound_bytes_received_top 受信された最大アウトバウンド・バイト数』

1244 ページの『outbound_bytes_sent 送信されたアウトバウンド・バイト数』

1244 ページの『outbound_bytes_sent_bottom 送信された最小アウトバウンド・バイト数』

1245 ページの『outbound_bytes_sent_top 送信された最大アウトバウンド・バイト数』

1508 ページの『sql_chains 試行された SQL チェーンの数』

1509 ページの『sql_stmts 試行された SQL ステートメントの数』

subsection 論理データ・グループ

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1519 ページの『ss_exec_time サブセクション実行経過時間』

1519 ページの『ss_node_number サブセクション・ノード番号』

1520 ページの『ss_number サブセクション番号 : モニター・エレメント』

1520 ページの『ss_status サブセクションの状況 : モニター・エレメント』

1521 ページの『ss_sys_cpu_time サブセクションに使用されたシステム CPU 時間』

1521 ページの『ss_usr_cpu_time サブセクションに使用されたユーザー CPU 時間』

1698 ページの『tq_cur_send_spills オーバーフローした表キュー・バッファーの現在数 : モニター・エレメント』

1698 ページの『tq_id_waiting_on ノード上の表キュー待機 : モニター・エレメント』

1698 ページの『tq_max_send_spills 表キュー・バッファー・オーバーフローの最大数』

1699 ページの『tq_node_waited_for 表キュー上のノード待機』

1699 ページの『tq_rows_read 表キューから読み取られた行数』

1700 ページの『tq_rows_written 表キューに書き込まれた行数』

1705 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

1706 ページの『tq_wait_for_any 表キュー上のノード送信待機』

table 論理データ・グループ

959 ページの『data_object_pages データ・オブジェクト・ページ数』

960 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

1098 ページの『index_object_pages 索引オブジェクト・ページ数』

1124 ページの『lob_object_pages LOB オブジェクト・ページ数』

1174 ページの『long_object_pages 長いオブジェクト・ページ数』

1246 ページの『overflow_accesses - オーバーフロー・レコードへのアクセス : モニター・エレメント』

1251 ページの『page_reorgs ページ再編成 : モニター・エレメント』

1466 ページの『rows_read - 読み取り行数 : モニター・エレメント』

1472 ページの『rows_written 書き込み行数』

1554 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

1555 ページの『table_name - 表名 : モニター・エレメント』

1557 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

1559 ページの『table_type - 表タイプ : モニター・エレメント』

table_list 論理データ・グループ

table_reorg 論理データ・グループ

tablespace 論理データ・グループ

998 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

1057 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

1058 ページの『fs_caching - ファイル・システム・キャッシング : モニター・エレメント』

1271 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

1273 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

1274 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

1277 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

1278 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

1279 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

1280 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

1281 ページの『pool_async_write_time - バッファー・プール非同期書き込み時間 : モニター・エレメント』

1284 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

1285 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

1286 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

1298 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

1302 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

1335 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

1339 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

1343 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

1373 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

tablespace_container 論理データ・グループ

926 ページの『container_accessible - コンテナーのアクセス可能性 : モニター・エレメント』

927 ページの『container_id - コンテナー ID : モニター・エレメント』

927 ページの『container_name - コンテナー名 : モニター・エレメント』

928 ページの『container_stripe_set - コンテナー・ストライプ・セット : モニター・エレメント』

928 ページの『container_total_pages - コンテナー内の合計ページ数 : モニター・エレメント』

929 ページの『container_type - コンテナー・タイプ : モニター・エレメント』

929 ページの『container_usable_pages - コンテナー内の使用可能なページ数 : モニター・エレメント』

tablespace_list 論理データ・グループ

965 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』

967 ページの『db_name データベース名 : モニター・エレメント』

968 ページの『db_path データベース・パス』

1100 ページの『input_db_alias 入力データベース別名』

1121 ページの『last_reset 最後のリセット・タイム・スタンプ』

tablespace_nodeinfo 論理データ・グループ

1561 ページの『tablespace_current_size 表スペースの現行サイズ』

1562 ページの『tablespace_free_pages 表スペース内のフリー・ページ数 : モニター・エレメント』

1564 ページの『tablespace_increase_size バイト単位のサイズの増加』

1564 ページの『tablespace_increase_size_percent パーセント単位のサイズの増加 : モニター・エレメント』

1564 ページの『tablespace_initial_size 表スペースの初期サイズ』

1565 ページの『tablespace_last_resize_failed 失敗した最後のサイズ変更』

1565 ページの『tablespace_last_resize_time 最後にサイズ変更が正常に行われた時刻』

1565 ページの『tablespace_max_size 表スペースの最大サイズ』

1566 ページの『tablespace_min_recovery_time - ロールフォワードの最小リカバリー時間 : モニター・エレメント』

1568 ページの『tablespace_num_containers 表スペース内のコンテナー数』

1568 ページの『tablespace_num_quiescers - 静止プログラム数』

1569 ページの『tablespace_num_ranges 表スペース・マップ内の範囲数』

1569 ページの『tablespace_page_top 表スペース最高水準点 : モニター・エレメント』

1570 ページの『tablespace_paths_dropped - ドロップされたパスを使用している表スペース : モニター・エレメント』

1570 ページの『tablespace_pending_free_pages 表スペース内のペンディング・フリー・ページ数 : モニター・エレメント』
1571 ページの『tablespace_prefetch_size - 表スペースのプリフェッチ・サイズ : モニター・エレメント』
1571 ページの『tablespace_rebalancer_extents_processed リバランサーで処理されたエクステントの数』
1572 ページの『tablespace_rebalancer_extents_remaining リバランサーで処理されるエクステントの合計数』
1572 ページの『tablespace_rebalancer_last_extent_moved リバランサーによって最後に移動されたエクステント』
1574 ページの『tablespace_rebalancer_priority 現行のリバランサー優先順位』
1574 ページの『tablespace_rebalancer_restart_time リバランサー再始動時刻』
1575 ページの『tablespace_rebalancer_start_time リバランサー開始時刻』
1577 ページの『tablespace_state - 表スペースの状態 : モニター・エレメント』
1579 ページの『tablespace_state_change_object_id 状態変更オブジェクト ID』
1579 ページの『tablespace_state_change_ts_id 状態変更表スペース ID』
1580 ページの『tablespace_total_pages 表スペース内の合計ページ数 : モニター・エレメント』
1581 ページの『tablespace_usable_pages 表スペース内の使用可能ページ数 : モニター・エレメント』
1582 ページの『tablespace_used_pages 表スペース内の使用されているページ数 : モニター・エレメント』

tablespace_quiescer 論理データ・グループ

1437 ページの『quiescer_agent_id 静止プログラム・エージェント ID』
1437 ページの『quiescer_auth_id 静止プログラム・ユーザー許可 ID』
1437 ページの『quiescer_obj_id 静止プログラム・オブジェクト ID』
1438 ページの『quiescer_state 静止プログラムの状態』
1438 ページの『quiescer_ts_id 静止プログラム表スペース ID』

tablespace_range 論理データ・グループ

1439 ページの『range_adjustment 範囲調整』
1439 ページの『range_container_id 範囲コンテナー』
1439 ページの『range_end_stripe 終了ストライプ』
1439 ページの『range_max_extent 範囲内の最大エクステント』
1439 ページの『range_max_page_number 範囲内の最大ページ』
1440 ページの『range_num_containers 範囲内コンテナーの数』
1440 ページの『range_number 範囲番号』
1440 ページの『range_offset 範囲オフセット』
1440 ページの『range_start_stripe 開始ストライプ』
1440 ページの『range_stripe_set_number ストライプ・セット番号』

utility_info 論理データ・グループ

1208 ページの『node_number ノード番号』
1722 ページの『utility_dbname ユーティリティーで操作されるデータベース』
1722 ページの『utility_description ユーティリティー記述』
1723 ページの『utility_id ユーティリティー ID』
1724 ページの『utility_invoker_type - ユーティリティー呼び出し側タイプ』
1726 ページの『utility_priority ユーティリティー優先度』
1727 ページの『utility_start_time ユーティリティー開始時刻』
1727 ページの『utility_state - ユーティリティー状態』
1729 ページの『utility_type ユーティリティー・タイプ』

第 11 章 モニター・エレメントのリファレンス

モニター・エレメントによって収集されるデータの説明。

システム・モニターが戻すモニター・エレメントは次のように分類できます。

- モニターされるデータベース・マネージャー、アプリケーション、またはデータベース接続の**識別**。
- システムの**構成**に最初に必要とされるデータ。
- データベース、アプリケーション、表、またはステートメントを含むさまざまなレベルのデータベース・**アクティビティー**。この情報を使用して、アクティビティーのモニター、問題判別、およびパフォーマンス分析を行うことができます。この情報を構成にも使用することができます。
- **DB2 Connect** アプリケーションに関する情報。この中には、ゲートウェイで実行中の DCS アプリケーション、実行中の SQL ステートメント、およびデータベース接続に関する情報が含まれます。
- **フェデレーテッド・データベース・システム**に関する情報。これには、 DB2 フェデレーテッド・システム内で実行中の各アプリケーションからデータ・ソースへのすべてのアクセスに関する情報、およびフェデレーテッド・サーバー・インスタンス内で実行中の特定のアプリケーションからデータ・ソースへのアクセスに関する情報が含まれます。

モニター・エレメントは次の標準形式で記述されます。

エレメント ID
エレメントの名前。データ・ストリームをそのまま構文解析すると、エレメント ID が大文字となり、 SQLM_ELM_ の接頭部が付きます。

エレメント・タイプ
モニター・エレメントが戻す情報のタイプ。例えば db2start_time モニター・エレメントはタイム・スタンプを戻します。

表関数モニター情報
モニター・エレメントが表関数によって戻される場合は、次のフィールドがある表が記載されています。

- *表関数*: モニター・エレメントを戻す表関数の名前
- *モニター・エレメントの収集レベル*: モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。

スナップショット・モニター情報
モニター・エレメントがスナップショット・モニター情報を戻す場合は、次のフィールドがある表が記載されています。

- *スナップショット・レベル* : スナップショット・モニターでキャプチャー可能な情報レベル。例えば appl_status モニター・エレメントは、アプリケーション・レベルおよびロック・レベルでの情報を戻します。
- *論理データ・グループ* : キャプチャーされたスナップショット情報が戻される論理データ・グループ。データ・ストリームをそのまま構文解析す

ると、論理データ・グループ ID が大文字となり、 SQLM_ELM_ の接頭部が付きます。例えば、appl_status モニター・エレメントは、appl_id_info グループおよび appl_lock_list グループの情報を戻します。

- モニター・スイッチ : この情報を取得するために設定が必要なシステム・モニター・スイッチ。スイッチが「基本」の場合は、モニター・エレメントのデータが常に収集されます。

イベント・モニター情報

モニター・エレメントがイベント・モニターによって収集される場合は、次のフィールドがある表が記載されています。

- イベント・タイプ : イベント・モニターで収集可能な情報のレベル。この情報を収集するには、このイベント・タイプを使用してイベント・モニターを作成する必要があります。例えば appl_status モニター・エレメントは、「接続」イベント・モニターで収集されます。
- *論理データ・グループ* : 取り込まれたイベント情報が戻される論理データ・グループ。データ・ストリームをそのまま構文解析すると、論理データ・グループ ID が大文字となり、 SQLM_ELM_ の接頭部が付きます。例えば appl_status モニター・エレメントは、 event_conn グループの情報を戻します。
- モニター・スイッチ : この情報を取得するために設定が必要なシステム・モニター・スイッチ。イベント・モニターの場合、イベント・データの収集を制約できるモニター・スイッチは TIMESTAMP スイッチのみです。このフィールドにダッシュが示されている場合は、モニター・エレメントのデータが常に収集されます。

使用法 データベース・システムをモニターする際の、モニター・エレメントによって収集された情報の使用方法に関する情報。

acc_curs_blk 受け入れられたブロック・カーソル要求

入出力ブロック要求が受け入れられた回数。

エレメント ID

acc_curs_blk

エレメント・タイプ

カウンター

表 152. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 153. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法 このエレメントと *rej_curs_blk* を組み合わせて使用すると、受け入れられたブロッキング要求、リジェクトされたブロッキング要求、またはその両方のパーセンテージを計算できます。

この情報を使用して構成パラメーターを調整する方法については、『*rej_curs_blk*』を参照してください。

act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント

エラーで終了した、任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーが異常終了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、そのアクティビティーのカウントは、異常終了時のサブクラスでの合計にのみ含められます。

表 154. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 155. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、システム上のアクティビティーが正常に完了しているかを知ることができます。アクティビティーは、取り消し、エラー、または反応的しきい値のために異常終了することがあります。

act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント

正常に完了した、任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーが完了前に REMAP ACTIVITY アクションで別のサブクラスに再マップされた場合、そのアクティビティーのカウントは、完了時のサブクラスでの合計にのみ含められます。

表 156. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 156. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得	REQUEST METRICS BASE

表 157. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、システム内のアクティビティーのスループットを判別します。

act_cpu_time_top - アクティビティーの CPU 時間の最上位 : モニター・エレメント

サービス・クラス、ワークロード、または作業クラスでの、すべてのネスト・レベルにおけるアクティビティーで使用されるプロセッサー時間の最高水準点。この値はマイクロ秒単位で報告されます。

アクティビティーが実行されるサービス・クラスまたはワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。要求メトリックが使用可能になっている場合にのみ、アクティビティーはこの最高水準点に寄与します。アクティビティー・メトリックの収集が使用可能になっていない場合には、値 0 が返されます。

サービス・クラスでは、REMAP ACTIVITY アクションを使用してサービス・サブクラス間でアクティビティーを再マップすると、新規の最高水準点に達した場合、アクティビティーが完了するサービス・サブクラスの act_cpu_time_top 最高水準点

のみが更新されます。アクティビティーがマップされるものの完了していない他のサービス・サブクラスの act_cpu_time_top 最高水準点は、影響を受けません。

表 158. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーでアクティビティーによって使用された最大プロセッサー時間を判別することができます。

act_exec_time アクティビティー実行時間 : モニター・エレメント

act_exec_time エレメントは、このメンバーで実行するために費やされた時間 (マイクロ秒単位) を格納します。

カーソルの場合、実行時間はオープン、フェッチ、およびクローズの時間を組み合わせたものです。カーソルのアイドル時間は実行時間にカウントされません。ルーチンの場合、実行時間はルーチン呼び出しの開始から終了までです。ルーチンの完了後にそのルーチンによって (結果セットを戻すために) オープンされたままになっているカーソルの存続期間は、ルーチンの実行時間にカウントされません。他のすべてのアクティビティーの場合、実行時間は開始時刻から停止時刻までの時間です。どの場合でも、実行時間には、初期化されている時間またはキューに入れられている時間は含まれません。

表 159. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを単独で使用すると、メンバーごとに DB2 によるアクティビティーの実行に費やされた経過時間を知ることができます。このエレメントは、**time_started** および **time_completed** モニター・エレメントと一緒にコーディネーター・メンバーで使用して、カーソル・アクティビティーにおけるアイドル時間を計算することもできます。以下の公式を使用できます。

```
カーソルのアイドル時間 = (time_completed - time_started) - act_exec_time
```

act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント

実行が許可されず、リジェクトされた任意のネスト・レベルのコーディネーター・アクティビティーの合計数。

表 160. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 161. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用すると、実行を阻止する予測しきい値および作業アクションが有効であるかどうか、および、それらの制限が大きすぎないかどうかを判別する助けになります。

act_remapped_in - 再マッピングするアクティビティー : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスに再マッピングするアクティビティーの数。

表 162. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このカウントを使用して、サービス・サブクラスへのアクティビティーの再マップが期待どおりに行われているかを判別します。

act_remapped_out - 再マッピングの際に除外されるアクティビティー : モニター・エレメント

最後のリセット以降、再マッピングの際にこのサービス・サブクラスから除外されるアクティビティーの数。

表 163. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このカウントを使用して、サービス・サブクラスからのアクティビティーの再マップが期待どおりに行われているかを判別します。

act_rows_read_top - アクティビティーの読み取り行数の最上位 : モニター・エレメント

サービス・クラス、ワークロード、または作業クラスでの、すべてのネスト・レベルにおけるアクティビティーによって読み取られる行数の最高水準点。

アクティビティーが実行されるサービス・クラスまたはワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。要求メトリックが使用可能になっている場合にのみ、アクティビティーはこの最高水準点に寄与します。アクティビティー・メトリックの収集が使用可能になっていない場合には、値 0 が返されます。

サービス・クラスでは、REMAP ACTIVITY アクションを使用してサービス・サブクラス間でアクティビティーを再マップすると、新規の最高水準点に達した場合、アクティビティーが完了するサービス・サブクラスの act_rows_read_top 最高水準点のみが更新されます。アクティビティーがマップされるものの完了していないサービス・サブクラスの act_rows_read_top 最高水準点は、影響を受けません。

表 164. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーでアクティビティーによって読み取られる最大行数を判別することができます。

act_rqsts_total - アクティビティー要求の合計数 : モニター・エレメント

アクティビティーの一部として完了される個別のコーディネーター要求およびサブエージェント要求の数。例えば、カーソル・アクティビティーにおけるフェッチなどです。

表 165. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 165. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 166. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

act_throughput - アクティビティー・スループット : モニター・エレメント

任意のネスト・レベルでコーディネーター・アクティビティーが完了するレート。1 秒単位のコーディネーター・アクティビティー数で測定されます。

表 167. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 168. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	常に収集される
統計	event_wlstats (メトリック文書に報告されます)	常に収集される

使用法

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降のアクティビティー・スループットを表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降のアクティビティー・スループットを表します。

act_total アクティビティーの合計 : モニター・エレメント

最後にリセットしてから指定した作業クラスに対応する作業アクションが適用された、任意のネスト・レベルのアクティビティーの合計数。

表 169. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE

表 170. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	-

使用法

作業クラスに関連付けられた 1 つ以上の作業アクションがアクティビティーに適用されるたびに、この作業クラスのカウンターが更新されます。**act_total** モニター・エレメントを使用すると、このカウンターが公開されます。このカウンターを使用して、作業アクション・セットの有効性 (例えば、アクションが適用されているアクティビティーの数) を判断できます。また、システム上のアクティビティーのさまざまなタイプを理解するためにも使用できます。

activate_timestamp タイム・スタンプの活動化 : モニター・エレメント

イベント・モニターがアクティブにされた時刻。

表 171. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

表 171. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-
アクティビティー	event_activityvals	-
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、上記のイベント・タイプで戻された情報を関連付けることができます。

active_hash_joins - アクティブ・ハッシュ結合

現在実行中でメモリーを消費しているハッシュ結合の合計数

表 172. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-

active_olap_funcs アクティブ OLAP 関数 : モニター・エレメント

現在実行中でソート・ヒープ・メモリーを消費している OLAP 関数の合計数

表 173. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-

スナップショット・モニターの場合、このカウンターはリセットできます。

active_sorts アクティブ・ソート

現在ソート・ヒープが割り振られているデータベース内のソート数。

表 174. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 この値と *sort_heap_allocated* を組み合わせて使用すると、各ソートで使用される平均ソート・ヒープ・スペースを判別できます。使用されている平均ソート・ヒープと比較して、 *sortheap* 構成パラメーターが非常に大きい場合は、このパフォーマンス値を低くできます。

この値には、関連操作で作成された一時表のソートのヒープが含まれます。

activity_collected 収集されたアクティビティー : モニター・エレメント

このエレメントは、しきい値の違反が発生した場合にアクティビティー・イベント・モニター・レコードが収集されるかどうかを示します。

表 175. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値を違反したアクティビティーのアクティビティー・イベントがアクティビティー・イベント・モニターに書き込まれるかどうかを判別できます。

アクティビティーが完了またはアボートし、その時点でアクティビティー・イベント・モニターがアクティブである場合、このモニター・エレメントの値が「Y」である場合には、このしきい値に違反したアクティビティーは収集されます。このモニター・エレメントの値が「N」である場合、それは収集されません。

activity_id アクティビティー ID : モニター・エレメント

特定の作業単位内のアプリケーションのアクティビティーを一意的に識別するカウンター。

表 176. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 177. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
アクティビティー	event_activity	常に収集される
アクティビティー	event_activitystmt	常に収集される
アクティビティー	event_activityvals	常に収集される
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
しきい値違反	event_thresholdviolations	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

アクティビティーをその作業単位外から一意的に識別するには、**activity_id** および **uow_id** と、**appl_id** または **agent_id** のいずれかのモニター・エレメントを組み合わせて使用してください。

activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント

このエレメントの値は、同じアクティビティーに関してアクティビティー・レコードが書き込まれるたびに増分されます。

例えば、アクティビティー・レコードが、WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを呼び出した結果として 1 回目に書き込まれ、アクティビティーが終了した時に 2 回目に書き込まれた場合、エレメントの値は、最初のレコードについては 0、2 番目のレコードについては 1 となります。

表 178. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
アクティビティー	event_activitystmt	-
アクティビティー	event_activityvals	-
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE

使用法

このエレメントを **activity_id**、**uow_id**、および **appl_id** モニター・エレメントと一緒に使用すると、同一のアクティビティーに関する情報がアクティビティー・イベント・モニターに複数回書き込まれた場合にアクティビティー・レコードを一意的に識別できます。

例えば、以下の場合には、アクティビティーに関する情報がアクティビティー・イベント・モニターに 2 回送信されます。

- WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを使用して、実行中のアクティビティーに関する情報をキャプチャーした場合
- アクティビティーが関連付けられているサービス・クラス上で COLLECT ACTIVITY DATA 文節を指定したために、そのアクティビティーの完了時にそのアクティビティーに関する情報を収集した場合。

activity_state - アクティビティーの状態 : モニター・エレメント

アクティビティーの現在の状態。

表 179. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、アクティビティーの現在の状況を判別します (例えば、アクティビティーがキューに入れられている、あるいはクライアントからの入力を待機している、など)。可能な値は以下のとおりです。

- CANCEL_PENDING
- EXECUTING
- IDLE
- INITIALIZING
- QP_CANCEL_PENDING
- QP_QUEUED
- QUEUED
- TERMINATING
- UNKNOWN

activity_type アクティビティー・タイプ : モニター・エレメント

アクティビティーのタイプ。

表 180. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 181. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

可能な値は以下のとおりです。

- LOAD
- READ_DML
- WRITE_DML
- DDL
- CALL
- OTHER

SQL を実行しない SET ステートメント (SET 特殊レジスターや SET EVENT MONITOR STATE など) および LOCK TABLE ステートメント場合、値 OTHER が戻ります。

activitytotaltime_threshold_id - アクティビティー合計時間しきい値 ID : モニター・エレメント

アクティビティーに適用されていた ACTIVITYTOTALTIME しきい値の ID。

表 182. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 ACTIVITYTOTALTIME しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

activitytotaltime_threshold_value - アクティビティー合計時間しきい値 : モニター・エレメント

アクティビティー・エントリー時からの ACTIVITYTOTALTIME の合計。アクティビティーがこのタイム・スタンプに達してもまだ実行している場合、しきい値に違反したことになります。

表 183. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、ACTIVITYTOTALTIME しきい値がアクティビティーに適用されている場合、その値を判別します。

activitytotaltime_threshold_violated - アクティビティー合計時間しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが ACTIVITYTOTALTIME しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 184. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた ACTIVITYTOTALTIME しきい値にアクティビティーが違反したかどうかを判別します。

adapter_name - アダプター名のモニター・エレメント

このホスト上のネットワーク・アダプターの名前。

表 185. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE

address - 接続の開始元となった IP アドレス

アクティビティー接続の開始元となった IP アドレス。

表 186. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 187. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

これを使用して、アクティビティー接続の開始元となった IP アドレスを識別できます。セキュア・ドメイン名は、IP アドレスに変換されて表示されます。

agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント

システム全体での、アプリケーションのユニーク ID。単一メンバーのデータベース構成の場合、この ID は 16 ビット・カウンターで構成されます。複数メンバーの構成の場合には、この ID はコーディネーター・メンバー番号と 16 ビット・カウンターが連結されて構成されます。さらに、アプリケーションが 2 次接続を行う可能性のあるメンバーには、すべて同一の ID が使用されます。

表 188. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 189. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本
トランザクション	event_xact	-

表 190. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
ロッキング	-	常に収集される

表 190. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
作業単位	-	常に収集される
接続	event_connheader	常に収集される
ステートメント	event_stmt	常に収集される
ステートメント	event_subsection	常に収集される
デッドロック[1]	event_dlconn	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
アクティビティー	event_activity	常に収集される
変更履歴	changesummary	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

アプリケーション・ハンドルは、エージェント ID とも呼ばれ、これを使用するとアクティブ・アプリケーションを一意的に識別できます。

注: **agent_id** モニター・エレメントは、使用する DB2 のバージョンによって動作が異なります。バージョン SQLM_DBMON_VERSION1 または SQLM_DBMON_VERSION2 の DB2 から DB2 (バージョン 5 またはそれ以上) のデータベースへスナップショットをとる時、戻される **agent_id** はアプリケーション ID として使用できず、アプリケーションを提供するエージェントの **agent_pid** として有効です。このような場合、**agent_id** は以前のリリースとの互換性のために引き続き戻されますが、内部的には、DB2 データベース・サーバーはその値を **agent_id** として認識しません。

この値は、エージェント ID を必要とする GET SNAPSHOT コマンドへの入力として、またはアプリケーション・ハンドルを必要とするモニター表関数への入力として使用できます。

またイベント・トレースを読み取るとき、イベント・レコードを指定のアプリケーションと一致させるために使用できます。

FORCE APPLICATION コマンドまたは API への入力として使用することができます。マルチノード・システムでは、アプリケーションが接続されているノードから、このコマンドを出すことができます。効力範囲はグローバルです。

agent_id_holding_lock ロックを保持しているエージェント ID

このアプリケーションが待機しているロックを保持しているエージェントのアプリケーション・ハンドル。この情報を取得するには、ロック・モニター・グループをオンにする必要があります。

表 191. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock_wait	ロック

使用法 このエレメントは、リソースの競合状態にあるアプリケーションの判別に利用できます。

このエレメントが 0 (ゼロ) で、アプリケーションがロックを待機中の場合は、ロックが未確定トランザクションによって保持されていることを示します。 appl_id_holding_lk または コマンド行プロセッサー LIST INDOUBT TRANSACTIONS コマンドのいずれかを使用して (未確定となったときにトランザクションを処理していた CICS® エージェントのアプリケーション ID を表示します)、未確定トランザクションを識別してから、コミットまたはロールバックを行います。

このアプリケーションが待機している 1 つのオブジェクトに対して、複数のアプリケーションが共有ロックを保有していることがあるので注意してください。アプリケーションが保留しているロックのタイプについては、『lock_mode』を参照してください。アプリケーションのスナップショットをとる場合は、オブジェクトのロックを保留しているエージェント ID のうち戻されるのは 1 つだけです。ロックのスナップショットを取れば、オブジェクトのロックを保留しているすべてのエージェント ID を識別できます。

agent_pid エンジン・ディスパッチ可能単位 (EDU) ID : モニター・エレメント

エージェントのエンジン・ディスパッチ可能単位 (EDU) のユニーク ID。Linux オペレーティング・システムを除いて、EDU ID はスレッド ID にマップされます。Linux オペレーティング・システムでは、EDU ID は DB2 生成のユニーク ID です。

表 192. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	agent	ステートメント

表 193. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

使用法

このエレメントを使用すると、データベース・システム・モニター情報をシステム・トレースなどの他の診断情報のソースにリンクできます。また、このエレメントを使用すると、データベース・アプリケーションの処理を行うエージェントがシステム・リソースをどのように使用しているのかをモニターできます。

agent_status DCS アプリケーション・エージェント

接続コンセントレーター環境では、この値は、関連エージェントを現在持っているアプリケーションを示します。

エレメント ID

agent_status

エレメント・タイプ

情報

表 194. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 次の値があります。

- SQLM_AGENT_ASSOCIATED

  このアプリケーションに代わって作業中のエージェントがアプリケーションに関連付けられています。

- SQLM_AGENT_NOT_ASSOCIATED

  このアプリケーションに代わって作業していたエージェントは、もはやアプリケーションには関連付けられていません。現在はほかのアプリケーションで使用されています。この後、関連エージェントがないままこのアプリケーションの作業が行われると、エージェントが再び関連付けられます。

agent_sys_cpu_time エージェントが使用したシステム CPU 時間

データベース・マネージャーのエージェント・プロセスで使用されたシステム CPU 時間の合計 (秒単位およびマイクロ秒単位)。

エレメント ID

agent_sys_cpu_time

エレメント・タイプ

time

表 195. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	タイム・スタンプ

アプリケーション・レベルでのスナップショット・モニターの場合、このカウンターはリセットできます。その他のレベルではこのカウンターはリセットできません。

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

このエレメントには、SQL および非 SQL のステートメントの両者の CPU 時間、およびその他の unfenced ユーザー定義関数 (UDF) の CPU 時間が含まれます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

agent_tid - エージェント・スレッド ID モニター・エレメント

エージェントまたはシステム・エンティティーのスレッド ID。この ID が使用できない場合、列の値は NULL です。

表 196. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE

表 197. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

agent_usr_cpu_time エージェントが使用したユーザー CPU 時間

データベース・マネージャーのエージェント・プロセスで使用された CPU 時間の合計 (秒およびマイクロ秒単位)。

エレメント ID

agent_usr_cpu_time

エレメント・タイプ

time

表 198. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントとその他の CPU 時間関連のエレメントを組み合わせて使用すると、 CPU を大量に使用しているアプリケーションや照会を識別できます。

このカウンターには、SQL および非 SQL のステートメントに要した時間のほか、アプリケーションが実行した unfenced ユーザー定義関数 (UDF) およびストアード・プロシージャーも含まれます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

注: 使用しているオペレーティング・システムでこの情報を得られない場合は、このエレメントはゼロ (0) を戻します。

agent_wait_time - エージェント待機時間 : モニター・エレメント

キューに入れられたアプリケーションが、コンセントレーター構成のもとでエージェントの待機に費やした時間。値はミリ秒単位で示されます。

表 199. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 199. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 200. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	常に収集される

使用法

agent_wait_time モニター・エレメントを使用すると、システムがコンセントレーター環境でどの程度効率的に稼働しているかを評価するのに役立ちます。エージェント待機時間が **total_request_time** モニター・エレメントの値に対して相対的に大きい場合、要求がエージェントの待機のためにキューに入れられて長い時間を費やしていることを示します。これは、以下のイベントの 1 つ以上の状況を示している可能性があります。

- **max_coordagents** 構成パラメーターの構成が、ワークロードに対して小さすぎる。アプリケーション要求に適切なタイミングで対応するのに十分な数のコーディネーター・エージェントを使用できるようにするために、**max_coordagents**構成パラメーターの値を大きくするか、 **max_connections** 構成パラメーターに対する **max_coordagents** 構成パラメーターの比率を大きくする (両方のパラメーターとも AUTOMATIC に設定して実行している場合) 必要がある可能性があります。
- ワークロードのコミットの頻度が十分でない。コンセントレーターが効率的に処理を行うには、アプリケーションはコミットを比較的頻繁に発行して、他のアプリケーションの要求に対応するためにエージェントを解放できるようにする必要があります。アプリケーションで頻繁にコミットを行わない場合、コーディネー

ター・エージェントの数をそれに応じて多く構成し、エージェントが使用可能になるのを待機する時間を短くする必要がある可能性があります。

agent_waits_total - エージェント待機の合計 : モニター・エレメント

コンセントレーター構成において、アプリケーションが、エージェントが割り当てられるのを待機しなければならなかった回数。

表 201. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 202. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを **agent_wait_time** モニター・エレメントと組み合わせて使用すると、コンセントレーター環境においてアプリケーション要求がエージェントを待機するのにかかる平均時間を判別できます。

agents_created_empty_pool - エージェント・プールが空のために作成されたエージェント

エージェント・プールが空だったために作成されたエージェントの数。これには、**db2start** のときに開始されるエージェントの数 (*num_initagents)* が含まれます。

表 203. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 agents_from_pool と組み合わせて使用すると、次の比率を計算できます。

エージェント・プールが空のために作成されたエージェント/
プールから割り当てられたエージェント

このエレメントの使用法については、『agents_from_pool』を参照してください。

agents_from_pool - プールから割り当てられたエージェント

エージェント・プールから割り当てられたエージェントの数。

表 204. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントと **agents_created_empty_pool** モニター・エレメントを組み合わせて使用すると、プールが空になってエージェントの作成が必要となる頻度を判別できます。

以下の比率を使用します。

エージェント・プールが空のために作成されたエージェント/
プールから割り当てられたエージェント

この比率は **num_poolagents** 構成パラメーターの適切な値を設定するために利用できます。

ほとんどのユーザーの場合、デフォルト値 100 と AUTOMATIC で最適なパフォーマンスが確保されます。

この比率は、ワークロードに応じて多少変動する可能性があります。システム上のアクティビティーが少ない場合は、追加のエージェントの作成や終了が生じる可能

性があります。システム上のアクティビティーが多い場合は、エージェントの再使用が増えます。比率が低い場合は、再使用されるエージェントが多いので、システム上のアクティビティーが多いと予期されることを意味します。比率が高い場合は、再使用されるエージェントより作成されるエージェントの方が多いことを示します。問題がある場合は、**num_poolagents** 構成パラメーターの値を大きくして、比率を低くしてください。しかし、この場合はシステム上で追加のリソースが消費されます。

agents_registered 登録済みエージェント

モニター中のデータベース・マネージャー・インスタンスに登録されているエージェント (コーディネーター・エージェントとサブエージェント) の数。

表 205. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントは、**max_coordagents** および **max_connections** 構成パラメーターの設定や、照会内並列処理の設定を評価するときに利用してください。

agents_registered_top - エージェント最大登録数

データベース・マネージャーが開始されてからこれまでに同時に登録されていたエージェント (コーディネーター・エージェントとサブエージェント) の最大数。

表 206. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントは、**max_coordagents** および **max_connections** 構成パラメーターの設定や、照会内並列処理の設定を評価するときに利用できます。

スナップショットの実行時に登録されたエージェントの数は、agents_registered モニター・エレメントにより記録されます。

agents_stolen スチールされたエージェント

データベース・マネージャーのスナップショットのレベルでは、このモニター・エレメントは、別のアプリケーション上で作業するよう再割り当てされているアプリケーションに関連したアイドル・エージェントの数を表します。

アプリケーションのスナップショットのレベルでは、このモニター・エレメントは、このアプリケーション上で作業するよう再割り当てされている別のアプリケーションに関連したアイドル・エージェントの数を表します。

表 207. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

デフォルトでは **num_poolagents** 構成パラメーターは AUTOMATIC に設定されます。この場合、DB2 により自動的にアイドル・エージェントのプールが管理され、その中には別のアプリケーションに関連したアイドル・エージェントに作業を割り当てることが含まれます。

agents_top 作成されたエージェントの数

アプリケーション・レベルでは、ステートメントの実行時に使用されたエージェントの最大数です。データベース・レベルでは、これはすべてのアプリケーション用のエージェントの最大数です。

エレメント ID
agents_top

エレメント・タイプ
水準点

表 208. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント
アプリケーション	stmt	ステートメント

使用法 照会内並列処理の実現の度合いを示します。

agents_waiting_on_token - トークン待ちエージェント

データベース・マネージャー内でトランザクションを実行するためにトークンを待機中のエージェントの数。

注: agents_waiting_on_token モニター・エレメントは、DB2 バージョン 9.5 以降では非推奨になっています。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 209. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントを使用すると、**maxcagents** 構成パラメーターの設定値を評価するのに役立ちます。

各アプリケーションには、データベース・マネージャー内でデータベース要求を処理するための専用コーディネーター・エージェントが 1 つずつ組み込まれます。各エージェントは、トークンを取得してから、トランザクションを実行できます。データベース・マネージャーのトランザクションを実行できるエージェントの最大数は、 **maxcagents** 構成パラメーターの値によって制限されます。

agents_waiting_top - エージェント最大待機数 : モニター・エレメント

データベース・マネージャーが開始されてから、同時にトークンを待機していたエージェントの最大数。

注: agents_waiting_top モニター・エレメントは、DB2 バージョン 9.5 以降では非推奨になっています。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 210. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントは、**maxcagents** 構成パラメーターの設定値を評価するときに利用できます。

スナップショットの実行時にトークンを待機していたエージェントの数は、**agents_waiting_on_token** モニター・エレメントにより記録されます。

maxcagents パラメーターをデフォルト値 (-1) に設定すると、トークンを待つエージェントがなくなるため、このモニター・エレメントの値はゼロになります。

agg_temp_tablespace_top 集約 TEMPORARY 表スペースの最上位 : モニター・エレメント

agg_temp_tablespace_top モニター・エレメントは、サービス・クラスでの、すべてのネスト・レベルにおける DML アクティビティーの TEMPORARY 表スペースの集約された使用量の最高水準点 (KB 単位) を格納します。

集約は、サービス・サブクラスのすべてのアクティビティーに渡る TEMPORARY 表スペースの使用量を合計して計算されます。この最高水準点は、最後のリセット以降のこの集約の最大値を表しています。 サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合に、このモニター・エレメントは -1 を返します。このレコードのあるサブクラスと同じスーパー

クラスに、AGGSQLTEMPSPACE しきい値が定義され使用可能になっているサービス・サブクラスが少なくとも 1 つ必要です。そうでない場合、0 の値が返されます。

表 211. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される

使用法

このエレメントを使用して、収集された時間間隔にサービス・サブクラスのメンバーで到達した DML アクティビティーの SYSTEM TEMPORARY 表スペース使用量の最大集約値を判別することができます。

aggsqltempspace_threshold_id - 集約 SQL 一時スペースしきい値 ID : モニター・エレメント

アクティビティーに適用されていた AGGSQLTEMPSPACE しきい値の数値 ID。

表 212. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 AGGSQLTEMPSPACE しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

aggsqltempspace_threshold_value - AggSQL 一時スペースしきい値 : モニター・エレメント

アクティビティーに適用されていた AGGSQLTEMPSPACE しきい値の上限。

表 213. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、AGGSQLTEMPSPACE しきい値がアクティビティーに適用されている場合、その値を判別します。

aggsqltempspace_threshold_violated - AggSQL 一時スペースしきい値の違反 : モニター・エレメント

オプションのモニター・エレメント。これが「Yes」に設定された場合、アクティビティーが自身に適用されていた AGGSQLTEMPSPACE しきい値に違反したことを示します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 214. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた AGGSQLTEMPSPACE しきい値にアクティビティーが違反したかどうかを判別します。

app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの合計数のモニター・エレメント

エラーで終了した、ネストなしの外部コーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーが異常終了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、そのアクティビティーのカウントは、異常終了時のサブクラスでの合計にのみ含められます。

表 215. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 215. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 216. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

app_act_completed_total - 正常実行された外部コーディネーター・アクティビティーの合計数のモニター・エレメント

正常に完了した、ネストなしの外部コーディネーター・アクティビティーの合計数。

サービス・クラスでは、アクティビティーが完了前に REMAP ACTIVITY アクションで別のサブクラスに再マップされた場合、そのアクティビティーのカウントは、完了時のサブクラスでの合計にのみ含められます。

表 217. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 217. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 218. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

app_act_rejected_total - 拒否された外部コーディネーター・アクティビティーの合計数のモニター・エレメント

実行が許可されずに拒否された、どのネスト・レベルでもネストされていない外部コーディネーター・アクティビティーの合計数。

表 219. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 219. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 220. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

appl_action - アプリケーション・アクションのモニター・エレメント

クライアント・アプリケーションが実行中のアクションまたは要求。

表 221. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント

コーディネーターによって実行された外部 (アプリケーション) 要求の合計数。サービス・サブクラスでは、このモニター・エレメントはアプリケーション要求が完了したサブクラスについてのみ更新されます。

表 222. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 223. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 223. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このモニター・エレメントを使用して、アプリケーションからシステムにサブミットされている要求の数を判別します。

appl_con_time 接続要求開始タイム・スタンプ

アプリケーションが接続要求を開始した日時。

エレメント ID

appl_con_time

エレメント・タイプ

timestamp

表 224. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	タイム・スタンプ

使用法 このエレメントを使用すると、アプリケーションがデータベースへの接続要求を開始した日時を判別できます。

appl_id - アプリケーション ID : モニター・エレメント

アプリケーションがデータベース・マネージャーのデータベースに接続したとき、または DB2 Connect が DRDA データベースへの接続要求を受け取ったときにこの ID が生成されます。

表 225. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE

表 225. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
PDLOGMSGS_LAST24HOURS 管理ビューおよび PD_GET_LOG_MSGS 表関数 - 問題診断メッセージの取得	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 226. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
DCS アプリケーション	dcs_appl_info	基本
ロック	appl_lock_list	基本

表 227. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング [1]	lock_participants	常に収集される
作業単位 [1]	uow, uow_executable_list、uow_metrics	常に収集される
接続	event_conn	常に収集される
接続	event_connheader	常に収集される
ステートメント	event_stmt	常に収集される
トランザクション [2]	event_xact	常に収集される
デッドロック [3]	event_dlconn	常に収集される
詳細付きデッドロック [3]	event_detailed_dlconn	常に収集される
アクティビティー	event_activitystmt	常に収集される
アクティビティー	event_activity	常に収集される
アクティビティー	event_activityvals	常に収集される
アクティビティー	event_activitymetrics	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

1. このイベント・モニターでは、このモニター・エレメントは APPLICATION_ID 列に返されます。
2. このイベント・モニターは推奨されなくなりました。このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 トランザクション・イベントをモニターするには、 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用してください。
3. このイベント・モニターは推奨されなくなりました。このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 ロック・タイムア

ウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

この ID はクライアントとサーバーの両者により認識されるため、この ID を使用すると、アプリケーションのクライアント部分とサーバー部分を相関させることができます。 DB2 Connect アプリケーションでアプリケーションのクライアント部分とサーバー部分を相関させるには **outbound_appl_id** モニター・エレメントも使用する必要があります。

この ID は、ネットワーク内でユニークな ID です。アプリケーション ID にはさまざまな形式があり、データベース・マネージャー、DB2 Connect、またはその両方を実行中のクライアントとサーバー・マシン間の通信プロトコルにより形式が異なります。どの形式の場合もピリオドで区切られた 3 つの部分で構成されます。

1. TCP/IP

   形式 `IPAddr.Port.Timestamp`

   IPv4

   例 `9.26.120.63.43538.090924175700`

   詳細 IPv4 では、TCP/IP の生成するアプリケーション ID は、 3 つのセクションで構成されます。最初のセクションは IP アドレスです。`a.b.c.d` 形式の 4 つの 10 進数として表されます。2 番目のセクションはポート番号であり、5 桁の 10 進文字として表されます。 3 番目のセクションはです。概算のタイム・スタンプであり、 12 桁の 10 進数で表記されます。

   IPv6

   例 `2002:91a:519:13:20d:60ff:feef:cc64.5309.090924175700`

   詳細 IPv6 では、TCP/IP の生成するアプリケーション ID は、3 つのセクションで構成されます。最初のセクションには、a:b:c:d:e:f:g:h 形式 (a から h はそれぞれ最大 4 桁の 16 進数字) の IPv6 アドレスが入ります。 2 番目のセクションはポート番号です。3 番目のセクションはです。概算のタイム・スタンプ ID であり、このアプリケーションのインスタンスの ID です。

2. ローカル・アプリケーション

   形式 `*LOCAL.DB2 instance.Application instance`

   例

   ```
   *LOCAL.DB2INST1.930131235945
   ```

   詳細 ローカル・アプリケーション用に生成されるアプリケーション ID は、*LOCAL のストリング、DB2 インスタンスの名前、およびこのアプリケーションのインスタンスのユニーク ID の連結で構成されます。

複数データベース・パーティション・インスタンスの場合は、LOCAL は Nx に置き換えられます。 x は、クライアントからデータベースへの接続元のパーティション番号です。例えば、*N2.DB2INST1.0B5A12222841 となります。

client_protocol モニター・エレメントを使用すると、接続に使用している通信プロトコルを判別できるので、**appl_id** モニター・エレメントの形式も判別できます。

appl_id_holding_lk ロックを保持しているアプリケーション ID

このアプリケーションが取得のために待機しているオブジェクトのロックを保留しているアプリケーションのアプリケーション ID。

エレメント ID

appl_id_holding_lk

エレメント・タイプ

情報

表 228. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock_wait	ロック

表 229. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	常に収集される
詳細付きデッドロック	event_detailed_dlconn	常に収集される

使用法 このエレメントは、リソースの競合状態にあるアプリケーションの判別に利用できます。具体的には、ロックを保留しているアプリケーション・ハンドル (エージェント ID) と表 ID を識別するときに利用します。 LIST APPLICATIONS コマンドを使用してアプリケーション ID をエージェントに関連付ける情報を取得することもできます。ただし、このタイプの情報はスナップショットをとる際に収集する方が賢明です。 LIST APPLICATIONS コマンドを実行する前にアプリケーションが終了してしまうと情報を得られなくなることがあるからです。

このアプリケーションがロックの取得を待機している 1 つのオブジェクトに対して、複数のアプリケーションが共有ロックを保留していることがあるので注意してください。アプリケーションが保留しているロックのタイプについては、『lock_mode』を参照してください。アプリケーションのスナップショットをとる場合は、オブジェクトに対してロックを保留しているアプリケーション ID のうち戻されるのは 1 つだけです。ロックのスナップショットをとる場合は、オブジェクトに対してロックを保留しているすべてのアプリケーション ID が戻されます。

appl_id_oldest_xact 最も古いトランザクションを持つアプリケーション

最も古いトランザクションを持つアプリケーションのアプリケーション ID (アプリケーション・スナップショットからの *agent_id* 値に対応)。

エレメント ID

appl_id_oldest_xact

エレメント・タイプ

情報

表 230. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを使用すると、最も古いアクティブ・トランザクションを持つアプリケーションを判別できます。このアプリケーションにログ・スペースの解放を強制することができます。ログ・スペースを大量に消費している場合は、アプリケーションを調べて、より頻繁にコミットするよう変更できるかどうか判別してください。

ロギングを保留しているトランザクションがない場合や、最も古いトランザクションがアプリケーション ID を持たない場合 (例えば、未確定トランザクションまたは非アクティブ・トランザクション) もあります。これらの場合には、このアプリケーションの ID はデータ・ストリームに返されません。

appl_idle_time アプリケーション・アイドル時間

アプリケーションがサーバーに対して何らかの要求を出してから経過した秒数。これには、トランザクションを終了せずに、例えばコミットまたはロールバックを発行していないアプリケーションが含まれます。

エレメント ID

appl_idle_time

エレメント・タイプ

情報

表 231. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント
DCS アプリケーション	dcs_appl	ステートメント

使用法 この情報を使用すると、ユーザーが指定秒数の間アイドル状態になったときに、ユーザーに強制するようなアプリケーションをインプリメントできます。

appl_name アプリケーション名 : モニター・エレメント

クライアントで実行中のアプリケーションの名前。データベースまたは DB2 Connect サーバーが識別できる名前です。

表 232. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMINTEMPCOLUMNS 管理ビューおよび ADMIN_GET_TEMP_COLUMNS 表関数 - 一時表の列情報を取得する	ACTIVITY METRICS BASE
ADMINTEMPTABLES 管理ビューおよび ADMIN_GET_TEMP_TABLES 表関数 - 一時表の情報を取得する	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 233. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本

表 234. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
接続	event_connheader	常に収集される
アクティビティー	event_activity	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

使用法

このエレメントと **appl_id** を使用すると、データ項目をアプリケーションに関連付けることができます。

クライアント/サーバー環境において、この名前はデータベース接続を確立する際にクライアントからサーバーに送られます。アプリケーション名に含まれている英語

以外の文字はすべて除去されます。 CLI アプリケーションは、 SQLSetConnectAttr への呼び出しを使用して SQL_ATTR_INFO_PROGRAMNAME 属性を設定できます。サーバーへの接続が確立される前に SQL_ATTR_INFO_PROGRAMNAME が設定されると、指定された値は実際のクライアント・アプリケーション名をオーバーライドし、 **appl_name** モニター・エレメント中に表示されます。

クライアント・アプリケーションのコード・ページと実行中のデータベース・システム・モニターが使用しているコード・ページが異なる場合は、 **appl_name** を変換するときに **codepage_id** を利用できます。

appl_priority アプリケーション・エージェント優先順位

このアプリケーションのために作業するエージェントの優先順位。

エレメント ID

appl_priority

エレメント・タイプ

情報

表 235. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 236. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、アプリケーションが所定の優先順位で実行されているかどうかを確認できます。アプリケーションの優先順位は、管理者が設定できます。優先順位は、ガバナー・ユーティリティー (**db2gov**) で変更できます。

DB2 はガバナーを使用して、データベースに使用するアプリケーションの動作のモニターおよび変更を行います。この情報は、アプリケーションのスケジュール処理とシステム・リソースのバランス処理に使用されます。

ガバナー・デーモンはスナップショットをとることにより、アプリケーションの統計データを収集します。さらに、そのデータベース上で実行されるアプリケーションを管理する規則に照らして、この統計内容をチェックします。ガバナーが規則違反を発見すると、適切なアクションを実行します。このような規則やアクションの内容は、ガバナー構成ファイル内にユーザーが指定します。

規則に関連したアクションによりアプリケーションの優先順位が変更される場合、違反が検出されたパーティション内のエージェントの優先順位がガバナーによって変更されます。

appl_priority_type アプリケーション優先順位タイプ

アプリケーションのために作業しているエージェントの、オペレーティング・システムの優先順位タイプ。

エレメント ID

appl_priority_type

エレメント・タイプ

情報

表 237. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 238. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法 使用状況に基づいて、動的優先順位がオペレーティング・システムによって再計算されます。静的優先順位は変更されません。

appl_section_inserts セクション挿入数 : モニター・エレメント

共有 SQL 作業スペースからのアプリケーションによる SQL セクション挿入数。

表 239. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

表 240. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法

実行可能セクションの作業用コピーは、共有 SQL 作業スペースに保管されます。このカウンターは、コピーが使用できなかったために挿入が必要だった場合のカウントです。

appl_section_lookups - セクション検索

共有 SQL 作業スペースからのアプリケーションによる SQL セクション検索数。

表 241. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 242. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法

個々のエージェントには、実行可能セクションの作業用コピーが保持される共有 SQL 作業スペースへのアクセス権があります。このカウンターは、アプリケーションのエージェントにより SQL 作業域がアクセスされた回数を示します。

appl_status アプリケーション状況 : モニター・エレメント

アプリケーションの現在の状況。

表 243. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本

表 244. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
接続	event_conn	常に収集される

使用法

ロック・イベント・モニターのロック待機イベント、ロック・タイムアウト・イベント、またはデッドロック・イベントでは、ロック要求側は、現在のロック待機の状況ではなく、ロック待機状況になる前に有効であった appl_status 値をレポートします。

このエレメントは、潜在的なアプリケーション問題の診断に役立ちます。このフィールドの値は次に示す表でリストされています。

API 定数	説明
SQLM_AUTONOMOUS_WAIT	**自律型待機:** アプリケーションは自律型ルーチンが完了するのを待機しています。
SQLM_BACKUP	**データベースのバックアップ中:** アプリケーションはデータベースのバックアップを実行しています。
SQLM_COMMIT_ACT	**コミット・アクティブ:** 作業単位が、そのデータベース変更をコミットしています。
SQLM_COMP	**コンパイル中:** データベース・マネージャーはアプリケーションに代わって SQL ステートメントのコンパイル中またはプランのプリコンパイル中です。
SQLM_CONNECTED	**データベース接続完了:** アプリケーションがデータベース接続を開始し、要求は完了しています。
SQLM_CONNECTPEND	**データベース接続ペンディング:** アプリケーションはデータベース接続を開始していますが、要求は完了していません。
SQLM_CREATE_DB	**データベース作成中:** エージェントは、データベース作成の要求を開始しましたが、要求はまだ完了していません。
SQLM_DECOUPLED	**エージェントから分離済み:** アプリケーションに現在関連付けられているエージェントはありません。これは正常な状態です。接続コンセントレーターが使用可能なときは、専用コーディネーター・エージェントがないので、アプリケーションをコーディネーター・パーティションで分離することができます。非コンセントレーター環境では、専用コーディネーター・エージェントが常に存在するので、アプリケーションをコーディネーター・パーティションで分離することができません。
SQLM_DISCONNECTPEND	**データベース切断ペンディング:** アプリケーションはデータベースの切断を開始していますが、コマンドの実行は完了していません。アプリケーションがデータベース切断コマンドを明示的に実行していない可能性があります。切断せずにアプリケーションが終了すると、データベース・マネージャーがデータベースとの接続を切断します。
SQLM_INTR	**要求が割り込みを受けました:** 要求の割り込みが進行しています。
SQLM_IOERROR_WAIT	**表スペースを使用不可にするための待機:** アプリケーションが入出力エラーを検出し、特定の表スペースを使用不可にしようとしています。アプリケーションは、表スペースを使用不可にする前に、表スペース上のその他のすべてのアクティブなトランザクションが完了するまで待機する必要があります。
SQLM_LOAD	**データの高速ロード:** アプリケーションは、データベースにデータの「高速ロード」を実行中です。
SQLM_LOCKWAIT	**ロック待機:** 作業単位がロックを待機中です。ロックが付与されると、状況は直前の値に復帰します。

API 定数	説明
SQLM_QUIESCE_TABLESPACE	**表スペースの静止中:** アプリケーションが表スペース静止要求を実行中です。
SQLM_RECOMP	**再コンパイル中:** データベース・マネージャーがアプリケーションに代わってプランを再コンパイル (再バインド) 中です。
SQLM_REMOTE_RQST	**フェデレーテッド要求ペンディング:** アプリケーションはフェデレーテッド・データ・ソースからの結果を待っています。
SQLM_RESTART	**データベース再始動中:** アプリケーションは、クラッシュ・リカバリーを実行するためにデータベースを再始動しています。
SQLM_RESTORE	**データベースのリストア中:** アプリケーションは、バックアップ・イメージをデータベースにリストアしています。
SQLM_ROLLBACK_ACT	**ロールバック・アクティブ:** 作業単位がデータベースの変更をロールバックしています。
SQLM_ROLLBACK_TO_SAVEPOINT	**セーブポイントへのロールバック:** アプリケーションがセーブポイントにロールバック中です。
SQLM_TEND	**トランザクション終了:** 作業単位は、終了したグローバル・トランザクションの一部ですが、2 フェーズ・コミット・プロトコルの準備段階にはまだ入っていません。
SQLM_THABRT	**トランザクションをヒューリスティックにロールバック:** 作業単位は、ヒューリスティックにロールバックされたグローバル・トランザクションの一部です。
SQLM_THCOMT	**トランザクションをヒューリスティックにコミット:** 作業単位は、ヒューリスティックにコミットされたグローバル・トランザクションの一部です。
SQLM_TPREP	**トランザクション準備済み:** 作業単位は、2 フェーズ・コミット・プロトコルの準備段階に入っているグローバル・トランザクションの一部です。
SQLM_UNLOAD	**データの高速アンロード:** アプリケーションは、データベースからデータの「高速アンロード」を実行中です。
SQLM_UOWEXEC	**作業単位実行中:** データベース・マネージャーが作業単位に代わって要求を実行中です。
SQLM_UOWQUEUED	**キューに入れられた作業単位:** 作業単位はキューに入れられており、別のアクティビティーが実行を完了するのを待機しています。作業単位がキューに入れられているのは、並行実行アクティビティー数のしきい値に達しているためです。
SQLM_UOWWAIT	**作業単位の待機中:** データベース・マネージャーがアプリケーション内の作業単位に代わって待機中です。通常、この状況はシステムがアプリケーションのコード内で実行中であることを示します。

API 定数	説明
SQLM_WAITFOR_REMOTE	**リモート要求ペンディング:** アプリケーションは、パーティション・データベース・インスタンス内のリモート・パーティションからの応答を待っています。

application_handle - アプリケーション・ハンドル : モニター・エレメント

システム全体での、アプリケーションのユニーク ID。単一メンバーのデータベース構成の場合、この ID は 16 ビット・カウンターで構成されます。複数メンバーの構成の場合には、この ID はコーディネーター・メンバー番号と 16 ビット・カウンターが連結されて構成されます。さらに、アプリケーションが 2 次接続を行う可能性のあるメンバーには、すべて同一の ID が使用されます。

表 245. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 246. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本
トランザクション	event_xact	-

表 247. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
ロッキング	-	常に収集される
作業単位	-	常に収集される
接続	event_connheader	常に収集される
ステートメント	event_stmt	常に収集される
ステートメント	event_subsection	常に収集される
デッドロック[1]	event_dlconn	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
アクティビティー	event_activity	常に収集される
変更履歴	changesummary	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このモニター・エレメントは、**agent_id** モニター・エレメントの別名です。

このモニター・エレメントは、MON_GET_MEMORY_POOL によって戻される場合、記述されているメモリー・プールが以下のタイプのいずれかの場合を除き、NULL になります。

- APPLICATION
- STATISTICS
- STATEMENT
- SORT_PRIVATE

appls_cur_cons - 現在接続されているアプリケーション

現在データベースに接続されているアプリケーションの数を示します。

表 248. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
ロック	db_lock_list	基本

使用法 このエレメントを使用して、データベース内のアクティビティー・レベルおよび使用中のシステム・リソースの量を確認することができます。

maxappls および *max_coordagents* 構成パラメーターの設定値を調整するときに利用できます。例えば、この値が *maxappls* の値と常に同じ場合は、

maxappls の値を増やすことができます。詳しくは、『*rem_cons_in*』および『*local_cons*』モニター・エレメントの項を参照してください。

appls_in_db2 - データベースで現在実行中のアプリケーション

現在データベースに接続されており、データベース・マネージャーが要求を処理中のアプリケーションの数を示します。

表 249. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

arm_correlator アプリケーション応答測定相関関係子 : モニター・エレメント

アプリケーション応答測定 (ARM) 標準のトランザクションの ID。

表 250. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを使用すると、アクティビティー・イベント・モニターによって収集されるアクティビティーに関連付けられたアプリケーションもアプリケーション応答測定 (ARM) 標準をサポートする場合には、このアクティビティーをそのアプリケーションにリンクさせることができます。

associated_agents_top - 関連エージェント最大数

このアプリケーションに関連付けられているサブエージェントの最大数。

表 251. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

async_read_time - 非同期読み取り時間のモニター・エレメント

非同期エンジン・ディスパッチ可能単位 (EDU) がバッファー・プールまたは表スペースからの読み取りに費やした合計時間。この値はミリ秒単位で報告されます。

表 252. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

async_write_time - 非同期書き込み時間のモニター・エレメント

非同期エンジン・ディスパッチ可能単位 (EDU) がバッファー・プールまたは表スペースへの書き込みに費やした合計時間。この値はミリ秒単位で報告されます。

表 253. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

async_runstats - 非同期 RUNSTATS 要求の合計数 : モニター・エレメント

データベース内のすべてのアプリケーションのリアルタイム統計収集により正常に実行された非同期 RUNSTATS アクティビティーの合計数。すべてのデータベース・パーティションで報告された値が集約されます。

表 254. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
SNAP_GET_DB_V97 表関数 - dbase 論理グループからのスナップショット情報の検索	
SNAPDB 管理ビューおよび SNAP_GET_DB 表関数 - dbase 論理グループからのスナップショット情報の検索	

表 255. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 256. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このエレメントを使用すると、リアルタイム統計収集により正常に実行された非同期 RUNSTATS アクティビティーの数を判別できます。この値は頻繁に変わります。システム使用量をより正確に知るには、長期にわたり特定のインターバルを設けてスナップショットを取ってください。このエレメントを **sync_runstats** および **stats_fabrications** モニター・エレメントと組み合わせて使用すると、リアルタイム統計収集に関連した異なるタイプの統計収集アクティビティーを追跡し、それら

のパフォーマンスへの影響を分析する助けになります。

audit_events_total - 監査イベントの合計 : モニター・エレメント

生成された監査イベントの合計数。

表 257. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 257. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 258. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント

監査レコードの書き込みの待機にかかった時間。値はミリ秒単位で示されます。

表 259. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES _BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 259. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 260. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 260. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、エージェントが監査イベントをオープンし、ディスクに同期的に書き込むのを待機していた時間を判別します。

典型的なシナリオでは、監査ログ・ファイルのオープンを試行するのは一度に 1 つのエージェントのみであり、他のエージェントは監査共通サブシステムにアクセスできるようになるのを待機してから、そのファイルをオープンします。そのため、この待機時間は通常、オペレーティング・システムがディスクへのファイルの書き込みを待機していた時間を表します。監査ユーティリティーは実行時に監査ログ・ファイルをロックすることがあります。その場合、エージェントが監査ログ・ファイルをオープンしてそれに書き込むのを待機する時間が通常より長くなります。非同期監査が使用可能になっている場合、非同期監査バッファーより大きい監査イベントはバッファーにではなくディスクに直接書き込まれます。この時間も、待機時間に含まれます。

特別な監査ユーティリティーを使用する場合を除いて、待機時間は、ディスクの速度と、オペレーティング・システムがどれだけの速度でディスクにデータを書き込めるかに依存します。特定のアプリケーションおよび監査構成でこの待機時間を短くするには、オペレーティング・システムをチューニングするか、より高速のディスクを使用することができます。

audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント

監査イベントをディスクに直接書き込むためにエージェントが待機しなければならなかった回数の合計。

表 261. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 261. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 262. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

表 262. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを **audit_file_write_wait_time** モニター・エレメントと組み合わせて使用して、アプリケーション要求が監査イベントをオープンし、ディスクに同期的に書き込むのを待機していた平均時間を判別します。

audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント

監査バッファー内のスペースが空くのを待機した時間。待機は、監査バッファーが満杯になり、監査デーモンがバッファーをディスクに書き込むのをエージェントが待機しなければならない場合に発生します。値はミリ秒単位で示されます。

表 263. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 263. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 264. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、監査共通サブシステムが他のエージェントのイベント処理でビジー状態である間、監査共通サブシステムにアクセスするためにエージェントが待機する時間を判別します。

監査サブシステムの特定の共通部分には、一度に 1 つのエージェントしかアクセスできません。このモニター・エレメントの値は、監査共通サブシステムにアクセスするためにエージェントが待機しなければならない時間を示します。これには、現在の非同期バッファーを満たしたエージェントが、監査デーモンが前の非同期バッファーをディスクに書き出し終わるのを待機していた時間が含まれます。監査ログ・ファイルへの書き込みの間に待機している、または監査デーモンの要求を出すのを待機している他のエージェントも、監査共通サブシステムにアクセスしており、そこでの待機時間もこの値に反映されます。

非同期監査が使用されている場合、**audit_buf_sz** 構成パラメーターの値を変更すると、この待機時間を短くできる場合があります。 **audit_buf_sz** 構成パラメーターの値を増やしていき、それ以上値を大きくしても監査共通サブシステムの待機時間が少なくならないという値を見つけることができます。その時点では、次のバッファーが満杯になる前にデーモンが 1 つのバッファー全体をディスクに書き込むことができるほど、非同期バッファーの大きさが十分になっています。その場合、デーモンがボトルネックになることはなくなります。 **audit_buf_sz** 構成パラメーターをそのような値にまで増やすと、システム障害が起きた場合に失われる可能性がある監査レコードの数が多すぎるという場合には、オペレーティング・システムをチューニングするか、より高速なディスクを使用することによって、待機時間を短くすることができます。待機時間をさらに短くする必要がある場合は、監査ポリシーを使用して、生成される監査イベントの数を減らしてください。

audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント

監査がバッファーの書き込みを待機した回数。

表 265. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 265. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 266. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、エージェントが監査共通サブシステムにアクセスしなければならなかった合計回数を判別します。 1 つの監査イベントを生成する場合、イベントを記録するために、監査共通サブシステムへのアクセスが必要ない場合もあれば、1 回のアクセスが必要な場合、あるいは複数回のアクセスが必要な場合もあります。生成された監査イベントの正確な数を判別するには、**audit_events_total** モニター・エレメントを使用します。

auth_id 許可 ID

モニターされているアプリケーションを呼び出したユーザーの許可 ID。この ID は、DB2 Connect のゲートウェイ・ノード上では、ホスト上にあるユーザーの許可 ID となります。

表 267. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE

表 268. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本

表 269. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
接続	event_connheader	常に収集される

使用法

明示的なトラステッド接続では、ユーザーを切り替えた直後に **auth_id** 値が変更されるわけではありません。直後ではなく、ユーザーを切り替えてから最初にデータベースにアクセスしたときに、**auth_id** が更新されます。これは、ユーザー切り替え操作には必ずその後の操作が続くためです。

このエレメントを使用すると、アプリケーションを呼び出したユーザーを判別できます。

authority_bitmap ユーザー許可レベル : モニター・エレメント

ユーザー、およびユーザーが所属するグループに付与された権限。これには、ユーザーに付与されたロールに付与された権限、およびユーザーが所属するグループに付与されたロールに付与された権限も含まれます。

ユーザーに付与された権限、またはユーザーに付与されたロールに付与された権限は、ユーザー権限と見なされます。ユーザーが所属するグループに付与された権限、またはユーザーが所属するグループに付与されたロールに付与された権限は、グループ権限と見なされます。

表 270. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本
アプリケーション	appl_info	基本

表 271. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法

authority_bitmap モニター・エレメントには、配列の形式があります。各配列エレメントは、ユーザー ID に特定の権限が付与されているかどうか、およびユーザーがどのようにその権限を受け取ったかを示す単一文字です。

個々の配列エレメントは、sql.h ファイルで定義された索引値によって索引付けされています。 authority_bitmap 配列の索引の値は、*権限索引*と呼ばれています。例えば、SQL_DBAUTH_SYSADM は、ユーザーに SYSADM 権限があるかを判別するための索引です。

権限索引で識別される authority_bitmap 配列にある 1 つのエレメントの値は、権限が許可 ID により保持されているかを示します。許可 ID が保持される方法を判別するには、権限索引により識別される配列エレメントごとに、sql.h の以下の定義を使用してください。

SQL_AUTH_ORIGIN_USER
このビットがオンである場合、許可 ID には、ユーザーに付与された権限、またはユーザーに付与されたロールに付与された権限があります。

SQL_AUTH_ORIGIN_GROUP
このビットがオンである場合、許可 ID には、グループに付与された権限、またはグループに付与されたロールに付与された権限があります。

例えば、ユーザーが DBADM 権限を保持しているかを判別するには、以下の値を確認します。

```
authority_bitmap[SQL_DBAUTH_DBADM]
```

DBADM 権限がユーザーにより直接保持されているかを判別するには、以下を確認します。

```
authority_bitmap[SQL_DBAUTH_DBADM] & SQL_AUTH_ORIGIN_USER
```

authority_lvl ユーザー許可レベル : モニター・エレメント

アプリケーションに対して付与されている最高権限レベル。

注: authority_lvl モニター・エレメントは、DB2 データベース・バージョン 9.5 以降では推奨されていません。その代わりに、authority_bitmap モニター・エレメントを使用してください。 866 ページの『authority_bitmap ユーザー許可レベル : モニター・エレメント』を参照してください。

表 272. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本
アプリケーション	appl_info	基本

表 273. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法 アプリケーションで許可される操作は、直接または間接的に付与されます。

sql.h の次の定義を使用して、ユーザーに明示的に付与されている許可を判別できます。

- SQL_SYSADM
- SQL_DBADM
- SQL_CREATETAB
- SQL_BINDADD
- SQL_CONNECT
- SQL_CREATE_EXT_RT
- SQL_CREATE_NOT_FENC
- SQL_SYSCTRL
- SQL_SYSMAINT

sql.h の次の定義を使用して、グループまたは PUBLIC 権限から継承されている間接許可を判別できます。

- SQL_SYSADM_GRP
- SQL_DBADM_GRP
- SQL_CREATETAB_GRP
- SQL_BINDADD_GRP
- SQL_CONNECT_GRP
- SQL_CREATE_EXT_RT_GRP
- SQL_CREATE_NOT_FENC_GRP
- SQL_SYSCTRL_GRP
- SQL_SYSMAINT_GRP

auto_storage_hybrid - ハイブリッド自動ストレージの表スペース標識 : モニター・エレメント

表スペースが非自動ストレージ・コンテナーを含む自動ストレージ表スペースである場合、このモニター・エレメントは値 1 を戻します。それ以外の場合は、値 0 を戻します。

表 274. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

使用法

ハイブリッド自動ストレージ表スペースは、ALTER TABLESPACE コマンドを使用して自動ストレージによって管理するように変換されているものの、まだリバランスされていない表スペースです。この表スペースには、依然として非自動ストレージ・コンテナーが含まれます。リバランスされた後、表スペースには自動ストレージ・コンテナーのみが含まれ、ハイブリッド表スペースと見なされなくなります。

automatic - バッファー・プール自動 : モニター・エレメント

特定のバッファー・プールでセルフチューニングが使用可能になっているかどうかを示します。このエレメントは、バッファー・プールでセルフチューニングが使用可能になっている場合は 1 に、使用可能になっていない場合は 0 に設定されます。

表 275. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

backup_timestamp - バックアップ・タイム・スタンプ

バックアップ・イメージのタイム・スタンプ

表 276. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
変更履歴	changesummary	常に収集される

使用法

変更履歴イベント・モニターの場合:

- UTILITY_TYPE が BACKUP で、EVENT_TYPE が UTILSTART である場合、BACKUP_TIMESTAMP 値はバックアップ・イメージのタイム・スタンプです。UTILITY_TYPE が RESTORE で、EVENT_TYPE が UTILSTOP である場合、

BACKUP_TIMESTAMP 値はバックアップ・イメージのタイム・スタンプです。それ以外の場合、BACKUP_TIMESTAMP は空ストリングです。

- RESTORE の場合、イメージのタイム・スタンプは、ユーティリティーの開始時に明らかであるとは限りません。

BACKUP_TIMESTAMP は、 SYSIBMADM.DB_HISTORY 管理ビューを使用して、データベース履歴ファイルに保管されている情報と照合する (シーケンス情報を参照するなど) ことができます。

bin_id ヒストグラム・ビン ID : モニター・エレメント

ヒストグラム・ビンの ID。**bin_id** はヒストグラム内で固有です。

表 277. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントを使用すると、同じヒストグラム内でビンを区別できます。

binds_precompiles 試行されたバインド/プリコンパイル

試みられたバインドおよびプリコンパイルの数。

エレメント ID
binds_precompiles

エレメント・タイプ
カウンター

表 278. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 279. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、データベース・マネージャー内の現在のアクティビティーのレベルがわかります。

この値には *int_auto_rebinds* のカウントは含まれませんが、 REBIND PACKAGE コマンドの結果として起こるバインド数は含まれます。

block_ios - ブロック入出力要求数 : モニター・エレメント

ブロック入出力要求の数。より厳密には、 DB2 がバッファー・プールのブロック域に対してページの順次プリフェッチを実行する回数。

表 280. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 281. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール

使用法

ブロック・ベースのバッファー・プールを使用可能にすると、このモニター・エレメントがブロック入出力の実行頻度を報告します。それ以外の場合、このモニター・エレメントは 0 を戻します。ブロック入出力要求の数がモニターされるのは、ブロック・ベースのバッファー・プールの使用中に順次プリフェッチが実行される間だけです。

ブロック・ベースのバッファー・プールを使用可能にしてあり、この数が非常に小さい場合、またはベクトル化入出力の数 (**vectored_ios** モニター・エレメントの値) に近い場合は、ブロック・サイズを変更することを考慮してください。このようなときは、以下の状態の発生を示している場合があります。

- バッファー・プールにバインドされている 1 つ以上の表スペースのエクステント・サイズが、バッファー・プールに指定されているブロック・サイズよりも小さい。
- プリフェッチ要求されたページのうち、いくつかのページがバッファー・プールのページ・エリアにすでに存在している。

プリフェッチャーはそれぞれのバッファー・プール・ブロックに無駄なページが多少あっても見過ごしますが、無駄なページの数が非常に多くなると、プリフェッチャーはバッファー・プールのページ・エリアにベクトル化入出力を実行します。

ブロック・ベースのバッファー・プールを使用することで順次プリフェッチのパフォーマンスが改善される点を活用するには、ブロック・サイズに適切な値を選ぶことが非常に重要です。しかし、エクステント・サイズの異なる複数の表スペースが同じブロック・ベースのバッファー・プールにバインドされていることがあるので、値の選択が難しいこともあります。最適なパフォーマンスを得るためには、同じエクステント・サイズの表スペースを、ブロック・サイズがエクステント・サイズと等しいブロック・ベースのバッファー・プールにバインドすることをお勧めし

ます。表スペースのエクステント・サイズがブロック・サイズより大きい場合にも良好なパフォーマンスが得られますが、ブロック・サイズよりも小さい場合にはパフォーマンスが低下します。

例えば、エクステント・サイズが 2 でブロック・サイズが 8 の場合は、ブロック入出力の代わりにベクトル化入出力が使用されます (ブロック入出力では 6 ページの無駄が発生します)。ブロック・サイズを 2 に下げると、この問題は解決できます。

blocking_cursor ブロック・カーソル

このエレメントは、実行中のステートメントがブロッキング・カーソルを使用しているかどうかを示します。

エレメント ID
blocking_cursor

エレメント・タイプ
情報

表 282. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 283. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック	event_detailed_dlconn	常に収集される
ステートメント	event_stmt	常に収集される

使用法 照会のデータ転送にブロッキングを使用すると、パフォーマンスが改善されます。照会に使用される SQL はブロッキングの使用と関係があるため、多少の変更が必要になる場合があります。

blocks_pending_cleanup クリーンアップ保留中のロールアウト済みブロック : モニター・エレメント

ロールアウト削除に続いて非同期クリーンアップを保留中のデータベースにおける MDC 表ブロックの合計数。

表 284. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMINTABINFO 管理ビューおよび ADMIN_GET_TAB_INFO 表関数 - 表のサイズおよび状態に関する情報の検索	ACTIVITY METRICS BASE

表 285. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-
データベース	event_db	-

使用法

このエレメントを使用すると、延期できるクリーンアップ・ロールアウトの削除の後で使用可能なストレージとしてシステムに解放されていない MDC 表ブロックの数を判別できます。

bottom ヒストグラム・ビンの最下位 : モニター・エレメント

ヒストグラム・ビンの範囲外の最終点。このモニター・エレメントの値は、前のヒストグラム・ビンがある場合には、その最終範囲に含まれる最初の値になります。

表 286. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントと対応する **top** エレメントを一緒に使用して、ヒストグラム中のビンの範囲を判別します。

boundary_leaf_node_splits - 境界リーフ・ノードの分割 : モニター・エレメント

挿入操作時に境界リーフ・ノードが分割された回数。

表 287. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

bp_cur_buffsz - バッファー・プールの現行サイズ

現行のバッファー・プール・サイズ (ページ数)。

表 288. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	ACTIVITY METRICS BASE

表 289. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool_nodeinfo	バッファー・プール

bp_id バッファー・プール ID : モニター・エレメント

このエレメントには、モニターされているバッファー・プールのバッファー・プール ID が含まれます。

表 290. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	基本

bp_name - バッファー・プール名 : モニター・エレメント

バッファー・プールの名前。

表 291. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	ACTIVITY METRICS BASE

表 292. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	基本

使用法 各データベースには少なくとも 1 つのバッファー・プールが必要です。必要に応じて、単一のデータベースのためにそれぞれサイズの違ういくつかのバッファー・プールを作成できます。 CREATE、ALTER、および DROP BUFFERPOOL ステートメントを使用して、バッファー・プールを作成、変更、ドロップすることが可能です。

データベースを作成する場合、データベースには IBMDEFAULTBP というデフォルトのバッファー・プールが設定され、そのサイズはプラットフォームによって決まります。さらに、それぞれ異なるページ・サイズに対応する次のようなシステム・バッファー・プールも設定されます。

- IBMSYSTEMBP4K
- IBMSYSTEMBP8K
- IBMSYSTEMBP16K
- IBMSYSTEMBP32K

これらのシステム・バッファー・プールは変更できません。

bp_new_buffsz 新規バッファー・プール・サイズ

データベースが再始動されるとバッファー・プールが変更されるサイズ。 ALTER BUFFERPOOL ステートメントが DEFERRED として実行されると、データベースを停止するか再始動するまでバッファー・プール・サイズは変更されません。

表 293. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool_nodeinfo	バッファー・プール

bp_pages_left_to_remove 除去残ページ数

バッファー・プールのサイズ変更が完了する前に、バッファー・プールからの除去が残っているページ数。これは IMMEDIATE として実行される ALTER BUFFERPOOL ステートメントによって呼び出されるバッファー・プール・サイズ変更操作にのみ適用されます。

表 294. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool_nodeinfo	バッファー・プール

bp_tbsp_use_count バッファー・プールにマップされている表スペースの数

このバッファー・プールを使用している表スペースの数。

表 295. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool_nodeinfo	バッファー・プール

buff_auto_tuning - FCM バッファー自動チューニング標識 : モニター・エレメント

高速コミュニケーション・マネージャー (FCM) バッファー数の設定とチューニングが自動的に行われるかどうかを示します。値 1 は「はい」を意味し、値 0 は「いいえ」を意味します。

表 296. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

使用法

FCM バッファー自動チューニングは、**fcm_num_buffers** 構成パラメーターを AUTOMATIC に設定することによって有効になります。

buff_free - 現在空いている FCM バッファー

このエレメントは、現在空いている FCM バッファーの数を示します。

エレメント ID
buff_free

エレメント・タイプ
ゲージ

表 297. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

表 298. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法

フリー FCM バッファーのパーセンテージを計算するには、以下の公式を使用します。

```
(buff_free/buff_total) * 100
```

フリー FCM バッファーのパーセンテージが 20% を下回る場合、FCM バッファー自動チューニングが有効であれば、DB2 データベース・マネージャーは FCM バッファー番号を調整します。

フリー FCM バッファーのパーセンテージが 20% を下回る場合、FCM バッファー自動チューニングが無効であれば、**fcm_num_buffers** 構成パラメーターを調整する必要があります。

buff_free_bottom 空き FCM バッファーの最小数

処理中に到達した空き FCM バッファーの最小数。

表 299. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

表 300. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法

このエレメントを **fcm_num_buffers** 構成パラメーターを共に使用すると、FCM バッファー・プール使用率の最大値を判別できます。 **buff_free_bottom** モニター・エレメントの値が小さい場合は、**fcm_num_buffers** 構成パラメーターの値を大きく

して、処理中に FCM バッファーが不足しないようにしてください。
buff_free_bottom モニター・エレメントの値が大きい場合は、**fcm_num_buffers** 構成パラメーターの値を小さくして、システム・リソースを節約してください。

buff_max - FCM バッファーの最大可能数 : モニター・エレメント

高速コミュニケーション・マネージャー (FCM) バッファーの割り振り可能最大数。この値は、インスタンスの開始時に予約された仮想メモリー量に基づいています。

表 301. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

表 302. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法

この内部モニター・エレメントを使用するのは、IBM サポートだけです。

buff_total - 現在割り振られている FCM バッファー数 : モニター・エレメント

現在割り振られている高速コミュニケーション・マネージャー (FCM) バッファーの数。この数には、使用中バッファーと空きバッファーの両方が含まれます。

表 303. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

表 304. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法

FCM が自動的にチューニングされていることを **buff_auto_tuning** モニター・エレメントが示す場合、**buff_total** モニター・エレメントの値は FCM バッファーの需要に基づいて調整されます。

現在使用されている FCM バッファー数を判別するには、以下の公式を使用します。

```
buff_total - buff_free
```

フリー FCM バッファーのパーセンテージを計算するには、以下の公式を使用します。

```
(buff_free/buff_total) * 100
```

フリー FCM バッファーのパーセンテージが 20% を下回る場合、FCM バッファー自動チューニングが有効であれば、DB2 データベース・マネージャーは FCM バッファー番号を調整します。

フリー FCM バッファーのパーセンテージが 20% を下回る場合、FCM バッファー自動チューニングが無効であれば、**fcm_num_buffers** 構成パラメーターを調整する必要があります。

byte_order イベント・データのバイト・オーダー

数値データのバイト配列。具体的には、イベント・データ・ストリームが「ビッグ・エンディアン」サーバー (RS/6000® など) で生成されたか、あるいは「リトル・エンディアン」サーバー (Windows 2000 が稼働する Intel 系 PC) で生成されたかを示します。

表 305. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	常に収集される

使用法 「ビッグ・エンディアン」サーバーの整数のバイト・オーダーは、「リトル・エンディアン」サーバーのバイト・オーダーとは逆になるため、データ・ストリームの数値データを解釈するときにこの情報が必要になります。

データを処理するアプリケーションが一方のタイプのコンピューター・ハードウェア上で実行していることを認識し (例えば、ビッグ・エンディアン・コンピューター)、イベント・データがもう一方のタイプのコンピューター・ハードウェア上で生成された場合 (例えば、リトル・エンディアン・コンピューター)、モニター・アプリケーションはこれらのデータを解釈する前に数値データ・フィールドのバイトを逆転する必要があります。タイプが同じ場合は、バイトの再配列は必要ありません。

このエレメントは、次のいずれかの API 定数に設定できます。

- SQLM_BIG_ENDIAN
- SQLM_LITTLE_ENDIAN

cached_timestamp - キャッシュ・タイム・スタンプのモニター・エレメント

サーバー・リストがキャッシュに入れられた時刻。

表 306. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVERLIST 表関数 - メンバーの優先順位の詳細を取得	常に収集される

call_stmt_routine_id - CALL ステートメント・ルーチン ID のモニター・エレメント

CALL ステートメントの場合、これは、呼び出し中のプロシージャーのルーチン ID です。

行が CALL ステートメントに対応していない場合、このエレメントは NULL を返します。

表 307. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される

使用法

このエレメントを使用すると、組み込みルーチン呼び出しおよび再帰的ルーチン呼び出しの階層を再作成できます。

call_stmt_subroutine_id - CALL ステートメント・サブルーチン ID のモニター・エレメント

サブルーチンに対する CALL ステートメントの場合、これは呼び出し中のプロシージャーのサブルーチン ID です。

表 308. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される

使用法

このエレメントを使用すると、組み込みルーチン呼び出しおよび再帰的ルーチン呼び出しの階層を再作成できます。

cat_cache_heap_full - カタログ・キャッシュ・ヒープ・フルのモニター・エレメント

カタログ・キャッシュへの挿入が、データベース・ヒープのヒープ満杯状態により失敗した回数。

表 309. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
接続	event_conn	常に収集される

使用法

カタログ・キャッシュは、データベース・ヒープから動的にストレージを取り出します。キャッシュ・ストレージが限度に達していなくても、データベース・ヒープのスペースがないために、カタログ・キャッシュへの挿入が失敗する可能性があります。カタログ・キャッシュ・ヒープ・フルのカウントがゼロではない場合、データベース・ヒープ・サイズを増加するか、カタログ・キャッシュ・サイズを削減することで、この挿入に失敗する状態を直すことができます。

cat_cache_inserts - カタログ・キャッシュ挿入数 : モニター・エレメント

システムが表記述子または許可情報をカタログ・キャッシュに挿入しようとした回数。

表 310. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 310. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 311. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 312. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

カタログ・キャッシュ検索と組み合わせると、次の公式を使用してカタログ・キャッシュ・ヒット率を計算できます。

```
1 - (カタログ・キャッシュ挿入数 / カタログ・キャッシュ検索数)
```

このエレメントの使用法について詳しくは、**cat_cache_lookups** モニター・エレメントを参照してください。

cat_cache_lookups - カタログ・キャッシュ検索 : モニター・エレメント

表の記述子情報または許可情報を取得するためにカタログ・キャッシュが参照された回数。

表 313. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 314. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 315. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このエレメントには、カタログ・キャッシュへの正常に行われたアクセスと失敗したアクセスの両方が含まれます。カタログ・キャッシュは、次の場合に参照されます。

- SQL ステートメントのコンパイル中に、表、ビュー、または別名を処理したとき。
- データベース許可情報にアクセスがあったとき。
- SQL ステートメントのコンパイル中にルーチンを処理したとき。

カタログ・キャッシュ・ヒット率の計算には次の公式を使用します。

```
(1 - (cat_cache_inserts / cat_cache_lookups))
```

この値は、カタログ・キャッシュがどの程度カタログ・アクセスを回避しているかを示します。この比率が高い場合 (0.8 を超える値)、キャッシュは効果的に動作しています。比率が低い場合は、 **catalogcache_sz** 構成パラメーターを大きくする必要があることを示します。データベースへの最初の接続の直後は、この比率は高くなります。

表、ビュー、別名などに関係するデータ定義言語 (DDL) SQL ステートメントは、そのようなオブジェクトに関する表記述子情報をカタログ・キャッシュから取り除くため、それらのオブジェクトは次の参照で再挿入されることになります。さらに、データベース許可およびルーチンの実行特権のための GRANT および REVOKE のステートメントによって、該当する許可情報がカタログ・キャッシュから取り除かれます。したがって、DDL ステートメントと GRANT/REVOKE ステートメントを多用した場合も、この比率は大きくなります。

cat_cache_overflows カタログ・キャッシュ・オーバーフロー数

割り振られたメモリーの境界からカタログ・キャッシュがオーバーフローした回数。

表 316. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 316. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 317. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法

このエレメントと **cat_cache_size_top** モニター・エレメントを組み合わせて使用すると、オーバーフローを回避するのにカタログ・キャッシュのサイズを大きくする必要があるかどうかを判別できます。

カタログ・キャッシュのスペースは、どのトランザクションでも現在使用されていない、表、ビュー、または別名に関する表記述子情報や、許可情報を除去することで取り戻します。

cat_cache_overflows モニター・エレメントの値が大きい場合は、ワークロードに対してカタログ・キャッシュが小さすぎることが考えられます。カタログ・キャッシュを大きくすると、パフォーマンスが改善されることがあります。多数の表、ビュー、別名、ユーザー定義関数またはストアード・プロシージャーを参照する多数の SQL ステートメントを、 1 つの作業単位にコンパイルするトランザクションがワークロードに含まれている場合は、 1 つのトランザクションにコンパイルする SQL ステートメントの数を少なくすると、カタログ・キャッシュのパフォーマンスが改善されることがあります。あるいは多数の表、ビュー、別名、ユーザー定義関数またはストアード・プロシージャーを参照する多数の SQL ステートメントが入ったパッケージのバインドがワークロードに含まれる場合は、パッケージを分割してその中に含まれる SQL ステートメントの数を少なくすると、パフォーマンスが改善されることがあります。

cat_cache_size_top - カタログ・キャッシュの最高水準点 : モニター・エレメント

カタログ・キャッシュが到達した最大論理サイズ。

表 318. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 319. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このエレメントは、データベースが活動化されて以降、データベースでワークロードを実行したときに論理的に必要となったカタログ・キャッシュの最大バイト数を示します。

カタログ・キャッシュは論理サイズで管理されます。論理サイズにはメモリー管理のための使用量が含まれません。データベース・スナップショット内の **pool_watermark** エレメントは、カタログ・キャッシュによって使用されたメモリーの物理最高水準点値を示します。カタログ・キャッシュのモニター作業とチューニング作業には、物理サイズよりも論理サイズを使用するようにしてください。

カタログ・キャッシュがオーバーフローした場合、このエレメントは、オーバーフロー時にカタログ・キャッシュが到達した最大サイズになります。このような状態が発生したかどうかを確認するには、**cat_cache_overflows** モニター・エレメントをチェックしてください。

ワークロードに必要なカタログ・キャッシュの最小サイズは次のように決定できます。

```
maximum catalog cache size / 4096
```

この結果を切り上げた整数が、オーバーフローを避けるためにカタログ・キャッシュが必要とする 4K ページの最小数になります。

catalog_node カタログ・ノード番号

データベース・カタログ表が保管されているノードのノード番号。

エレメント ID
catalog_node

エレメント・タイプ
情報

表 320. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 321. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 カタログ・ノードは、すべてのシステム・カタログ表が保管されているノードです。システム・カタログ表へのアクセスは、すべてこのノードを通る必要があります。

catalog_node_name カタログ・ノード・ネットワーク名

カタログ・ノードのネットワーク名。

エレメント ID

catalog_node_name

エレメント・タイプ

情報

表 322. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 323. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントを使用して、データベースのロケーションを判別できます。

cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント

DB2 データベース・システムがクラスター・キャッシング・ファシリティーと通信した際に待機した回数。

表 324. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 325. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント

cf_wait_time モニター・エレメントは、クラスター・キャッシング・ファシリティーと通信するために費やした時間を格納します。

この時間には、ロックの認可やページ再利用の実行など、要求された処理または通信の結果として生じる場合がある処理を実行するための時間は含まれません。測定単位はミリ秒です。

表 326. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 326. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

使用法

この値は、DB2 がクラスター・キャッシング・ファシリティーとの通信の際に待機に費やした時間の指標です。

cfg_collection_type - 構成収集タイプ

構成パラメーター値がいつ収集されたかを示します。

表 327. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DBDBMCFG	常に収集される

使用法

変更履歴イベント・モニターで収集されるこの値は、以下のとおりです。

I イベント・モニターが活動化されたときにキャプチャーされた初期値。

U 更新された値

cfg_name - 構成名

構成パラメーターの名前。

表 328. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DBDBMCFG	常に収集される

使用法

変更履歴イベント・モニターでは、このエレメントは、DBCFG または DBMCFG イベントの一環として更新された構成パラメーター、または DBCFGVALUES または DBMCFGVALUES イベントの一環としてイベント・モニターの開始時にキャプチャーされた構成パラメーターを示します。これらのイベントは、以下が行われたことを表します。

DBCFG
データベース構成パラメーターの変更

DBMCFG
データベース・マネージャー構成パラメーターの変更

DBCFGVALUES
イベント・モニターが非アクティブであった間に、データベース構成パラメーターが変更されていた場合の、イベント・モニター開始時のデータベース構成パラメーター値のキャプチャー

DBMCFGVALUES
イベント・モニターが非アクティブであった間に、データベース・マネージャー構成パラメーターが変更されていた場合の、イベント・モニター開始時のデータベース・マネージャー構成パラメーター値のキャプチャー

cfg_old_value - 古い構成値

構成パラメーターの古い値、またはメモリー内の構成パラメーター値。

表 329. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DBDBMCFG	常に収集される

使用法

変更履歴イベント・モニターの場合:

- イベントが、データベース構成パラメーターの変更 (DBCFG) またはデータベース・マネージャー構成パラメーターの変更 (DBMCFG) である場合、これは構成パラメーターの古い値です。
- イベントが、イベント・モニターが非アクティブであった間に変更されていたデータベース構成パラメーター値のキャプチャー (DBCFGVALUES) またはデータベース・マネージャー構成パラメーター値のキャプチャー (DBMCFGVALUES) である場合、これは現行のメモリー内の構成パラメーター値です。これは、現在使用中の構成パラメーター値です。

cfg_old_value_flags - 古い構成値のフラグ

このフラグは、古い構成パラメーターの値がどのように決定されたかを示します。

表 330. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DBDBMCFG	常に収集される

使用法

変更履歴イベント・モニターでは、このエレメントは、古い構成パラメーターの値が決定された方法を示します。

- AUTOMATIC
- COMPUTED
- NONE

イベントが、イベント・モニターが非アクティブであった間に変更されていたデータベース構成パラメーター値のキャプチャー (DBCFGVALUES) またはデータベー

ス・マネージャー構成パラメーター値のキャプチャー (DBMCFGVALUES) である場合、このフラグは現行のメモリー内の構成パラメーター値を表します。

cfg_value - 構成値

構成パラメーターの新しい値、またはディスク上の構成パラメーター値。

表 331. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DBDBMCFG	常に収集される

使用法

変更履歴イベント・モニターの場合:

- イベントが、データベース構成パラメーターの変更 (DBCFG) またはデータベース・マネージャー構成パラメーターの変更 (DBMCFG) である場合、これは構成パラメーターの新しい値です。
- イベントが、イベント・モニターが非アクティブであった間に変更されていたデータベース構成パラメーター値のキャプチャー (DBCFGVALUES) またはデータベース・マネージャー構成パラメーター値のキャプチャー (DBMCFGVALUES) である場合、これは、ディスク上の構成パラメーター値です。ディスク上の構成パラメーター値は、最新の値であり、まだ反映されていない可能性があります。

cfg_value_flags - 構成値フラグ

このフラグは、新しい構成パラメーターの値がどのように決定されたかを示します。

表 332. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DBDBMCFG	常に収集される

使用法

変更履歴イベント・モニターでは、このエレメントは、新しい構成パラメーターの値が決定された方法を示します。

- AUTOMATIC
- COMPUTED
- NONE

イベントが、イベント・モニターが非アクティブであった間に変更されていたデータベース構成パラメーター値のキャプチャー (DBCFGVALUES) またはデータベース・マネージャー構成パラメーター値のキャプチャー (DBMCFGVALUES) である場合、このフラグは、現行のディスク上の構成パラメーター値を表します。

ch_auto_tuning - FCM チャネル自動チューニング標識 : モニター・エレメント

高速コミュニケーション・マネージャー (FCM) チャネル数の設定とチューニングが自動的に行われるかどうかを示します。値 1 は「はい」を意味し、値 0 は「いいえ」を意味します。

表 333. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

使用法

FCM チャネル自動チューニングは、**fcm_num_channels** 構成パラメーターを AUTOMATIC に設定することによって有効になります。

ch_free - 現在空いているチャネル

このエレメントは、現在空いている FCM 通信チャネルの数を示します。

表 334. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

表 335. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法

フリー FCM チャネルのパーセンテージを計算するには、以下の公式を使用します。

```
(ch_free/ch_total) * 100
```

フリー FCM チャネルのパーセンテージが 20% を下回る場合、FCM チャネル自動チューニングが有効であれば、DB2 データベース・マネージャーは FCM チャネル番号を調整します。

フリー FCM チャネルのパーセンテージが 20% を下回る場合、FCM チャネル自動チューニングが無効であれば、**fcm_num_channels** 構成パラメーターを調整する必要があります。

ch_free_bottom 空いているチャネルの最小

処理中に到達した空き FCM 通信チャネルの最小数。

表 336. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

表 337. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法

このモニター・エレメントを **fcm_num_channels** 構成パラメーターと併せて使用することにより、接続項目の最大使用率を判別できます。

ch_max - FCM チャネルの最大可能数 : モニター・エレメント

高速コミュニケーション・マネージャー (FCM) チャネルの割り振り可能最大数。この値は、インスタンスの開始時に予約された仮想メモリー量に基づいています。

表 338. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

表 339. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法

この内部モニター・エレメントを使用するのは、IBM サポートだけです。

ch_total - 現在割り振られている FCM チャネル数 : モニター・エレメント

現在割り振られている高速コミュニケーション・マネージャー (FCM) チャネルの数。この数には、使用中チャネルと空きチャネルの両方が含まれます。

表 340. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE

表 341. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法

FCM が自動的にチューニングされていることを **ch_auto_tuning** モニター・エレメントが示す場合、**ch_total** モニター・エレメントの値は FCM チャネルの需要に基づいて調整されます。

現在使用されている FCM チャネル数を判別するには、以下の公式を使用します。

```
ch_total - ch_free
```

フリー FCM チャネルのパーセンテージを計算するには、以下の公式を使用します。

```
(ch_free/ch_total) * 100
```

フリー FCM チャネルのパーセンテージが 20% を下回る場合、FCM チャネル自動チューニングが有効であれば、DB2 データベース・マネージャーは FCM チャネル番号を調整します。

フリー FCM チャネルのパーセンテージが 20% を下回る場合、FCM チャネル自動チューニングが無効であれば、**fcm_num_channels** 構成パラメーターを調整する必要があります。

client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント

この接続で sqleseti API が発行された場合に、ロギングおよび診断の目的でターゲット・データベースに渡されたデータです。この接続、作業単位、またはアクティビティーの CLIENT_ACCTNG 特殊レジスターの現行値。

注: このエレメントは、コーディネーター・メンバーでのみ報告されます。リモートのメンバーに報告される値は、長さ 0 のストリングです。

このモニター・エレメントは、**tpmon_acc_str** モニター・エレメントと同義です。**client_acctng** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_acc_str** モニター・エレメントは、表、ファイル、およびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 342. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE

表 342. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 343. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
ロッキング	-	-
作業単位	-	-
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

使用法

このエレメントは、問題判別およびアカウンティングの目的で使用します。

client_applname - クライアント・アプリケーション名 : モニター・エレメント

この接続で sqleseti API が発行された場合に、トランザクションを実行中のサーバー・トランザクション・プログラムを示します。この接続、作業単位、またはアクティビティーの CLIENT_APPLNAME 特殊レジスターの現行値。

注: このエレメントは、コーディネーター・メンバーでのみ報告されます。リモートのメンバーに報告される値は、長さ 0 のストリングです。

このモニター・エレメントは、**tpmon_client_app** モニター・エレメントと同義です。 **client_applname** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_app** モニター・エレメントは、表、ファイル、およびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 344. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 345. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
ロッキング	-	-
作業単位	-	-
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

使用法

このエレメントは、問題判別およびアカウンティングの目的で使用します。

client_db_alias アプリケーションで使用するデータベース別名

アプリケーションがデータベースに接続するときに指定するデータベースの別名。

エレメント ID

client_db_alias

エレメント・タイプ

情報

表 346. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本

表 347. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_connheader	常に収集される

使用法 このエレメントを使用して、アプリケーションがアクセスしている実際のデータベースを識別できます。この名前と *db_name* 間のマッピングには、クライアント・ノードとデータベース・マネージャーのサーバー・ノードにあるデータベース・ディレクトリーを使用します。

この別名は、データベース接続要求を発行したデータベース・マネージャー内に定義されている別名です。

データベースの別名が異なると認証タイプが異なるので、このエレメントを認証タイプの判別に利用することもできます。

client_hostname - クライアント・ホスト名 : モニター・エレメント

クライアント・アプリケーションの接続元のマシンのホスト名。

表 348. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 349. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
作業単位	-	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント

このモニター・エレメントは、クライアントによる次の要求の送信を待機していた時間を記録します。値はミリ秒単位で示されます。

表 350. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES _BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 351. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 351. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	常に収集される

使用法

このモニター・エレメントを使用して、要求の処理ではなく、クライアントからの要求の待機にかかった時間を判別します。クライアントのアイドル時間が大きい場合、サーバー上ではなくクライアント上で対処する必要のあるパフォーマンス上の問題を示している可能性があります。

client_nname - クライアント名モニター・エレメント

このモニター・エレメントは、使用しないでください。返される値は無効な値です。

表 352. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_connheader	
ステートメント	event_connheader	
デッドロック	event_connheader	
接続	event_connheader	

client_pid - クライアント・プロセス ID : モニター・エレメント

データベースへの接続を行ったクライアント・アプリケーションのプロセス ID。

表 353. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE

表 354. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本
DCS アプリケーション	dcs_appl_info	基本

表 355. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
作業単位	-	常に収集される
接続	event_connheader	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

使用法

このエレメントを使用して、CPU および入出力時間などのモニター情報をクライアント・アプリケーションに関連付けることができます。

DRDA AS 接続のとき、このエレメントは 0 に設定されます。

client_platform - クライアント・オペレーティング・プラットフォーム : モニター・エレメント

クライアント・アプリケーションが稼働中のオペレーティング・システム。

表 356. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 357. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本
DCS アプリケーション	dcs_appl_info	基本

表 358. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
作業単位	-	常に収集される
接続	event_connheader	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

使用法

このエレメントを使用して、リモート・アプリケーションの問題判別を行えます。このフィールドの値は、ヘッダー・ファイルの sqlmon.h にあります。

client_port_number - クライアント・ポート番号 : モニター・エレメント

TCP/IP 接続の場合、データベース・サーバーと通信するためにアプリケーションが使用しているクライアント・マシンのポート番号。

表 359. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 360. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
作業単位	-	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

client_prdid クライアント製品およびバージョン ID : モニター・エレメント

クライアント上で実行中の製品とバージョン。このモニター・エレメントは、client_product_id モニター・エレメントのシノニムです。

表 361. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 362. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
DCS アプリケーション	dcs_appl_info	基本

表 363. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される
接続	event_connheader	常に収集される
しきい値違反	event_thresholdviolations	常に収集される

使用法

このエレメントを使用すると、IBM Data Server Clientの製品とコード・バージョンを識別できます。 PPPVVRRM の形式になっています。各部分の定義は次のとおりです。

- PPP は製品を示し、DB2 製品の場合は 『SQL』 である。
- VV は 2 桁でバージョン番号を示す (バージョン番号が 1 桁の場合には、高位の桁は 0 になります)。
- RR は 2 桁でリリース番号を示す (リリース番号が 1 桁の場合には高位の桁は 0 になります)。
- M は 1 文字で修正レベルを示します (0-9 または A-Z)。

client_protocol - クライアント通信プロトコル : モニター・エレメント

クライアント・アプリケーションがサーバーとの通信に使用している通信プロトコル。

表 364. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 365. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本
DCS アプリケーション	dcs_appl_info	基本

表 366. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
作業単位	-	常に収集される
接続	event_connheader	常に収集される
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

使用法

このエレメントを使用して、リモート・アプリケーションの問題判別を行えます。このフィールドの値は以下のとおりです。

SQLM_PROT_UNKNOWN
クライアントは、不明のプロトコルを使用して通信を行っています。この値は、今後のクライアントが以前のレベルのサーバーに接続した場合にのみ戻されます。

SQLM_PROT_LOCAL
クライアントは、サーバーと同一のノードで実行されており、使用中の通信プロトコルはありません。

SQLM_PROT_TCPIP
TCP/IP

client_userid - クライアントのユーザー ID : モニター・エレメント

sqleseti API が使用された場合は、トランザクション・マネージャーが生成してサーバーに提供したクライアント・ユーザー ID。この接続、作業単位、またはアクティビティーの CLIENT_USERID 特殊レジスターの現行値。

注: このエレメントは、コーディネーター・メンバーでのみ報告されます。リモートのメンバーに報告される値は、長さ 0 のストリングです。

このモニター・エレメントは、**tpmon_client_userid** モニター・エレメントと同義です。 **client_userid** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_userid** モニター・エレメントは、表、ファイル、およびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 367. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE

表 367. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 368. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
ロッキング	-	-
作業単位	-	-
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

使用法

アプリケーション・サーバーまたはトランザクション処理モニター環境でこのエレメントを使用すると、トランザクションの実行対象になっているエンド・ユーザーを識別できます。

client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント

この接続で sqleseti API が発行された場合に、クライアントのシステムまたはワークステーション (CICS EITERMID など) を示します。この接続、作業単位、またはアクティビティーの CLIENT_WRKSTNNAME 特殊レジスターの現行値。

注: このエレメントは、コーディネーター・メンバーでのみ報告されます。リモートのメンバーに報告される値は、長さ 0 のストリングです。

このモニター・エレメントは、**tpmon_client_wkstn** モニター・エレメントと同義です。 **client_wrkstnname** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_wkstn** モニター・エレメントは、表、ファイル、およびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 369. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 369. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 370. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
ロッキング	-	-
作業単位	-	-
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

使用法

このエレメントを使用すると、ノード ID、端末 ID、またはその他の同様の ID を使用して、ユーザーのマシンを識別できます。

codepage_id アプリケーションで使用するコード・ページ ID

コード・ページ ID。

表 371. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本

表 372. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	常に収集される
接続	event_connheader	常に収集される

使用法 スナップショット・モニター・データの場合は、モニター対象のアプリケーションが開始されたパーティションでのコード・ページとなります。この

ID は、リモート・アプリケーションの問題判別に使用できます。この情報を使用すると、アプリケーションのコード・ページとデータベースのコード・ページ (DRDA ホスト・データベースの場合はホスト CCSID) の間でデータ変換がサポートされるように指定できます。サポート対象のコード・ページについては、*「管理ガイド」*を参照してください。

イベント・モニター・データの場合は、イベント・データを収集したデータベースのコード・ページとなります。このエレメントを使用すると、使用中のイベント・モニター・アプリケーションの実行に使用しているコード・ページとデータベースが使用しているコード・ページが異なるコード・ページかどうかを判別できます。イベント・モニターが書き込むデータには、データベースのコード・ページが使用されます。使用しているイベント・モニター・アプリケーションが別のコード・ページを使用する場合は、データを読み取るのに文字変換が必要になる場合があります。

comm_exit_wait_time - 通信バッファー出口待機時間のモニター・エレメント

通信バッファー出口ライブラリー API 関数からの戻りを待機するために費やした時間。値はミリ秒単位で示されます。

表 373. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	REQUEST METRICS BASE

表 374. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	常に収集される
統計	event_scstats (details_xml 文書に報告されます)。	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)。	REQUEST METRICS BASE

comm_exit_waits - 通信バッファー出口待機回数のモニター・エレメント

通信バッファー出口ライブラリー API 関数が呼び出された回数。

表 375. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	REQUEST METRICS BASE

表 376. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	常に収集される
統計	event_scstats (details_xml 文書に報告されます)。	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)。	REQUEST METRICS BASE

comm_private_mem - コミット済み専用メモリー

スナップショット時点で、データベース・マネージャーのインスタンスが現在コミットしている専用メモリーの量。返される comm_private_mem 値は、Windows オペレーティング・システムのみに関係します。

表 377. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

commit_sql_stmts - 試行されたコミット・ステートメント

試行された SQL COMMIT ステートメントの合計数。

表 378. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 379. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 モニター期間中にこのカウンターの変化量が少ない場合は、各アプリケーションのコミット頻度が少ないことを示し、ロギングとデータの並行性について問題となる場合があります。

このエレメントを使用して、作業単位の総数を計算することもできます。これには、以下の式を使用します。

```
  commit_sql_stmts
+ int_commits
+ rollback_sql_stmts
+ int_rollbacks
```

注: 計算した作業単位に含まれるのは、次の時点以降の作業単位だけです。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

この計算は、データベース・レベルとアプリケーション・レベルのいずれでも行えます。

comp_env_desc コンパイル環境 : モニター・エレメント

このエレメントは、SQL ステートメントをコンパイルする際に使用されるコンパイル環境に関する情報を保管します。

表 380. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 381. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	常に収集される
詳細付きデッドロック履歴	event_stmt_history	常に収集される
アクティビティー	event_activitystmt	常に収集される
パッケージ・キャッシュ	-	COLLECT BASE DATA

表 382. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このモニター・エレメントはコンパイル環境記述をバイナリー・ラージ・オブジェクトに保管します。 この情報を読み取り可能な形式で表示するには、COMPILATION_ENV 表関数を使用します。

このエレメントを COMPILATION_ENV 表関数への入力として、または SET COMPILATION ENVIRONMENT SQL ステートメントへの入力として提供することができます。

completion_status 完了状況 : モニター・エレメント

作業単位の状況。

表 383. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される

使用法

このエレメントを使用して、作業単位が終了した原因がデッドロックによるものか、または異常終了によるものかを判別できます。 sqllib/misc/DB2EvmonUOW.xsd ファイルに可能な値がリストされています。

- UNKNOWN
- COMMIT
- ROLLBACK
- GLOBAL_COMMIT
- GLOBAL_ROLLBACK
- XA_END
- XA_PREPARE

configured_cf_gbp_size - 構成済みクラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント

cf_gbp_sz 構成パラメーターを使用して指定された、割り振り済みの予約されたグループ・バッファー・プール・メモリー (ページ・サイズ 4 KB のページ単位)。

表 384. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

configured_cf_lock_size - 構成済みクラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント

構成されているグローバル・ロック・メモリー (ページ・サイズ 4 KB のページ単位)。この値は、**cf_lock_sz** 構成パラメーターで指定します。

表 385. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

configured_cf_sca_size - 構成済みクラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント

現在割り振られていて予約済みの共用通信域メモリー (ページ・サイズ 4 KB のページ単位)。この値は、**cf_sca_sz** 構成パラメーターで指定します。

表 386. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

configured_cf_mem_size - 構成済みクラスター・キャッシング・ファシリティーのメモリー・サイズ : モニター・エレメント

クラスター・キャッシング・ファシリティーの構成済み合計メモリー・サイズ (ページ・サイズ 4 KB のページ単位)。この値は、**cf_mem_sz** 構成パラメーターで指定します。

表 387. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

con_elapsed_time 最新の接続経過時間

このホスト・データベースから最後に切断された DCS アプリケーションが接続されていた経過時間。

エレメント ID

con_elapsed_time

エレメント・タイプ

time

表 388. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ

使用法

アプリケーションがホスト・データベースへの接続を維持していた時間の長さの標識として、このエレメントを使用します。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

con_local_dbases 現行接続を持つローカル・データベース

アプリケーションが接続されているローカル・データベースの数。

表 389. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 この数値は、データベース・レベルでデータを収集する際に、予想されるデータベース情報レコードの数を示します。

アプリケーションは、ローカルまたはリモートで実行中ですが、データベース・マネージャー内で作業単位を実行している場合としていない場合があります。

con_response_time 接続の最新応答時間

このデータベースに最後に接続された DCS アプリケーションにおける、接続処理の開始と実際の接続の確立との間の経過時間。

エレメント ID

con_response_time

エレメント・タイプ

time

表 390. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ

使用法

アプリケーションが特定のホスト・データベースに接続するために現在かかっている時間の標識として、このエレメントを使用します。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

concurrent_act_top 並行アクティビティーの最上位 : モニター・エレメント

最後にリセットされてからのサービス・サブクラスにおける並行アクティビティー (すべてのネスト・レベル) の最高水準点。

注: このエレメントは、すべてのアクティビティーの最大同時実行数をモニターします。これには、CONCURRENTDBCOORDACTIVITIES しきい値とは関係がないアクティビティーも含まれます。例えば、CALL ステートメントは、CONCURRENTDBCOORDACTIVITIES しきい値で制御される同時実行数には数えられませんが、現行アクティビティーの最高水準点の計測値には含まれます。

表 391. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE

表 392. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・サブクラス用のメンバーで到達したアクティビティー (ネストされたアクティビティーを含む) の並行性の最大数を調べることができます。

concurrent_connection_top 並行接続の最上位 : モニター・エレメント

最後にリセットされてからのこのサービス・クラスにおける並行コーディネーター接続の最高水準点。同じスーパークラスを持つすべてのサブクラスにおいて、このフィールドの値は同じです。

表 393. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE

表 394. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このエレメントは、現在の最高水準点がある場所を示すことにより、接続並行性のどの位置にしきい値を設定するかを判別する上で役立つ場合があります。さらに、そのようなしきい値が正しく構成され、作動しているかを検証する上でも役立ちます。

concurrent_wlo_act_top 並行 WLO アクティビティーの最上位 : モニター・エレメント

最後にリセットされてからの、このワークロードの任意のオカレンスにおける並行アクティビティー (すべてのネスト・レベル) の最高水準点。

表 395. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 396. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にこのワークロードの任意のオカレンス用のメンバーで到達した並行アクティビティーの最大数を調べることができます。

concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント

最後にリセットされてからのワークロードの並行オカレンスの最高水準点。

表 397. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 398. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-
統計	event_scstats	-

使用法

このエレメントを使用して、収集された時間間隔にワークロード用のメンバーで到達したワークロード・オカレンスの並行性の最大数を調べることができます。

concurrentdbcoordactivities_db_ threshold_id - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値 ID モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値の ID。

表 399. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES データベースしきい値がアクティビティーに適用されていた場合、どのデータベースしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_db_threshold _queued 並行データベース・コーディネーター・アクティビティーのデータベースしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES データベースしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 400. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES データベースしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_db_ threshold_value - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値の上限を戻します。

表 401. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES データベースしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_db_ threshold_violated - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値の違反モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES データベースしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 402. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_subclass_ threshold_id - 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 ID モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値の ID を戻します。

表 403. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値がアクティビティーに適用されていた場合、どのサービス・サブクラスしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_subclass_ threshold_queued 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 404. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_subclass_ threshold_value 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 : モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値の上限を戻します。

表 405. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_ACTIVITY_DETAILS 表関数 - 特定のアクティビティーに関する詳細情報を戻す	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_subclass_ threshold_violated 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 406. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_superclass_ threshold_id 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値 ID : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES_SUPERCLASS しきい値の ID。

表 407. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_superclass_ threshold_queued 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 408. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_superclass_ threshold_value 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値 : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値の上限。

表 409. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_superclass_ threshold_violated 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 410. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_wl_was_threshold _id - 並行データベース・コーディネーター・アクティビティーのワークロード作業アクション・セットしきい値 ID : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値の ID。

表 411. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_wl_was_threshold _queued - 並行データベース・コーディネーター・アクティビティーのワークロード作業アクション・セットしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 412. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_wl_was_threshold _value - 並行データベース・コーディネーター・アクティビティーのワークロード作業アクション・セットしきい値 : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値の上限。

表 413. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用された CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値の値を判別します。

concurrentdbcoordactivities_wl_was_threshold _violated - 並行データベース・コーディネーター・アクティビティーのワークロード作業アクション・セットしきい値違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 414. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_work_action_set_ threshold_id 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値 ID : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES 作業アクション・セットしきい値の ID。

表 415. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES 作業アクション・セットしきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_work_action_set_ threshold_queued 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 416. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_work_action_set_ threshold_value 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値 : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値の上限。

表 417. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES_WORK しきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_work_action_set_ threshold_violated 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 418. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値にアクティビティーが違反したかどうかを判別します。

conn_complete_time 接続要求完了タイム・スタンプ

接続要求が認可された日時。

エレメント ID

conn_complete_time

エレメント・タイプ

timestamp

表 419. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	タイム・スタンプ

使用法 このエレメントを使用すると、データベースへの接続要求が認可された日時を判別できます。

conn_time データベース接続時刻 : モニター・エレメント

データベースへの接続の日時 (データベース・レベルでは、これはデータベースへの最初の接続)、またはデータベースの活動化が発行された日時。

表 420. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される
データベース	event_dbheader	常に収集される
接続	event_connheader	常に収集される

使用法

このエレメントと **disconn_time** モニター・エレメントを組み合わせて使用すると、次の時点以降の経過時間を計算できます。

- データベースがアクティブだったとき (データベース・レベルの情報の場合)。
- 接続がアクティブだったとき (接続レベルの情報の場合)。

connection_start_time - 接続開始時刻 : モニター・エレメント

データベース・サーバーとの接続が確立された時刻。connection_time モニター・エレメントは、connection_start_time モニター・エレメントの別名です。

表 421. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 422. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	常に収集される

connection_status - 接続状況

スナップショット・モニターの場合、このモニター・エレメントは、 GET SNAPSHOT コマンドを発行するノードと db2nodes.cfg ファイルにリストされている他のノードの間の通信接続の状況を報告します。表関数モニターの場合、このモニター・エレメントは、FCM 接続状況を示すテキスト ID を報告します。

表 423. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続に関する詳細の取得	ACTIVITY METRICS BASE

表 424. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm_node	基本

使用法

スナップショット・モニターの場合の接続値は、以下のとおりです。

SQLM_FCM_CONNECT_INACTIVE
現在、接続されていません。

SQLM_FCM_CONNECT_ACTIVE
接続はアクティブです。

表関数モニターの場合の有効な値は、以下のとおりです。

Active 現在、接続されていません。

Inactive
接続はアクティブです。

2 つのメンバーがアクティブであっても、それらのメンバー間で何らかの通信が行われるまでは、それらの間の通信接続は非アクティブのままです。

connections_top 同時接続の最大数

データベースを活動化した時点からの、そのデータベースへの同時接続の最大数。

エレメント ID
connections_top

エレメント・タイプ
水準点

表 425. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 426. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントは、*maxappls* 構成パラメーターの設定値を評価するときに利用できます。

maxappls は、データベース接続数を制限するので、このエレメントの値が *maxappls* パラメーターと同じ場合は、一部のデータベース接続がリジェクトされていることを示します。

次の公式を使用すると、スナップショットを取った時点の接続数を計算できます。

```
rem_cons_in + local_cons
```

consistency_token パッケージ整合性トークン : モニター・エレメント

特定のパッケージ名および作成者について、複数のバージョンが存在する場合があります (DB2 バージョン 8 以降)。パッケージ整合性トークンを使用すると、現在実行中の SQL を含むパッケージのバージョンを識別できます。

表 427. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 428. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
ステートメント	event_stmt	常に収集される

使用法

このエレメントを、パッケージおよび実行中の SQL ステートメントの識別に利用できます。

container_accessible - コンテナーのアクセス可能性 : モニター・エレメント

このエレメントは、コンテナーがアクセス可能かどうかを示します。値 1 は「はい」を意味し、値 0 は「いいえ」を意味します。

表 429. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 430. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法 このエレメントを **container_id**、**container_name**、**container_type**、**container_total_pages**、**container_usable_pages**、および **container_stripe_set** の各エレメントと組み合わせて使用して、コンテナーを記述できます。

container_id - コンテナー ID : モニター・エレメント

表スペース内のコンテナーを一意的に定義する整数。

表 431. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 432. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法 このエレメントと container_name、 container_type、 container_total_pages、container_usable_pages、 container_stripe_set、および container_accessible の各エレメントを組み合わせて使用すると、コンテナーを記述できます。

container_name - コンテナー名 : モニター・エレメント

コンテナーの名前。

表 433. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 434. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法 このエレメントと container_id、 container_type、 container_total_pages、container_usable_pages、 container_stripe_set、および container_accessible の各エレメントを組み合わせて使用すると、コンテナーを記述できます。

container_stripe_set - コンテナー・ストライプ・セット : モニター・エレメント

コンテナーが属するストライプ・セット。

表 435. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 436. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法

このモニター・エレメントを **container_id**、**container_name**、**container_type**、**container_total_pages**、**container_usable_pages**、および **container_accessible** の各エレメントと組み合わせて使用して、コンテナーを記述します。これは、DMS 表スペースにのみ適用できます。

container_total_pages - コンテナー内の合計ページ数 : モニター・エレメント

コンテナーが占有するページ数の合計。

表 437. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 438. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法 このエレメントと container_id、 container_name、 container_type、 container_usable_pages、 container_stripe_set、および container_accessible の各エレメントを組み合わせて使用すると、コンテナーを記述できます。

container_type - コンテナー・タイプ : モニター・エレメント

コンテナーのタイプ。

表 439. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 440. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法

このエレメントはコンテナーのタイプを戻します。コンテナーのタイプは、ディレクトリー・パス (SMS の場合のみ)、ファイル (DMS の場合)、ロー・デバイス (DMS の場合) のいずれかです。このエレメントを、**container_id**、**container_name**、**container_total_pages**、**container_usable_pages**、**container_stripe_set**、および **container_accessible** の各エレメントと組み合わせて使用して、コンテナーを記述できます。

このモニター・エレメントの有効な値は `sqlutil.h` ファイルに定義されています。

container_usable_pages - コンテナー内の使用可能なページ数 : モニター・エレメント

コンテナー内の使用可能なページ数の合計。

表 441. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 442. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法 このエレメントと container_id、 container_name、 container_type、 container_total_pages、 container_stripe_set、および container_accessible の各エレメントを組み合わせて使用すると、コンテナーを記述できます。 SMS 表スペースの場合、この値は container_total_pages と同じになります。

coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント

最後にリセットしてからの、エラーで完了した任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

サービス・クラスでは、アクティビティーが異常終了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、このアクティビティーのカウントは、異常終了時のサブクラスでの合計にのみ含められます。

表 443. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 444. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、システム上のアクティビティーが正常に完了しているかを知ることができます。アクティビティーは、取り消し、エラー、または反作用しきい値のために打ち切られる場合があります。

coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント

最後にリセットしてからの、正常に完了した任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

サービス・クラスでは、アクティビティーが完了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、このアクティビティーのカウントは、完了時のサブクラスでの合計にのみ含められます。

表 445. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 446. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-
統計	event_scstats	-

使用法

このエレメントを使用すると、システムのアクティビティーのスループットを判別したり、複数のメンバー間の平均アクティビティー存続時間の計算を補助したりすることができます。

coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター DML アクティビティーの見積コストの算術平均。

内部的に追跡された平均がオーバーフローした場合、値 -2 が戻されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA EXTENDED 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。timeron 単位で測定されます。

サービス・クラスの場合、アクティビティーの見積コストは、アクティビティーがシステムに入るのに使用するサービス・サブクラスでしかカウントされません。REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの coord_act_est_cost_avg 平均は影響されません。

表 447. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される

表 447. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される

使用法

この統計を使用すると、最後の統計リセット以降に完了または異常終了したこのサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター DML アクティビティーの見積コストの算術平均を判別できます。

この平均を使用して、アクティビティー見積コストのヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー見積コストのヒストグラムからアクティビティーの平均見積コストを計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー見積コストのヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの実行時間の算術平均。

内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 単位はミリ秒です。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーがマップされたが完了しなかったサービス・サブクラスの coord_act_exec_time_avg 平均は影響されません。

表 448. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	COLLECT AGGREGATE ACTIVITY DATA

表 449. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーの実行時間の算術平均を判別できます。

この平均を使用して、アクティビティー実行時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー実行時間のヒストグラムから平均アクティビティー実行時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー実行時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの到着間隔の時間の算術平均。

内部的に追跡された平均がオーバーフローした場合、値 -2 が戻されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA EXTENDED 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。ミリ秒単位で測定されます。

サービス・クラスの場合、inter-arrival 時間の平均は、アクティビティーがシステムに入るのに使用するサービス・サブクラスから計算されます。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの coord_act_interarrival_time_avg は影響されません。

表 450. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、このサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの到着間隔の算術平均を判別できます。

到着間隔の時間を使用して、到着レートを判別できます。到着レートは到着間隔の時間の逆になります。この平均を使用して、アクティビティー到着間隔の時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー到着間隔の時間のヒストグラムから平均アクティビティー到着間隔の時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー到着間隔の時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの存続時間の算術平均。

内部的に追跡された平均がオーバーフローした場合、値 -2 が戻されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 ミリ秒単位で測定されます。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了した最後のサービス・クラスの coord_act_lifetime_avg 平均のみが影響されます。

表 451. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	COLLECT AGGREGATE ACTIVITY DATA

表 452. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーの存続時間の算術平均を判別できます。

この統計を使用して、アクティビティー存続時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー存続時間のヒストグラムから平均アクティビティー存続時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー存続時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント

coord_act_lifetime_top エレメントは、すべてのネスト・レベルでカウントされる、コーディネーター・アクティビティー存続期間の最高水準点です。この情報は、ミリ秒間単位の表記で格納されます。

サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップする際に、サービス・クラスでこの統計を効果的に使用するには、与えられたサービス・サブクラスの coord_act_lifetime_top 最高水準点と、同じ再マップのしきい値 (複数可) によって影響される他のサブクラスの最高水準点を集約する必要があります。 これは、アクティビティーが完了するのは、再マップしきい値によって別のサービス・サブクラスへ再マップされた後になるからであり、アクティビティーが再マップされるまで他のサービス・サブクラスにいた時間は、完了時のサブクラスでの合計にのみ含められるからです。

表 453. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	COLLECT AGGREGATE ACTIVITY DATA

表 454. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	-
統計	event_scstats	-
統計	event_wlstats	-

使用法

このエレメントを使用すると、アクティビティー存続時間のしきい値が有効であるかどうかを判別する助けになります。さらに、そのようなしきい値を構成する方法を判別する助けとすることもできます。

coord_agent_tid - コーディネーター・エージェントのエンジン・ディスパッチ可能単位 ID のモニター・エレメント

アプリケーションに関するコーディネーター・エージェントのエンジン・ディスパッチ可能単位 (EDU) ID。

表 455. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーのキュー時間の算術平均。

内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 ミリ秒単位で測定されます。

サービス・クラスの場合、キュー時間はアクティビティーが完了するもしくは異常終了したサービス・サブクラスでしかカウントされません。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーがマップされたが完了しなかったサービス・サブクラスの coord_act_queue_time_avg 平均は影響されません。

表 456. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	COLLECT AGGREGATE ACTIVITY DATA

表 457. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーのキュー時間の算術平均を判別できます。

この統計を使用して、アクティビティー・キュー時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー・キュー時間のヒストグラムから平均アクティビティー・キュー時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー・キュー時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント

coord_act_rejected_total は、最後にリセットしてからの、実行が許可されず、リジェクトされた任意のネスト・レベルのコーディネーター・アクティビティーの合計数を格納します。

このカウンターは、予測しきい値または実行阻止作業アクションのいずれかによりアクティビティーの実行が阻止された場合に更新されます。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

表 458. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 459. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-

使用法

このエレメントを使用すると、予測しきい値および実行を阻止する作業アクションが有効であるかどうか、および、それらの制限が大きすぎないかどうかを判別する助けになります。

coord_agent_pid コーディネーター・エージェント ID : モニター・エレメント

アプリケーションに関するコーディネーター・エージェントのエンジン・ディスパッチ可能単位 (EDU) ID。Linux オペレーティング・システムを除いて、EDU ID はスレッド ID にマップされます。Linux オペレーティング・システムでは、EDU ID は DB2 生成のユニーク ID です。

表 460. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本

表 461. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

使用法

このエレメントを使用すると、データベース・システム・モニター情報をシステム・トレースなどの他の診断情報のソースにリンクできます。

coord_agents_top - コーディネーター・エージェント最大数

同時に動作できるコーディネーター・エージェントの最大数。

表 462. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
データベース	dbase	基本

使用法

コーディネーター・エージェントの最大値がこのノードのワークロードとして大きすぎる場合は、**max_coordagents** 構成パラメーターを変更することで、この上限を下げることができます。

coord_member - コーディネーター・メンバー : モニター・エレメント

アプリケーションのコーディネーター・メンバー

表 463. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 464. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される
変更履歴	changesummary	常に収集される

coord_node コーディネーター・ノード

マルチノード・システムでは、インスタンスに接続つまりアタッチされたアプリケーションがあるノードのノード番号。

表 465. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 466. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法 接続されたアプリケーションは、それぞれ 1 つのコーディネーター・ノードにより処理されます。

coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント

作業単位またはアクティビティーのコーディネーター・パーティション。複数パーティションのシステムでは、コーディネーター・パーティションは、アプリケーションがデータベースに接続したパーティションです。

表 467. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 468. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
しきい値違反	event_thresholdviolations	常に収集される

使用法

このエレメントを使用して、コーディネーター以外のパーティションにレコードがあるアクティビティーまたは作業単位の、コーディネーター・パーティションを識別できます。

coord_stmt_exec_time - コーディネーター・エージェントのステートメントの実行時間 : モニター・エレメント

このメンバーのコーディネーター・エージェントがステートメントを実行するのにかかった時間の合計。値はミリ秒単位で示されます。

表 469. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 470. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

corr_token DRDA 相関トークン

DRDA AS 相関トークン。

表 471. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本

表 472. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_connheader	常に収集される

使用法 DRDA 相関トークンは、アプリケーション・サーバーとアプリケーション・リクエスターの間の処理の相関を求めるときに使用します。これはエラーが発生したときにログにダンプされる ID であるため、エラーとなった会話を識別するのに利用できます。場合によっては、会話の LUWID を示します。

通信に DRDA を使用していない場合は、このエレメントが *appl_id* を戻します (appl_id 参照)。

データベース・システム・モニター API を使用すると、このエレメントの長さを判別するために API 定数の SQLM_APPLID_SZ が使用されることに注意してください。

cost_estimate_top コスト見積もりの最上位 : モニター・エレメント

cost_estimate_top モニター・エレメントは、サービス・サブクラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの見積コストの最高水準点です。

サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、DML アクティビティーの見積コストは、アクティビティーがシステムに入るのに使用するサービス・サブクラスでしかカウントされません。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの cost_estimate_top は影響されません。

表 473. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーで到達した DML アクティビティー見積コストの最大値を判別することができます。

count - イベント・モニター・オーバーフロー数

連続して発生したオーバーフローの数。

エレメント ID
count

エレメント・タイプ
カウンター

表 474. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY _ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 475. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
オーバーフロー・レコード	event_overflow	常に収集される

使用法 このエレメントを使用すると、失われたモニター・データの量がわかります。

イベント・モニターは、一連のオーバーフローについて 1 つのオーバーフロー・レコードを送信します。

cpu_configured - 構成されている CPU 数のモニター・エレメント

オペレーティング・システムが認識している、このホスト上のプロセッサー数。

表 476. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_cores_per_socket - ソケットあたりの CPU コア数のモニター・エレメント

このホストでのプロセッサーの数。単一コア・システムでは、この値は 1 です。

表 477. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_hmt_degree - 論理 CPU 数のモニター・エレメント

ハードウェア・マルチスレッド化をサポートするシステムでは、マルチスレッド化の結果として存在しているように見える論理プロセッサーの数です。マルチスレッド化をサポートしていないシステムでは、この値は 1 です。

表 478. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_idle - プロセッサー・アイドル時間のモニター・エレメント

プロセッサー・アイドル時間 (プロセッサー・ティックで表されます)。Windows、AIX、および Linux システムの場合のみ報告されます。この測定値は、システム上のすべてのプロセッサーの総計です。

表 479. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

使用法

- この測定値は、システム上のすべてのプロセッサーの総計です。
- AIX の場合、このメトリックは、DB2 サーバーが稼働しているワークロード・パーティション (WPAR) およびロジカル・パーティション (LPAR) に関して報告されます。
- このモニター・エレメントを、関連するプロセッサー・タイマー・エレメントと一緒に使用すると、ホスト・システム上の特定の期間におけるプロセッサー使用率を計算できます。プロセッサー使用率をパーセンテージで計算するには、以下の手順を実行します。
  1. 期間の開始時点で ENV_GET_SYSTEM_RESOURCES 関数を使用して、以下のメトリックの値を取得します。
     - cpu_user_{t1} = **cpu_user**
     - cpu_system_{t1} = **cpu_system**
     - cpu_idle_{t1} = **cpu_idle**
     - cpu_wait_{t1} = **cpu_wait**
  2. プロセッサー使用率を計算する期間の終了時点で、上記の手順を再度実行し、同じメトリックのタイム・スタンプを判別します。
     - cpu_user_{t2} = **cpu_user**
     - cpu_system_{t2} = **cpu_system**
     - cpu_idle_{t2} = **cpu_idle**
     - cpu_iowait_{t2} = **cpu_iowait**
  3. 以下の数式を使用して、プロセッサー使用率を計算します。

$$100 \times \frac{(cpu\_system_{t2} - cpu\_system_{t1}) + (cpu\_user_{t2} - cpu\_user_{t1})}{(cpu\_system_{t2} - cpu\_system_{t1}) + (cpu\_user_{t2} - cpu\_user_{t1}) + (cpu\_idle_{t2} - cpu\_idle_{t1}) + (cpu\_iowait_{t2} - cpu\_iowait_{t1})}$$

cpu_iowait - 入出力待機時間モニター・エレメント

入出力の待機に費やした時間 (Linux、UNIX)。ハードウェア割り込みの受信および処理に費やした時間 (Windows)。プロセッサー・ティックで表されます。Windows、AIX、および Linux システムの場合のみ報告されます。この測定値は、システム上のすべてのプロセッサーの総計です。

表 480. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

使用法

- この測定値は、システム上のすべてのプロセッサーの総計です。
- AIX の場合、このメトリックは、DB2 サーバーが稼働しているワークロード・パーティション (WPAR) およびロジカル・パーティション (LPAR) に関して報告されます。
- このモニター・エレメントを、関連するプロセッサー・タイマー・エレメントと一緒に使用すると、ホスト・システム上の特定の期間におけるプロセッサー使用率を計算できます。プロセッサー使用率をパーセンテージで計算するには、以下の手順を実行します。
  1. 期間の開始時点で ENV_GET_SYSTEM_RESOURCES 関数を使用して、以下のメトリックの値を取得します。
     - cpu_user_{t1} = **cpu_user**
     - cpu_system_{t1} = **cpu_system**
     - cpu_idle_{t1} = **cpu_idle**
     - cpu_wait_{t1} = **cpu_wait**
  2. プロセッサー使用率を計算する期間の終了時点で、上記の手順を再度実行し、同じメトリックのタイム・スタンプを判別します。
     - cpu_user_{t2} = **cpu_user**
     - cpu_system_{t2} = **cpu_system**
     - cpu_idle_{t2} = **cpu_idle**
     - cpu_iowait_{t2} = **cpu_iowait**
  3. 以下の数式を使用して、プロセッサー使用率を計算します。

$$100 \times \frac{(cpu\_system_{t2} - cpu\_system_{t1}) + (cpu\_user_{t2} - cpu\_user_{t1})}{(cpu\_system_{t2} - cpu\_system_{t1}) + (cpu\_user_{t2} - cpu\_user_{t1}) + (cpu\_idle_{t2} - cpu\_idle_{t1}) + (cpu\_iowait_{t2} - cpu\_iowait_{t1})}$$

cpu_limit - WLM ディスパッチャーの CPU リミット : モニター・エレメント

サービス・クラスに関して構成されている WLM ディスパッチャーの CPU リミット。

表 481. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE

cpu_load_long - プロセッサー負荷 (長い時間フレーム) のモニター・エレメント

システム定義の長い期間におけるプロセッサー負荷。例えば、過去 10 分間または 15 分間におけるプロセッサーの平均負荷。Windows 以外のすべてのプラットフォームで報告されます。

表 482. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_load_medium - プロセッサー負荷 (中程度の時間フレーム) のモニター・エレメント

システム定義の中程度の期間におけるプロセッサー負荷。例えば、過去 5 分間または 10 分間におけるプロセッサーの平均負荷。Windows 以外のすべてのプラットフォームで報告されます。

表 483. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_load_short - プロセッサー負荷 (短い時間フレーム) のモニター・エレメント

システム定義の短い期間におけるプロセッサー負荷。例えば、過去 1 分間または 5 分間におけるプロセッサーの平均負荷。 Windows 以外のすべてのプラットフォームで報告されます。

表 484. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_online - オンライン CPU 数のモニター・エレメント

現在オンラインになっている、このホスト上のプロセッサー数。

表 485. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_share_type - WLM ディスパッチャー CPU シェア・タイプのモニター・エレメント

サービス・クラスに関して構成されている WLM ディスパッチャーの CPU シェアのタイプ。値は、soft および hard のどちらかです。

表 486. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE

cpu_shares - WLM ディスパッチャーの CPU 共有 : モニター・エレメント

サービス・クラスに関して構成されている WLM ディスパッチャーの CPU 共有数。

表 487. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE

cpu_speed - CPU クロック速度のモニター・エレメント

このホスト上のプロセッサーのクロック速度 (MHz 単位)。

表 488. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_system - カーネル時間 : モニター・エレメント

カーネル・コードの実行に費やした時間 (プロセッサー・ティックで表されます)。Windows、AIX、および Linux システムの場合のみ報告されます。この測定値は、システム上のすべてのプロセッサーの総計です。

表 489. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE
ENV_GET_DB2_SYSTEM_RESOURCES 表関数 - DB2(r) システム情報を戻す	ACTIVITY METRICS BASE

使用法

- この測定値は、システム上のすべてのプロセッサーの総計です。
- AIX の場合、このメトリックは、DB2 サーバーが稼働しているワークロード・パーティション (WPAR) およびロジカル・パーティション (LPAR) に関して報告されます。
- このモニター・エレメントを、関連するプロセッサー・タイマー・エレメントと一緒に使用すると、ホスト・システム上の特定の期間におけるプロセッサー使用率を計算できます。プロセッサー使用率をパーセンテージで計算するには、以下の手順を実行します。
  1. 期間の開始時点で ENV_GET_SYSTEM_RESOURCES 関数を使用して、以下のメトリックの値を取得します。
     - cpu_user_{t1} = **cpu_user**
     - cpu_system_{t1} = **cpu_system**
     - cpu_idle_{t1} = **cpu_idle**
     - cpu_wait_{t1} = **cpu_wait**
  2. プロセッサー使用率を計算する期間の終了時点で、上記の手順を再度実行し、同じメトリックのタイム・スタンプを判別します。
     - cpu_user_{t2} = **cpu_user**
     - cpu_system_{t2} = **cpu_system**
     - cpu_idle_{t2} = **cpu_idle**
     - cpu_iowait_{t2} = **cpu_iowait**
  3. 以下の数式を使用して、プロセッサー使用率を計算します。

$$100 \times \frac{(cpu\_system_{t2} - cpu\_system_{t1}) + (cpu\_user_{t2} - cpu\_user_{t1})}{(cpu\_system_{t2} - cpu\_system_{t1}) + (cpu\_user_{t2} - cpu\_user_{t1}) + (cpu\_idle_{t2} - cpu\_idle_{t1}) + (cpu\_iowait_{t2} - cpu\_iowait_{t1})}$$

cpu_timebase - 時間基準のレジスター増分の周波数のモニター・エレメント

時間基準のレジスター増分の周波数 (Hz 単位)。 Linux および PowerPC® システムの場合のみ

表 490. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_total - CPU 数モニター・エレメント

このホストでのプロセッサーの数。

表 491. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

使用法

このモニター・エレメントに報告される数は、オペレーティング環境によって異なるものを意味します。例えば、Windows システムから戻された場合、**cpu_total** は搭載されているプロセッサーの総数を指し、 AIX の場合は、構成されているプロセッサー数を表します。

cpu_usage_total - プロセッサー使用率のモニター・エレメント

カーネル処理時間を含む、このホスト上のプロセッサー全体の使用率。パーセンテージで表されます。 AIX、Linux、および Windows システムの場合のみ報告されます。

表 492. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

cpu_user - 非カーネル処理時間 : モニター・エレメント

ユーザー (非カーネル) のコードの実行に費やした時間 (プロセッサー・ティックで表されます)。 Windows、AIX、および Linux システムでのみ報告されます。 この測定値は、システム上のすべてのプロセッサーの総計です。

表 493. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE
ENV_GET_DB2_SYSTEM_RESOURCES 表関数 - DB2(r) システム情報を戻す	ACTIVITY METRICS BASE

使用法

- この測定値は、システム上のすべてのプロセッサーの総計です。
- AIX の場合、このメトリックは、DB2 サーバーが稼働しているワークロード・パーティション (WPAR) およびロジカル・パーティション (LPAR) に関して報告されます。
- このモニター・エレメントを、関連するプロセッサー・タイマー・エレメントと一緒に使用すると、ホスト・システム上の特定の期間におけるプロセッサー使用率を計算できます。プロセッサー使用率をパーセンテージで計算するには、以下の手順を実行します。
  1. 期間の開始時点で ENV_GET_SYSTEM_RESOURCES 関数を使用して、以下のメトリックの値を取得します。
     - cpu_user_{t1} = **cpu_user**
     - cpu_system_{t1} = **cpu_system**
     - cpu_idle_{t1} = **cpu_idle**
     - cpu_wait_{t1} = **cpu_wait**
  2. プロセッサー使用率を計算する期間の終了時点で、上記の手順を再度実行し、同じメトリックのタイム・スタンプを判別します。
     - cpu_user_{t2} = **cpu_user**
     - cpu_system_{t2} = **cpu_system**
     - cpu_idle_{t2} = **cpu_idle**
     - cpu_iowait_{t2} = **cpu_iowait**
  3. 以下の数式を使用して、プロセッサー使用率を計算します。

$$100 \times \frac{(cpu\_system_{t2} - cpu\_system_{t1}) + (cpu\_user_{t2} - cpu\_user_{t1})}{(cpu\_system_{t2} - cpu\_system_{t1}) + (cpu\_user_{t2} - cpu\_user_{t1}) + (cpu\_idle_{t2} - cpu\_idle_{t1}) + (cpu\_iowait_{t2} - cpu\_iowait_{t1})}$$

cpu_utilization - CPU 使用率 : モニター・エレメント

特定の論理パーティション上のサービス・クラスまたはワークロードが消費した合計 CPU 時間を、特定の期間にホストまたは LPAR で使用可能だった CPU 時間で除算した数値。

表 494. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 495. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このモニター・エレメントが WLM_GET_WORKLOAD_STATS 関数または WLM_GET_SERVICE_SUBCLASS_STATS 関数によって戻される場合、最後に統計を最後にリセットして以降の CPU 使用率を表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降の CPU 使用率を表します。

cpu_velocity - CPU 速度モニター・エレメント

CPU リソースの競合の度合いの測定。 0 から 1 までのスケールで表され、数値が小さいほど、CPU リソースの競合が大きいことを意味します。

CPU 速度は、サービス・クラス内の処理が CPU にアクセスした時間を、CPU にアクセスしたり、 CPU にアクセスするために待機したりした合計時間で除算して

算出します。この指標によって、作業が CPU をまったく待機しないで実行された場合の効率に比べ、どれほど効率的に実行されているかを測定することができます。数式は、次のとおりです。

```
cpu_velocity = total_cpu_time / (total_cpu_time + total_disp_run_queue_time)
```

cpu_velocity を収集するには、 **wlm_dispatcher** データベース・マネージャー構成パラメーターを ON に設定する必要があります。

表 496. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラス・メトリックのサンプリング	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロード・メトリックのサンプリング	REQUEST METRICS BASE

使用法

ディスパッチャーは、サービス・クラスまたはワークロードが、ある瞬間に付与可能量より多くの CPU リソースを要求する場合に、サービス・クラスまたはワークロードを優先順位付けする効果を持っています。このような場合、サービス・クラスまたはワークロード内で実行される処理は、 CPU リソースにアクセスするためのキューイングに時間を費やします。ディスパッチャーが、他のサービス・クラスまたはワークロードに付与する CPU リソースを減らして、そのようなサービス・クラスまたはワークロードに、より多くの CPU リソースを付与することができるのは、このようなときです。 CPU 速度が高い場合、現行のレベルの CPU 要求は既に満たされているため、ディスパッチャーは、このサービス・クラスに、応答時間またはスループットを向上させる影響をほとんど与えていないことを意味します。 CPU 速度が低い場合、ディスパッチャーは、現行レベルの CPU 要求のサービス・クラスまたはワークロードに、応答時間またはスループットを向上させる重大な影響を与えている可能性があることを意味します。

このエレメントを使用すると、サービス・クラスまたはワークロード内で実行される処理が、 CPU リソースを使用するためのキューイングに費やす時間の比率が、高いかどうかを調べることができます。サービス・クラスの CPU 速度が低い場合、高くするには、 CPU シェア数を増やしたり、低い CPU 速度を示すサービス・クラスに割り当てられている CPU リミットを大きくしたりして、CPU リソースの WLM ディスパッチャー制御を調整します。

cputime_threshold_id - CPU 時間しきい値 ID : モニター・エレメント

アクティビティーに適用されていた CPUTIME しきい値の ID。

表 497. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 CPUTIME しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

cputime_threshold_value - CPU 時間しきい値 : モニター・エレメント

アクティビティーに適用されていた CPUTIME しきい値の上限。

表 498. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CPUTIME しきい値がアクティビティーに適用されている場合、その値を判別します。

cputime_threshold_violated - CPU 時間しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CPUTIME しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 499. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CPUTIME しきい値にアクティビティーが違反したかどうかを判別します。

cputimeinsc_threshold_id - サービス・クラス内の CPU 時間しきい値 ID : モニター・エレメント

アクティビティーに適用されていた CPUTIMEINSC しきい値の ID。

表 500. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 CPUTIMEINSC しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

cputimeinsc_threshold_value - サービス・クラス内の CPU 時間しきい値 : モニター・エレメント

アクティビティーに適用されていた CPUTIMEINSC しきい値の上限。

表 501. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CPUTIMEINSC しきい値がアクティビティーに適用されている場合、その値を判別します。

cputimeinsc_threshold_violated - サービス・クラス内の CPU 時間しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CPUTIMEINSC しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 502. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CPUTIMEINSC しきい値にアクティビティーが違反したかどうかを判別します。

create_nickname ニックネーム作成回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソース上にあるオブジェクトのニックネームを作成した合計回数が含まれています。

このモニターは最新の値を格納します。

表 503. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このフェデレーテッド・サーバー・インスタンスまたはアプリケーションによるこのデータ・ソースへの CREATE NICKNAME アクティビティーの量を判別できます。 CREATE NICKNAME 処理を行うと、データ・ソース・カタログに対して複数の照会が実行されるので、このエレメントの値が大きい場合は、原因を突き止めるか、またはアクティビティーを制限する必要があります。

create_nickname_time ニックネーム作成応答時間

このエレメントには、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの CREATE NICKNAME ステートメントを、このデータ・ソースが処理するのに要した合計時間が含まれています (ミリ秒単位)。

この応答時間は、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降から測定されます。応答時間は、フェデレーテッド・サーバーが CREATE NICKNAME ステートメントを処理するためにデータ・ソースから情報の検索を開始してから必要なデータの取り出しがすべて完了するまでの時間です。

表 504. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、このデータ・ソースでニックネームを作成するのに要した実際の時間を判別できます。

creator アプリケーション作成者

アプリケーションをプリコンパイルしたユーザーの許可 ID。

表 505. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 506. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-

使用法 このエレメントとカタログ内のパッケージ・セクション情報の CREATOR 列を組み合わせて使用し、処理中の SQL ステートメントを識別してください。

CURRENT PACKAGE PATH 特殊レジスターを設定した場合には、 SQL ステートメントの存続期間中に *creator* 値はさまざまな値を反映します。 PACKAGE PATH の解決の前にスナップショットまたはイベント・モニター・レコードが取られる場合は、 *creator* 値はクライアント要求から流れる値を反映します。 PACKAGE PATH の解決の後にスナップショットまたはイベント・モニター・レコードが取られる場合は、 *creator* 値は解決されたパッケージの作成者を反映します。解決されたパッケージの *creator* 値は CURRENT PACKAGE PATH SPECIAL REGISTER 中に最初に表示される値で、パッケージ名およびユニーク ID はクライアント要求の名前および ID と一致します。

current_cf_gbp_size - 現行クラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント

クラスター・キャッシング・ファシリティーで現在使用されているグループ・バッファー・プール・メモリー (ページ・サイズ 4 KB のページ単位)。

表 507. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

current_cf_lock_size - 現行クラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント

現在使用されているグローバル・ロック・メモリー (ページ・サイズ 4 KB のページ単位)。

表 508. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

current_cf_sca_size - 現行クラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント

現在使用されている共用通信域メモリー (ページ・サイズ 4 KB のページ単位)。

表 509. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

current_cf_mem_size - 現行クラスター・キャッシング・ファシリティーのメモリー・サイズ :モニター・エレメント

現在使用されている合計メモリー (ページ・サイズ 4 KB のページ単位)。

表 510. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

current_active_log 現行アクティブ・ログ・ファイル番号

現在 DB2 データベース・システムが書き込んでいるアクティブ・ログ・ファイルのファイル番号。

表 511. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 512. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	detail_log	基本

表 513. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *first_active_log* および *last_active_log* エレメントを組み合わせて使用すると、アクティブ・ログ・ファイルの範囲を判別できます。アクティブ・ログ・ファイルの範囲を知っていると、ログ・ファイルに必要なディスク・スペースを判別するのに役立ちます。

また、このエレメントを使用すると、どのログ・ファイルにデータがあるか判別でき、スプリット・ミラー・サポートが必要なログ・ファイルを識別するのに役立ちます。

current_archive_log 現行アーカイブ・ログ・ファイル番号

現在 DB2 データベース・システムがアーカイブしているログ・ファイルのファイル番号。 DB2 データベース・システムがログ・ファイルをアーカイブしていない場合は、このエレメントの値は SQLM_LOGFILE_NUM_UNKNOWN になります。

表 514. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 515. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	detail_log	基本

表 516. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントを使用して、ログ・ファイルのアーカイブに問題があるかどうかを判別します。この種の問題には、以下のものがあります。

- 低速のアーカイブ・メディア
- 使用不可であるアーカイブ・メディア

current_extent - 現在移動中のエクステント : モニター・エレメント

表スペースのリバランス処理によって現在移動中であるエクステントの数値 ID。

表 517. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE

current_request - 現在の操作要求のモニター・エレメント

エージェントが現在処理している操作、または最後に処理した操作。

表 518. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

cursor_name カーソル名

この SQL ステートメントに対応するカーソルの名前。

エレメント ID

cursor_name

エレメント・タイプ

情報

表 519. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 520. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック	event_detailed_dlconn	常に収集される
ステートメント	event_stmt	常に収集される

使用法 このエレメントを使用すると、処理中の SQL ステートメントを識別できます。この名前は、SQL SELECT ステートメントの OPEN、 FETCH、CLOSE、および PREPARE に使用されます。カーソルを使用しない場合は、このフィールドはブランクになります。

data_object_pages データ・オブジェクト・ページ数

表が使用するディスク・ページの数。このサイズは、基本表のサイズのみを表します。索引オブジェクトの消費スペースは *index_object_pages*、LOB データは *lob_object_pages*、および LONG データは *long_object_pages* で報告されます。

表 521. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 522. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法 このエレメントを使用すると、特定の表が使用する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに表が大きくなる比率を追跡できます。

data_object_l_pages - 表データ論理ページ : モニター・エレメント

この表に含まれているデータによって使用された、ディスク上の論理ページの数。

表 523. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

- この値は、オブジェクトに割り振られているスペース量より小さい場合があります。そのようになる可能性があるのは、REORG TABLE コマンドで RECLAIM EXTENTS ONLY オプションを使用した場合です。この場合、MON_GET_TABLE によって返される論理ページ数には、再利用されたエクステントが含まれています。
- この値は、オブジェクトに物理的に割り振られているスペース量より小さい場合があります。そのようになる可能性があるのは、TRUNCATE ステートメントの REUSE STORAGE オプションを使用した場合です。このオプションを指定すると、表に割り振られているストレージは引き続き割り振られますが、ストレージは空と見なされます。さらに、このモニター・エレメントの値は、オブジェクトに論理的に割り振られているスペース量よりも小さくなる可能性があります。その理由は、論理的に割り振られているスペースの合計には、量的にはわずかですが追加のメタデータが含まれているからです。

  オブジェクトの正確な論理サイズまたは物理サイズを取得するには、ADMIN_GET_TAB_INFO_V97 関数を使用します。この関数を使用すると、このモニター・エレメントに関して報告されるページ数とページ・サイズの積で得られる値よりも正確なオブジェクト・サイズに関する情報が得られます。

data_partition_id - データ・パーティション ID : モニター・エレメント

情報が戻されるデータ・パーティションの ID。

表 524. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMINTABINFO 管理ビューおよび ADMIN_GET_TAB_INFO 表関数 - 表のサイズおよび状態に関する情報の検索	ACTIVITY METRICS BASE
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

表 525. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
ロック	lock	ロック
ロック	lock_wait	ロック

表 526. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-
デッドロック	lock	-

使用法

このエレメントは、パーティション表およびパーティション索引にのみ適用されます。その他の場合、このモニター・エレメントの値は NULL です。

ロック・レベル情報が戻されるとき、-1 という値は、表全体へのアクセスを制御するロックを表します。

datasource_name データ・ソース名

このエレメントには、フェデレーテッド・サーバーによってリモート・アクセス情報が表示されているデータ・ソースの名前が含まれています。このエレメントは、SYSCAT.SERVERS の 'SERVER' 列に対応しています。

エレメント ID

datasource_name

エレメント・タイプ

情報

表 527. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

使用法 このエレメントを使用すると、アクセス情報が収集されて戻されているデータ・ソースを識別できます。

datataginsc_threshold_id - サービス・クラスしきい値 (IN 条件) ID のデータ・タグ

アクティビティーに適用されていた DATATAGINSC IN しきい値の ID。

表 528. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

使用上の注意

- このエレメントを使用して、DATATAGINSC IN しきい値がアクティビティーに適用されていた場合、その値を判別します。

datataginsc_threshold_value - サービス・クラスしきい値 (IN 条件) の値のデータ・タグ

アクティビティーに適用されていた DATATAGINSC IN しきい値のデータ・タグのコンマ区切りリスト。

表 529. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

使用上の注意

- このエレメントを使用して、DATATAGINSC IN しきい値がアクティビティーに適用されていた場合、その値を判別します。

datataginsc_threshold_violated - 違反したサービス・クラスしきい値 (IN 条件) のデータ・タグ

アクティビティーが DATATAGINSC IN しきい値に違反したかどうかを示します。アクティビティーが DATATAGINSC IN しきい値に違反した場合には、1 が戻ります。アクティビティーがしきい値に違反しなかった場合には 0 を戻します。

表 530. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

使用上の注意

- このエレメントを使用して、アクティビティーに適用されていた DATATAGINSC IN しきい値にアクティビティーが違反したかどうかを判別します。

datatagnotinsc_threshold_id - サービス・クラスしきい値 (NOT IN 条件) ID のデータ・タグ

アクティビティーに適用されていた DATATAGINSC NOT IN しきい値の ID。

表 531. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

使用上の注意

- このエレメントを使用して、DATATAGINSC NOT IN しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

datatagnotinsc_threshold_value - サービス・クラスしきい値 (NOT IN 条件) の値のデータ・タグ

アクティビティーに適用されていた DATATAGINSC NOT IN しきい値のデータ・タグのコンマ区切りリスト。

表 532. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

使用上の注意

- このエレメントを使用して、DATATAGINSC NOT IN しきい値がアクティビティーに適用されていた場合、その値を判別します。

datatagnotinsc_threshold_violated - 違反したサービス・クラスしきい値 (NOT IN 条件) のデータ・タグ

アクティビティーが DATATAGINSC NOT IN しきい値に違反したかどうかを示します。アクティビティーが DATATAGINSC NOT IN しきい値に違反した場合には、1 が戻ります。アクティビティーがしきい値に違反しなかった場合には、0 が戻ります。

表 533. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

使用上の注意

- このエレメントを使用して、アクティビティーに適用されていた DATATAGINSC NOT IN しきい値にアクティビティーが違反したかどうかを判別します。

db2_process_id - DB2 プロセス ID のモニター・エレメント

報告が行われるメンバー上で実行されている DB2 プロセスの数値 ID。

表 534. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_DB2_SYSTEM_RESOURCES 表関数 - DB2(r) システム情報を戻す	ACTIVITY METRICS BASE

db2_process_name - DB2 プロセス名のモニター・エレメント

報告が行われるメンバー上で実行されている DB2 プロセスの名前。

表 535. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_DB2_SYSTEM_RESOURCES 表関数 - DB2(r) システム情報を戻す	ACTIVITY METRICS BASE

db2_status DB2 インスタンス状況 : モニター・エレメント

データベース・マネージャーのインスタンスの現在の状況。

表 536. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントを使用して、データベース・マネージャー・インスタンスの状態を判別できます。

このエレメントの値は以下のとおりです。

API 定数	値	説明
SQLM_DB2_ACTIVE	0	データベース・マネージャー・インスタンスはアクティブになっている。
SQLM_DB2_QUIESCE_PEND	1	インスタンス内のインスタンスおよびデータベースは静止ペンディング状態となっている。インスタンス・データベースへの新規接続は許可されず、新規作業単位を開始することはできません。静止要求によっては、アクティブな作業単位の完了が許可される場合と即時ロールバックが行われる場合があります。

API 定数	値	説明
SQLM_DB2_QUIESCED	2	インスタンス内のインスタンスおよびデータベースは静止状態となっている。インスタンス・データベースへの新規接続は許可されず、新規作業単位を開始することはできません。

db2start_time - データベース・マネージャー開始タイム・スタンプ

db2start コマンドを使用してデータベース・マネージャーを開始した日時。

エレメント ID

db2start_time

エレメント・タイプ

timestamp

表 537. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

表 538. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	evmonstart	常に収集される

使用法

このエレメントと *time_stamp* モニター・エレメントを組み合わせて使用すると、データベース・マネージャーを開始してからスナップショットが取られるまでの経過時間を計算できます。

変更履歴イベント・モニターでは、このエレメントを使用すると、据え置かれたデータベース・マネージャー構成パラメーターの更新が、いつ有効になったかを追跡することができます。

db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント

データベースへの接続の日時 (データベース・レベルでは、これはデータベースへの最初の接続)、またはデータベースの活動化が発行された日時。

表 539. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	タイム・スタンプ
表スペース	tablespace_list	バッファー・プール、タイム・スタンプ
表	table_list	タイム・スタンプ

表 540. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
変更履歴	evmonstart	常に収集される

使用法

このエレメントと **disconn_time** モニター・エレメントを組み合わせて使用すると、合計接続時間を計算できます。

変更履歴イベント・モニターでは、このエレメントを使用すると、据え置かれたデータベース構成パラメーターの更新が、いつ有効になったかを追跡することができます。

db_heap_top 割り振られた最大データベース・ヒープ

このエレメントは、DB2 のバージョン間での互換性を確保するために維持されています。現在は、メモリーの使用量を計算しますが、データベース・ヒープの使用量だけが対象ではありません。

注: db_heap_top モニター・エレメントは、DB2 バージョン 9.5 以降では非推奨になっています。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 541. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 542. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

db_location データベース・ロケーション

アプリケーションに関連したデータベースのロケーション。

表 543. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 スナップショットをとるアプリケーションに対応する、データベース・サーバーの相対的なロケーションを判別します。次の値があります。

- SQLM_LOCAL
- SQLM_REMOTE

db_name データベース名 : モニター・エレメント

情報が収集されるデータベースの実名、またはアプリケーションの接続先のデータベースの実名。これはデータベースの作成時に与えられた名前です。

表 544. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_AUTO_MAINT_QUEUE 表関数 - 自動保守ジョブの情報の取得	ACTIVITY METRICS BASE
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	ACTIVITY METRICS BASE

表 545. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl_id_info	基本
アプリケーション	appl_remote	基本
表スペース	tablespace_list	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
表	table_list	表
ロック	db_lock_list	基本
動的 SQL	dynsql_list	基本
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl_info	基本

表 546. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbheader	常に収集される

使用法

このエレメントを使用すると、データが適用される特定のデータベースを識別できます。

ホストへの接続、または System i® のデータベース・サーバーへの接続で DB2 Connect を使用しないアプリケーションの場合は、このエレメントと **db_path** モニ

ター・エレメントを組み合わせて使用すると、データベースを個別に識別し、モニターが提供する情報の各レベルに関連付けることができます。

db_path データベース・パス

モニター対象のシステムに保管されているデータベースのロケーションを示す絶対パスです。

表 547. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl_id_info	基本
表スペース	tablespace_list	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
表	table_list	表
ロック	db_lock_list	基本
動的 SQL	dynsql_list	基本

表 548. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbheader	常に収集される

使用法 このエレメントと *db_name* モニター・エレメントを組み合わせて使用すると、データが適用される特定のデータベースを識別できます。

db_status データベース状況 : モニター・エレメント

データベースの現在の状況。

表 549. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法

このエレメントを使用して、データベースの状態を判別できます。

このフィールドの値は以下のとおりです。

API 定数	値	説明
SQLM_DB_ACTIVE	0	データベースはアクティブになっている。

API 定数	値	説明
SQLM_DB_QUIESCE_PEND	1	データベースは静止ペンディング状態となっている。データベースに対する新しい接続は許可されません。新しい作業単位も開始できません。静止要求によっては、アクティブな作業単位の完了が許可される場合と即時ロールバックが行われる場合があります。
SQLM_DB_QUIESCED	2	データベースは静止状態となっている。データベースに対する新しい接続は許可**されません**。新しい作業単位も開始できません。
SQLM_DB_ROLLFWD	3	データベースでロールフォワードが進行中。
SQLM_DB_ACTIVE_STANDBY	4	データベースは、読み取り可能な HADR スタンバイ・データベースです。
SQLM_DB_STANDBY	5	データベースは、HADR スタンバイ・データベースです。

db_storage_path 自動ストレージ・パス : モニター・エレメント

このエレメントは、自動ストレージ表スペースを配置するためにデータベースによって使用されるロケーションの絶対パスを示します。データベースに関連したストレージ・パスは 0 個以上あります。

表 550. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE

表 551. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	基本

使用法

このエレメントを **num_db_storage_paths** モニター・エレメントと一緒に使用して、このデータベースに関連したストレージ・パスを識別します。

db_storage_path_id - ストレージ・パス ID

ストレージ・グループにおけるストレージ・パスのそれぞれのオカレンスに対するユニーク ID。

表 552. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

db_storage_path_state - ストレージ・パスの状態 : モニター・エレメント

自動ストレージ・パスの状態は、ストレージ・パスがデータベースによって使用されているかどうかを示します。

表 553. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE

表 554. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	基本

使用法

このモニター・エレメントを使用して、ストレージ・パスがデータベースによって使用されているかどうか判別します。次の値が使用されます。

NOT_IN_USE
指定されたデータベース・パーティションに、このストレージ・パスを使用している表スペースはありません。

IN_USE
指定されたデータベース・パーティションに、このストレージ・パスを使用している表スペースがあります。

DROP_PENDING
このストレージ・パスはドロップされましたが、まだ使用している表スペースがあります。ストレージ・パスがデータベースから物理的にドロップされる前に、すべての表スペースはその使用を停止しなければなりません。ドロップされたストレージ・パスの使用を停止するには、表スペースをドロップするか、または ALTER TABLESPACE ステートメントの REBALANCE 節を使用して表スペースのリバランスを行います。

db_storage_path_with_dpe - データベース・パーティション式を含むストレージ・パス : モニター・エレメント

未評価のデータベース・パーティション式を含む自動ストレージ・パス。

表 555. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE

表 556. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	基本

使用法

ストレージ・パスにデータベース・パーティション式が含まれる場合、このモニター・エレメントを使用して、CREATE DATABASE コマンドまたは ALTER DATABASE ステートメントの一部としてデータベースに対して指定されたストレージ・パスを判別します。

ストレージ・パスにデータベース・パーティション式が含まれない場合は、このモニター・エレメントは NULL 値を戻します。

db_work_action_set_id データベース作業アクション・セット ID : モニター・エレメント

このアクティビティーがデータベース有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスが所属する作業クラス・セットに関連した作業アクション・セットの ID を示します。それ以外の場合、このモニター・エレメントは 0 の値を示します。

表 557. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

表 558. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントと **db_work_class_id** エレメントを組み合わせて使用すると、アクティビティーのデータベース作業クラスが存在する場合にはそれを一意的に識別できます。

db_work_class_id データベース作業クラス ID : モニター・エレメント

このアクティビティーがデータベース有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 559. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 560. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントと **db_work_action_set_id** エレメントを組み合わせて使用すると、アクティビティーのデータベース作業クラスが存在する場合にはそれを一意的に識別できます。

dbpartitionnum - データベース・パーティション番号 : モニター・エレメント

パーティション・データベース環境の場合、これは、データベース・メンバーの数値 ID です。DB2 Enterprise Server Edition および DB2 pureScale 環境の場合には、この値は 0 です。

表 561. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_MSGS 表関数 - ADMIN_CMD プロシージャーを通して実行するデータ移動ユーティリティーによって生成されたメッセージの検索	ACTIVITY METRICS BASE
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE

表 561. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
ADMIN_GET_TAB_COMPRESS_INFO 表関数 - 圧縮節約量の見積もり	ACTIVITY METRICS BASE
ADMIN_GET_TAB_DICTIONARY_INFO 表関数 - 既存の表ディクショナリーのプロパティーの報告	ACTIVITY METRICS BASE
ADMINTABINFO 管理ビューおよび ADMIN_GET_TAB_INFO 表関数 - 表のサイズおよび状態に関する情報の検索	ACTIVITY METRICS BASE
AUDIT_ARCHIVE プロシージャーおよび表関数 - 監査ログ・ファイルのアーカイブ	ACTIVITY METRICS BASE
DBCFG 管理ビューおよび DB_GET_CFG 表関数 - データベース構成パラメーター情報の取得	ACTIVITY METRICS BASE
DBPATHS 管理ビューおよび ADMIN_LIST_DB_PATHS 表関数 - データベース・パスの取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE
PDLOGMSGS_LAST24HOURS 管理ビューおよび PD_GET_LOG_MSGS 表関数 - 問題診断メッセージの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE

表 561. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

使用法

DB2 pureScale 環境では、複数のメンバーが単一のパーティションで作動します。そのような構成で実行される場合、表スペース内のフリー・ページの番号などのストレージに関する物理属性は、システム内のすべてのメンバーで重複します。それぞれのメンバーが、システムの正確なサイズ合計をレポートします。複数パーティション構成の場合、システム全体の値を把握するために各パーティションからの値がユーザーによって相互に関連付けられる必要があります。

dbpartitionnum モニター・エレメントは **data_partition_id** モニター・エレメントとは異なります。後者のモニター・エレメントは、値に基づいて、表にあるデータを細分化して作成されたデータ・パーティションを識別するのに使用されます。

dcs_appl_status DCS アプリケーション状況 : モニター・エレメント

DB2 Connect ゲートウェイでの DCS アプリケーションの状況。

表 562. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法

このエレメントは、DCS アプリケーションに関する問題判別に使用します。次の値があります。

- SQLM_DCS_CONNECTPEND_OUTBOUND

  アプリケーションが DB2 Connect ゲートウェイからホスト・データベースへのデータベース接続を開始しましたが、要求はまだ完了していません。

- SQLM_DCS_UOWWAIT_OUTBOUND

  DB2 Connect ゲートウェイは、ホスト・データベースがアプリケーションの要求に応答するのを待っています。

- SQLM_DCS_UOWWAIT_INBOUND

  DB2 Connect ゲートウェイからホスト・データベースへの接続が確立され、ゲートウェイがアプリケーションの SQL 要求を待っています。あるいは、アプリケーション内の作業単位に代わって、DB2 Connect ゲートウェイが待機しています。通常、これはアプリケーションのコードが実行中であることを意味します。

dcs_db_name DCS データベース名

DCS ディレクトリーにカタログされている DCS データベースの名前。

表 563. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

ddl_classification - DDL 種別

実行された DDL のタイプを示す種別。

表 564. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DDLSTMTEXEC	常に収集される

使用法

DDL の分類は以下のとおりです。

DDLSTORAGE
データベース、バッファー・プール、パーティション・グループ、ストレージ・グループ、および表スペースの変更 DDL の実行。

DDLWLM
ヒストグラム、サービス・クラス、しきい値、作業アクション・セット、作業クラス・セット、およびワークロードの DDL の実行。

DDLMONITOR
イベント・モニターおよび使用リストの DDL の実行。

DDLSECURITY
監査ポリシー、権限付与、マスク、権限ロール、権限取り消し、セキュリティー・ラベル、セキュリティー・ラベル・コンポーネント、セキュリティー・ポリシー、およびトラステッド・コンテキストの DDL 実行。

DDLSQL
別名、関数、メソッド、モジュール、パッケージ、プロシージャー、スキーマ、シノニム、トランスフォーム、トリガー、タイプ、変数、およびビューの DDL の実行。

DDLDATA
索引、シーケンス、表、および一時表の DDL の実行。

DDLXML
XSROBJECT の DDL の実行。

DDLFEDERATED
ニックネーム/サーバー、タイプ/ユーザー・マッピング、およびラッパーの DDL の実行。

ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント

このエレメントは、実行された SQL データ定義言語 (DDL) ステートメントの数を示します。

表 565. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 566. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、アプリケーション・レベルまたはデータベース・レベルのデータベース・アクティビティーのレベルを判別できます。DDL ステートメントは、システム・カタログ表への影響のために実行にコストがかかります。そのため、このエレメントの値が大きい場合は、その原因を突き止めて、このアクティビティーが実行されないように制約すべきです。

このエレメントを使用すると、次の公式を使用して、 DDL アクティビティーのパーセンテージも計算できます。

```
ddl_sql_stmts / total number of statements
```

この情報は、アプリケーションのアクティビティーおよびスループットの分析に役立ちます。 DDL ステートメントも次の項目に影響を与えます。

- カタログ・キャッシュ。保管されている表記述子情報と許可情報が無効になるので、システム・カタログから情報を取り出すためのシステム使用量が増加します。
- パッケージ・キャッシュ。保管されているセクションが無効になるので、セクションの再コンパイルのための処理時間が増加します。

DDL ステートメントの例としては、CREATE TABLE、CREATE VIEW、ALTER TABLE、および DROP INDEX があります。

deadlock_id デッドロック・イベント ID

デッドロックのデッドロック ID。

エレメント ID
deadlock_id

エレメント・タイプ
情報

表 567. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	常に収集される
デッドロック	event_dlconn	常に収集される
詳細付きデッドロック	event_detailed_dlconn	常に収集される
詳細付きデッドロック履歴	event_detailed_dlconn	常に収集される
詳細付きデッドロック履歴	event_stmt_history	常に収集される
詳細付きデッドロック履歴値	event_data_value	常に収集される
詳細付きデッドロック履歴値	event_detailed_dlconn	常に収集される
詳細付きデッドロック履歴値	event_stmt_history	常に収集される

使用法 モニター・アプリケーションでこのエレメントを使用すると、デッドロック接続およびステートメント履歴イベント・レコードと、デッドロック・イベント・レコードとを関連付けることができます。

deadlock_member - デッドロック・メンバー : モニター・エレメント

参加者がロックを要求しているメンバー。

表 568. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

deadlock_node デッドロック発生場所のパーティション番号

デッドロックが発生した場所のパーティション番号。

エレメント ID
deadlock_node

エレメント・タイプ
情報

表 569. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	常に収集される
デッドロック	event_dlconn	常に収集される
詳細付きデッドロック	event_detailed_dlconn	常に収集される

使用法 このエレメントが該当するのは、パーティション・データベースだけです。モニター・アプリケーションでこのエレメントを使用すると、デッドロック接続イベント・レコードとデッドロック・イベント・レコードを関連付けることができます。

deadlock_type - デッドロック・タイプのモニター・エレメント

発生したデッドロックのタイプ。値は LOCAL または GLOBAL のいずれかです。ローカル・デッドロックでは、すべての参加者が同じメンバーで実行します。グローバル・デッドロックでは、少なくとも 1 つのデッドロック参加者がリモート・メンバー上で実行されています。

表 570. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock	

deadlocks - デッドロック検出数 : モニター・エレメント

発生したデッドロックの合計数。

表 571. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 571. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 572. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	ロック

スナップショット・モニターの場合、このカウンターはリセットできます。

表 573. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントは、アプリケーション間で競合の問題が起きていることを示す場合があります。問題の原因としては、次の状態が考えられます。

- データベースでロック・エスカレーションが発生している場合。
- システムが生成した行のロッキング数が十分なときに、アプリケーションが表を明示的にロッキングしている場合。

- アプリケーションがバインディングのときに不適切な分離レベルを使用している場合。
- カタログ表が反復可能読み取りのためにロックされている場合。
- 複数のアプリケーションが同じロックを異なる順序で獲得しているために、デッドロックになっている場合。

この問題は、デッドロックが発生しているアプリケーション (またはアプリケーション処理) が判別できれば解決できます。この場合、アプリケーションが並行して実行できるようにアプリケーションを変更できます。ただし、一部のアプリケーションでは並行して実行できない場合があります。

接続タイム・スタンプ・モニター・エレメント (**last_reset**、**db_conn_time**、および **appl_con_time**) を使用すると、デッドロックの重大度を判別できます。例えば、デッドロックが 5 時間に 10 回起こるよりも、5 分間に 10 回起こるほうが重大です。

上記の関連エレメントについての記述箇所にも、調整に関するその他の推奨事項を示している場合があります。

deferred - 据え置き

構成パラメーター値の変更が据え置かれたかどうかを示します。

表 574. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DBDBMCFG	常に収集される

使用法

変更履歴イベント・モニターで収集されるこの値は、以下のとおりです。

Y 変更は次のデータベース活動化まで据え置かれます。

N 変更は即時に有効になります

degree_parallelism 並列処理の度合い

照会がバインドされたときに要求された並列処理の度合い。

エレメント ID
degree_parallelism

エレメント・タイプ
情報

表 575. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

使用法 agents_top と組み合わせて使用すると、照会で最大レベルの並列処理ができたかどうかを判別できます。

del_keys_cleaned - クリーンアップされた疑似削除キー : モニター・エレメント

クリーンアップされた疑似削除キーの数。

表 576. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

delete_sql_stmts 削除回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースに DELETE ステートメントを発行した合計回数が含まれています。

表 577. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、フェデレーテッド・サーバーまたはアプリケーションによりこのデータ・ソースに対して行われたデータベース・アクティビティーのレベルを判別できます。

このエレメントを使用すると、次の公式を使用して、フェデレーテッド・サーバーまたはアプリケーションによるこのデータ・ソースへの書き込みアクティビティーのパーセンテージも判別できます。

```
書き込みアクティビティー =
  (INSERT ステートメント + UPDATE ステートメント + DELETE ステートメント) /
  (SELECT ステートメント + INSERT ステートメント + UPDATE ステートメント +
  DELETE ステートメント)
```

delete_time 削除応答時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの DELETE に対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

応答時間とは、フェデレーテッド・サーバーが DELETE ステートメントをデータ・ソースにサブミットしてからデータ・ソースが DELETE を処理したことをフェデレーテッド・サーバーに応答するまでの時間です。

表 578. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、このデータ・ソースに対する DELETE が処理されるのを待機するために生じた実際の時間を判別できます。この情報は、キャパシティー・プランニングおよびチューニングを行うときに便利です。

destination_service_class_id - 宛先サービス・クラス ID : モニター・エレメント

このエレメントのしきい値違反レコードが生成された時に、アクティビティーが再マップされたサービス・サブクラスの ID。 しきい値アクションが REMAP ACTIVITY 以外の場合、このエレメントの値はゼロです。

表 579. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントは、アクティビティーが再マップされたサービス・クラスをたどるのに使用できます。 このエレメントを使用して、特定のサービス・サブクラスに対してマップされたアクティビティー数の総計を計算することもできます。

details_xml - 詳細 XML : モニター・エレメント

統計イベント・モニターによって収集されたシステム・モニター・エレメントの一部が入った XML 文書。

表 580. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
統計	EVENTS_SCSTATS	常に収集される
統計	EVENT_WLSTATS	常に収集される

使用法

返される XML 文書のスキーマは、ファイル `sqllib/misc/DB2MonCommon.xsd` に用意されています。最上位エレメントは **system_metrics** です。

このモニター・エレメントと **metrics** モニター・エレメントに関連付けられている各 XML 文書には、同じシステム・メトリックが入っていますが、重要な相違点が 1 つあります。このモニター・エレメントのメトリックは、一般的には 0 から始ま

り、次にデータベースが活動化されるときまで累積し続けます。一方、 **metrics** のメトリックは、統計が最後に収集された時からのメトリックの値の変化を示すために計算されます。

重要: バージョン 10.1 フィックスパック 1 以降、このモニター・エレメントは統計イベント・モニターでは推奨されなくなりました。今後のリリースでは除去される可能性があります。このエレメントに返される XML メトリック・データを使用している場合は、代わりに **metrics** 文書を使用してください。

device_type - 装置タイプ

このエレメントは、UTILSTART イベントに関連する装置タイプの ID です。

表 581. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILLOCATION	常に収集される

使用法

変更履歴イベント・モニターでは、このフィールドによって LOCATION フィールドの解釈が決まります。

A TSM

C クライアント

D ディスク

F スナップショット・バックアップ

L ローカル

N DB2 によって内部で生成される

O その他のベンダー・デバイス・サポート

P パイプ

Q カーソル

R フェッチ・データの削除

S サーバー

T テープ

U ユーザー出口

X X/Open XBSA インターフェース

diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント

db2diag ログ・ファイルへの書き込みの待機にかかった時間。値はミリ秒単位で示されます。

表 582. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES _BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 582. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 583. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用すると、db2diag ログ・ファイルへの書き込みに費やされた時間が分かります。パーティション・データベース環境では、診断ディレクトリー・パス (diagpath) に共有ストレージが使用されている場合、この時間の値が大きいことは、db2diag ログ・ファイルの競合を示している可能性があります 。値が大きい場合は、過剰なロギングが行われている可能性もあります (例えば、すべての通知メッセージをログに記録するように **diaglevel** が設定されている場合)。

diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント

エージェントが db2diag ログ・ファイルに書き込んだ回数。

表 584. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 585. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを **diaglog_write_wait_time** モニター・エレメントと組み合わせて使用すると、db2diag ログ・ファイルへの書き込みにかかった平均時間が分かります。

direct_read_reqs - 直接読み取り要求 : モニター・エレメント

1 つ以上のデータ・セクターの直接読み取り実行要求の数。

表 586. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 586. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 587. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 588. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
表スペース	event_tablespace	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

次の数式を使用すると、直接読み取りにより読み取られるセクターの平均数を計算できます。

```
direct_reads / direct_read_reqs
```

direct_read_time - 直接読み取り時間 : モニター・エレメント

直接読み取りの実行に要した経過時間。この値はミリ秒単位で示されます。

表 589. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 589. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 590. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール

表 590. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 591. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
表スペース	event_tablespace	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

次の数式を使用して、セクター当たりの直接読み取り平均時間を計算します。

```
direct_read_time / direct_reads
```

平均時間が長い場合には、入出力が競合していることがあります。

direct_reads - データベースからの直接読み取り : モニター・エレメント

バッファー・プールを使用しない読み取り操作の数。

表 592. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 592. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 592. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	REQUEST METRICS BASE

表 593. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 594. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
表スペース	event_tablespace	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

次の数式を使用すると、直接読み取りにより読み取られるセクターの平均数を計算できます。

```
direct_reads / direct_read_reqs
```

システム・モニターを使用して入出力を追跡するときにこのエレメントを利用すると、装置上のデータベース入出力とそれ以外の入出力を区別できます。

直接読み取りは、最小 512 バイト・セクター単位で処理されます。次の目的に使用します。

- LONG VARCHAR 列の読み取り
- LOB (ラージ・オブジェクト) 列の読み取り
- バックアップの実行

direct_write_reqs - 直接書き込み要求 : モニター・エレメント

1 つ以上のデータ・セクターの直接書き込み実行要求の数。

表 595. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

表 595. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 596. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 597. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
表スペース	event_tablespace	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

次の数式を使用すると、直接書き込みにより書き込まれるセクターの平均数を計算できます。

```
direct_writes / direct_write_reqs
```

direct_write_time - 直接書き込み時間 : モニター・エレメント

直接書き込みの実行に要した経過時間。この値はミリ秒単位で報告されます。

表 598. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 598. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 599. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 600. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される

表 600. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表スペース	event_tablespace	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

次の数式を使用して、セクター当たりの直接書き込み平均時間を計算します。

```
direct_write_time / direct_writes
```

平均時間が長い場合には、入出力が競合していることがあります。

direct_writes - データベースへの直接書き込み : モニター・エレメント

バッファー・プールを使用しない書き込み操作の数。

表 601. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 601. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	REQUEST METRICS BASE

表 602. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 603. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
表スペース	event_tablespace	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

次の数式を使用すると、直接書き込みにより書き込まれるセクターの平均数を計算できます。

```
direct_writes / direct_write_reqs
```

システム・モニターを使用して入出力を追跡するときにこのエレメントを利用すると、装置上のデータベース入出力とそれ以外の入出力を区別できます。

直接書き込みは、最小 512 バイト・セクター単位で処理されます。次の目的に使用します。

- LONG VARCHAR 列の書き込み
- LOB (ラージ・オブジェクト) 列の書き込み
- リストアの実行
- ロードの実行
- MPFA が使用可能である場合 (デフォルト) の SMS 表スペースへの新規エクステントの割り振り

disabled_peds - 無効化された partial early distinct のモニター・エレメント

使用できるソート・ヒープが不十分であるために、partial early distinct 操作が無効化された回数。

表 604. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 604. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 605. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 605. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 606. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
接続	event_conn	-
ステートメント	event_stmt	-
トランザクション	event_xact	-
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントと **total_peds** モニター・エレメントを組み合わせて使用すると、partial early distinct 操作に十分なソート・ヒープ・メモリーが付与されているかを大まかに調べることができます。**total_peds** モニター・エレメントに対する **disabled_peds** モニター・エレメントの比率が高い場合、データベースのパフォーマンスは最適ではない可能性があります。ソート・ヒープ・サイズかソート・ヒープしきい値のいずれか一方または両方を大きくすることを検討してください。

disconn_time データベース非アクティブ化タイム・スタンプ

アプリケーションがデータベースとの接続を切断した日時 (データベース・レベルでは、アプリケーションが最後に切断した日時)。

エレメント ID

disconn_time

エレメント・タイプ

timestamp

表 607. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

表 607. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、次の時点からの経過時間を計算できます。

- データベースがアクティブだったとき (データベース・レベルの情報の場合)。
- 接続がアクティブだったとき (接続レベルの情報の場合)。

disconnects 切断回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わって、フェデレーテッド・サーバーがこのデータ・ソースから切断した合計回数が含まれています。

表 608. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、フェデレーテッド・サーバーがいずれかのアプリケーションに代わってこのデータ・ソースから切断した合計回数を判別できます。このエレメントと CONNECT カウントを組み合わせて使用すると、データ・ソースに現在接続中であるとフェデレーテッド・サーバーのこのインスタンスが判断しているアプリケーションの数を判別できます。

dl_conns - デッドロックに関係している接続 : モニター・エレメント

デッドロックに関係している接続の数。

表 609. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	event_deadlock	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

モニター・アプリケーションでこのエレメントを使用すると、イベント・モニター・データ・ストリーム内で処理されるデッドロック接続イベント・レコードの数を確認できます。

dyn_compound_exec_id - 動的コンパウンド・ステートメントの実行可能 ID のモニター・エレメント

動的に作成されたコンパウンド SQL ステートメントまたは PL/SQL の無名ブロックを識別する実行可能 ID。

表 610. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される

使用法

MON_GET_PKG_CACHE_STMT 表関数を使用して、ルーチンのステートメント・テキストを検索するためにこの値を使用します。

dynamic_sql_stmts 試行された動的 SQL ステートメント

試行された動的 SQL ステートメントの数。

表 611. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 612. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

以下の各操作のオカレンスによって、dynamic_sql_stmts モニター・エレメントの値が 1 ずつ増分します。

- FETCH

- OPEN
- PREPARE
- CLOSE
- DESCRIBE

使用法 このエレメントを使用すると、データベース・レベルまたはアプリケーション・レベルで成功した SQL ステートメントの合計数を計算できます。

```
  dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= モニター期間中のスループット
```

edu_ID - エンジン・ディスパッチ可能単位 ID : モニター・エレメント

このメモリー・プールと関連付けられるエンジン・ディスパッチ可能単位の ID。

表 613. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE

使用法

このモニター・エレメントは、表関数 MON_GET_MEMORY_POOL によって戻される場合、記述されているメモリー・プールが PRIVATE でなければ、NULL になります。

eff_stmt_text - 有効なステートメント・テキスト : モニター・エレメント

SQL ステートメントがステートメント・コンセントレーターの結果として変更された場合の、そのステートメントの有効なテキスト。

表 614. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 615. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	常に収集される

使用法

ステートメント・コンセントレーターが使用可能になっていて、ステートメント・コンセントレーターの結果としてステートメント・テキストが変更された場合、こ

のモニター・エレメントには有効なステートメント・テキストが含まれます。そうでない場合は、このモニター・エレメントには 0 バイト長のテキスト・ストリングが含まれます。

effective_isolation - 有効な分離 : モニター・エレメント

このステートメントの有効な分離レベル。

表 616. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 617. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このエレメントを使用して、ステートメントの実行中に使用された分離レベルを判別します。

effective_lock_timeout - 有効なロック・タイムアウト : モニター・エレメント

このアクティビティーの有効なロック・タイムアウト値。この値は秒単位で報告されます。

表 618. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

effective_query_degree - 有効な照会の度合い : モニター・エレメント

このアクティビティーの有効な照会の並列処理の度合い。

表 619. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 620. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

elapsed_exec_time ステートメント実行経過時間

DCS ステートメント・レベルでは、ホスト・データベース・サーバー上で SQL 要求の処理に要した経過時間を示します。この値はこのサーバーによって報告されています。表に書き込むイベント・モニターでは、このエレメントの値は、BIGINT データ・タイプを使用してマイクロ秒単位で指定されます。

host_response_time エレメントと比較すると、このエレメントには DB2 Connect とホスト・データベース・サーバーの間のネットワーク経過時間が含まれていません。ほかのレベルでは、この値は特定のデータベースやアプリケーションのために実行されたすべてのステートメント、あるいは特定の回数のデータ転送を使用したステートメントについてのホスト実行時間の合計を示します。

表 621. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント、タイム・スタンプ
アプリケーション	appl	ステートメント、タイム・スタンプ
DCS データベース	dcs_dbase	ステートメント、タイム・スタンプ
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ
データ伝送	stmt_transmissions	ステートメント、タイム・スタンプ

ステートメント・レベルでのスナップショット・モニターの場合、このカウンターはリセットできません。その他のレベルではこのカウンターはリセットできます。

使用法 このエレメントと経過時間に関する他のモニター・エレメントを組み合わせて使用すると、データベース・サーバーでの SQL 要求の処理を評価したり、パフォーマンス上の問題を特定するときに利用できます。

host_response_time エレメントからこのエレメントの値を引くと、 DB2 Connect とホスト・データベース・サーバーの間のネットワーク経過時間を計算できます。

注: dcs_dbase、dcs_appl、dcs_stmt、および stmt_transmissions レベルの場合、 *elapsed_exec_time element* は z/OS® データベースのみに適用されます。 DB2 Connect ゲートウェイが Windows、Linux、AIX、またはその他の UNIX データベースに接続している場合は、 *elapsed_exec_time* はゼロとして報告されます。

empty_pages_deleted - 削除された空ページ : モニター・エレメント

削除された疑似空ページの数。疑似空ページとは、すべてのキーが疑似削除されたページのことです。

表 622. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

empty_pages_reused - 再利用された空ページ : モニター・エレメント

再利用された疑似空ページの数。疑似空ページとは、すべてのキーが疑似削除されたページのことです。

表 623. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

entry_time - エントリー時間 : モニター・エレメント

アクティビティーが開始された時刻。

表 624. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

使用法

estimated_cpu_entitlement - 見積もりの CPU 割り当て率のモニター・エレメント

サービス・サブクラスが CPU 割り当て率に基づいて消費するように構成されている、ホストまたは LPAR 上の合計 CPU の比率。サービス・サブクラスは、消費するように構成された比率を上回ることも下回ることもなく消費するという想定で構成されます。

この計算にどのサービス・クラスを参加させるかは、サンプリング期間にわたって測定された実際の CPU 使用率、対 WLM_DISP_MIN_UTIL データベース・マネージャー構成設定値に基づいて決定されます。サービス・クラス自体、サービスが競合するサービス・クラス、または親のサービス・クラス (ある場合) が CPU リミットによって受ける影響は、計算には考慮されていません。

表 625. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE

estimatedsqlcost_threshold_id - 見積もり SQL コストしきい値 ID : モニター・エレメント

アクティビティーに適用されていた ESTIMATEDSQLCOST しきい値の ID。

表 626. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 ESTIMATEDSQLCOST しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

estimatedsqlcost_threshold_value - 見積もり SQL コストしきい値 : モニター・エレメント

アクティビティーに適用されていた ESTIMATEDSQLCOST しきい値の上限。

表 627. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、ESTIMATEDSQLCOST しきい値がアクティビティーに適用されている場合、その値を判別します。

estimatedsqlcost_threshold_violated - 見積もり SQL コストしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが ESTIMATEDSQLCOST しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 628. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた ESTIMATEDSQLCOST しきい値にアクティビティーが違反したかどうかを判別します。

event_id - イベント ID モニター・エレメント

イベントに関連付けられた ID。これを他のモニター・エレメントと一緒に使用して、イベントを一意的に識別します。このモニター・エレメントは、**xmlid** モニター・エレメントを戻すインターフェースでは、このエレメントの一部となります。

表 629. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
変更履歴	changesummary dbdbmcfg ddlstmtexec evmonstart regvar txncompletion utillocation utilphase utilstart utilstop	常に収集される
ロッキング	• event_lock • event_lock_participants • event_lock_participants _activities • event_lock_activity_values	常に収集される

表 629. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
パッケージ・キャッシュ	• event_pkgcache • event_pkgcache_metrics • event_pkgcache_stmt_args	常に収集される
作業単位	event_uow	常に収集される

使用法

この ID の値は、以下のように、ID が出現するイベント・レコードのタイプによって異なります。

ロック・イベント・モニターのレコード
イベントの数値 ID。 ID はデータベースの活動化時にリサイクルされます。**event_timestamp**、**event_id**、**member**、および **event-type** の組み合わせによってユニーク性が保証されます。

作業単位イベント・モニターのレコード
接続ごとに固有の UOW ID の別名。 **event_timestamp**、**event_id**、**event-type**、**member**、および **appl_id** の組み合わせによってユニーク性が保証されます。

パッケージ・キャッシュ・イベント・モニターのレコード
イベントの数値 ID。 ID はデータベースの活動化時にリサイクルされます。**event_timestamp**、**event_id**、**member**、および **event-type** の組み合わせによってユニーク性が保証されます。

変更履歴イベント・モニターのレコード
イベントの数値 ID。 ID はデータベースの活動化時にリサイクルされます。イベントの固有性は、**event_timestamp**、**event_id**、**member**、および **event-type** の組み合わせによって保証されます。

event_monitor_name イベント・モニター名

イベント・データ・ストリームを作成したイベント・モニターの名前。

エレメント ID
event_monitor_name

エレメント・タイプ
情報

表 630. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	常に収集される

使用法 このエレメントを使用すると、分析中のデータとシステム・カタログ表内の特定のイベント・モニターを関連付けることができます。この名前は、

SYSCAT.EVENTMONITORS カタログ表の NAME 列にある名前 (CREATE EVENT MONITOR および SET EVENT MONITOR のステートメントに指定されている) と同じものです。

event_time イベント時刻

イベントが発生した日時。

表 631. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表スペース	event_tablespace	-
表	event_table	-

使用法 このエレメントは、イベントを時系列順に関連付けるのに利用できます。

event_timestamp - イベント・タイム・スタンプ : モニター・エレメント

このイベント・レコードをデータベース・マネージャーが生成した時刻。

表 632. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
変更履歴	changesummary dbdbmcfg ddlstmtexec evmonstart regvar txncompletion utillocation utilphase utilstart utilstop	常に収集される
ロッキング	• event_lock • event_lock_participants • event_lock_participants _activities • event_lock_activity_values	常に収集される
パッケージ・キャッシュ	• event_pkgcache • event_pkgcache_metrics • event_pkgcache_stmt_args	常に収集される
作業単位	• event_uow	常に収集される

event_type - イベント・タイプ・モニター・エレメント

報告されているイベントのイベント・タイプ。変更履歴、ロック、およびパッケージ・キャッシュのイベント・モニターによってのみ報告されます。

多くのイベント・モニターは、1 つのタイプのイベントのみをキャプチャーします。例えば、作業単位イベント・モニターは、作業単位完了イベントを記録します。このイベントは、作業単位の完了時にその作業単位に関連するデータをキャプチャーします。データベース、表、バッファー・プール、表スペースのイベント・モニターのような他のイベント・モニターは、データベースが非アクティブ化された後に、それぞれ 1 つのタイプのイベントをキャプチャーします。

一方、さまざまなタイプのイベントを生成するイベント・モニターもあります。このようなイベント・モニターは、イベントのタイプを **event_type** モニター・エレメントで報告します。 表 633 に、このモニター・エレメントを戻すイベント・モニターを示します。 1014 ページの表 634 には、このモニター・エレメントの取り得る値を示します。

表 633. イベント・モニター情報

イベント・モニター	論理データ・グループ	モニター・エレメントの収集レベル
変更履歴	changesummary dbdbmcfg ddlstmtexec evmonstart regvar txncompletion utillocation utilphase utilstart utilstop	常に収集される
ロッキング	• event_lock • event_lock_participants • event_lock_participants _activities • event_lock_activity_values	常に収集される
作業単位	該当なし。詳しくは、 1014 ページの表 634 でこのイベント・モニターの使用上の注意を参照してください。	常に収集される
パッケージ・キャッシュ	• pkgcache	常に収集される

表 634. **event_type** として可能な値

イベント・モニター	**event_type** エレメントの取り得る値	使用上の注意
変更履歴	• DBCFG • DBCFGVALUES • DBMCFG • DBMCFGVALUES • REGVAR • REGVARVALUES • DDLSTMTEXEC • TXNCOMPLETION • EVMONSTART • UTILSTART • UTILSTOP • UTILSTARTPROC • UTILSTOPPROC • UTILPHASESTART • UTILPHASESTOP	**event_type** モニター・エレメントは、このイベント・モニターの出力に列として含まれます。
ロッキング	• LOCKTIMEOUT • LOCKWAIT • DEADLOCK	通常の表または未フォーマット・イベント (UE) 表のいずれかに書き込む場合、**event_type** モニター・エレメントは、このイベント・モニターの出力に列として含まれますまた、ルーチン EVMON_FORMAT_UE_TO_TABLES または EVMON_FORMAT_UE_TO_XML によって作成される出力にも含まれます。
パッケージ・キャッシュ	• PKGCACHEBASE • PKGCACHEDETAILED	通常の表ではなく、未フォーマット・イベント (UE) 表に書き込む場合に、**event_type** モニター・エレメントは、このイベント・モニターの出力に列として含まれますまた、ルーチン EVMON_FORMAT_UE_TO_TABLES または EVMON_FORMAT_UE_TO_XML によって作成される出力にも含まれます。
作業単位	• UOW	未フォーマット・イベント (UE) 表に書き込む場合に限り、**event_type** モニター・エレメントは、このイベント・モニターの出力に列として含まれますまた、これは、このイベント・モニターによって作成された UE 表にのみ含まれるものであり、ルーチン EVMON_FORMAT_UE_TO_TABLES または EVMON_FORMAT_UE_TO_XML によって作成される出力には含まれません。

evmon_activates イベント・モニター活動化回数

イベント・モニターが活動化された回数。

エレメント ID

evmon_activates

エレメント・タイプ
カウンター

このモニター・エレメントは、すべてのイベント・モニターから報告されるわけではありません。詳しくは、表 635を参照してください。

表 635. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表	event_table	常に収集される
表スペース	event_tablespace	常に収集される
バッファー・プール	event_bufferpool	常に収集される
デッドロック	event_deadlock	常に収集される
デッドロック	event_dlconn	常に収集される
詳細付きデッドロック	event_detailed_dlconn	常に収集される
ステートメント	event_stmt_history	常に収集される
接続*	-	
トランザクション*	-	
統計*	-	
しきい値違反 *	-	

* このイベント・モニターは、カタログ表 SYSCAT.EVENTMONITORS の evmon_activates 列のみ更新します。イベント・モニター出力表に書き込まれる論理データ・グループには、このモニター・エレメントは含まれません。

使用法 このエレメントを使用すると、表 635 にリストしたイベント・モニターから戻される情報の相互関係を明らかにすることができます。このエレメントは、表書き込みイベント・モニターの場合にのみ使用されます。ファイルまたはパイプに書き込むイベント・モニターの場合には保持されません。

表 635 のイベント・モニターはすべて、イベント・モニターの活動化時に SYSCAT.EVENTMONITORS カタログ表の evmon_activates 列を更新します。この変更はログに記録されるため、DATABASE CONFIGURATION によって次が表示されます。

```
All committed transactions have been written to disk = NO
```

イベント・モニターが AUTOSTART オプションを使用して作成されている場合、最初のユーザーがデータベースに接続してデータベースを非アクティブ化するために即時に切断すると、ログ・ファイルが作成されます。

特に記載がない限り、これらのイベント・モニターは、イベント・データを書き込む表内にも列として evmon_activates の値を持っています。

この表に記載していないイベント・モニターは、 evmon_activates モニター・エレメントを報告しません。

evmon_wait_time - イベント・モニターの待機時間モニター・エレメント

イベント・モニター・レコードが使用できるようになるまでエージェントが待機した時間。

エージェントがイベント・モニター・レコードを書き込もうとしたが、高速ライター・レコードが使用できるようになるまでブロックされた場合に、待機が発生します。高速ライターは、大量のイベント・モニター・データを表、ファイル、またはパイプに並行して書き込むために使用されます。

測定単位はミリ秒です。

表 636. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY _ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 636. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	REQUEST METRICS BASE

表 637. イベント・モニター情報

イベント・タイプ	イベント・モニターの出力タイプ	論理データ・グループ	モニター・スイッチ
作業単位	TABLE、FILE、PIPE	uow_metrics	REQUEST METRICS BASE
作業単位	UE TABLE (EVMON_FORMAT_UE_TO_XML 関数で処理する場合)	system_metrics 文書に報告されます。常に収集される	REQUEST METRICS BASE
作業単位	UE TABLE (EVMON_FORMAT_UE_TO_TABLES 関数で処理する場合)	UOW_EVENT 表のメトリック文書および UOW_METRICS 表に報告されます常に収集される	REQUEST METRICS BASE
パッケージ・キャッシュ	TABLE、FILE、PIPE	pkgcache_metrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	UE TABLE (EVMON_FORMAT_UE_TO_XML 関数で処理する場合)	activity_metrics 文書に報告されます。常に収集される	ACTIVITY METRICS BASE
パッケージ・キャッシュ	UE TABLE (EVMON_FORMAT_UE_TO_TABLES 関数で処理する場合)	PKGCACHE_EVENT 表のメトリック文書および PKGCACHE_METRICS 表に報告されます常に収集される	ACTIVITY METRICS BASE
アクティビティー	TABLE、FILE、PIPE	event_activitymetrics	ACTIVITY METRICS BASE
アクティビティー	TABLE、FILE、PIPE	event_activity (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
統計	TABLE、FILE、PIPE	event_scstats (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
統計	TABLE、FILE、PIPE	event_wlstats (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

evmon_waits_total - イベント・モニター合計待機回数モニター・エレメント

イベント・モニター・レコードが使用できるようになるまでエージェントが待機した回数。

エージェントがイベント・モニター・レコードを書き込もうとしたが、高速ライター・レコードが使用できるようになるまでブロックされた場合に、待機が発生します。高速ライターは、大量のイベント・モニター・データを表、ファイル、またはパイプに並行して書き込むために使用されます。

表 638. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	REQUEST METRICS BASE

表 639. イベント・モニター情報

イベント・タイプ	イベント・モニターの出力タイプ	論理データ・グループ	モニター・スイッチ
作業単位	TABLE	uow_metrics	REQUEST METRICS BASE
作業単位	UE TABLE (EVMON_FORMAT_UE_TO_XML 関数で処理する場合)	system_metrics 文書に報告されます。常に収集される	REQUEST METRICS BASE
作業単位	UE TABLE (EVMON_FORMAT_UE_TO_TABLES 関数で処理する場合)	UOW_EVENT 表のメトリック文書および UOW_METRICS 表に報告されます常に収集される	REQUEST METRICS BASE
パッケージ・キャッシュ	TABLE	pkgcache_metrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	UE TABLE (EVMON_FORMAT_UE_TO_XML 関数で処理する場合)	activity_metrics 文書に報告されます。常に収集される	ACTIVITY METRICS BASE
パッケージ・キャッシュ	UE TABLE (EVMON_FORMAT_UE_TO_TABLES 関数で処理する場合)	PKGCACHE_EVENT 表のメトリック文書および PKGCACHE_METRICS 表に報告されます常に収集される	ACTIVITY METRICS BASE
アクティビティー	TABLE	event_activitymetrics	ACTIVITY METRICS BASE
アクティビティー	TABLE	event_activity (DETAILS_XML 文書に報告されます)	ACTIVITY METRICS BASE
統計	TABLE	event_scstats (DETAILS_XML 文書に報告されます)	REQUEST METRICS BASE
統計	TABLE	event_wlstats (DETAILS_XML 文書に報告されます)	REQUEST METRICS BASE

exec_list_cleanup_time - 実行リストのクリーンアップ時刻のモニター・エレメント

実行リストが最後に整理された時刻。

実行リストが整理されたことがない場合、このエレメントは NULL を返します。

表 640. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

exec_list_mem_exceeded - 実行リストのメモリー超過のモニター・エレメント

ルーチンのすべてのステートメント情報をキャプチャーするために、モニター・ヒープ内のメモリーが不十分だったかどうかを示します。

このエレメントで可能な値は、次のとおりです。

Y すべてのステートメント情報をキャプチャーするにはメモリーが不十分です。

N 他のすべての場合

ルーチンの実行リストが整理される時にこの値は「N」に設定されます。

表 641. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

executable_id 実行可能 ID : モニター・エレメント

実行された SQL ステートメント・セクションを一意的に識別する、データ・サーバーで生成された不透明なバイナリー・トークン。 非 SQL アクティビティーの場合、長さが 0 のストリング値が戻されます。

表 642. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 642. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 643. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このモニター・エレメントをさまざまなモニター・インターフェースに入力してセクションに関するデータを取得できます。 パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックを取得するのに使用される MON_GET_PKG_CACHE_STMT 表関数は、実行可能 ID を入力とみなします。

executable_list_size - 実行可能リスト・サイズのモニター・エレメント

ある特定の作業単位の実行可能 ID リスト内にある項目の数

表 644. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	uow	

executable_list_truncated - 実行可能リスト切り捨てのモニター・エレメント

実行可能リストが切り捨てられたかどうかを示します。値は YES または NO のいずれかです。処理中に実行可能リスト全体を格納するために十分なメモリーがない場合、リストは切り捨てられる可能性があります。

表 645. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	uow	

evmon_flushes イベント・モニター・フラッシュ回数

FLUSH EVENT MONITOR SQL ステートメントが発行された回数。

エレメント ID

evmon_flushes

エレメント・タイプ

情報

表 646. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表	event_table	常に収集される
表スペース	event_tablespace	常に収集される
バッファー・プール	event_bufferpool	常に収集される

使用法 アプリケーションがデータベースに接続した後、データベース・マネージャーが FLUSH EVENT MONITOR SQL 要求を処理するごとに、この ID が増分します。このエレメントを使用すると、データベース、表、表スペース、およびバッファー・プール・データを特定できます。

executable_id 実行可能 ID : モニター・エレメント

実行された SQL ステートメント・セクションを一意的に識別する、データ・サーバーで生成された不透明なバイナリー・トークン。 非 SQL アクティビティーの場合、長さが 0 のストリング値が戻されます。

表 647. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE

表 647. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 648. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このモニター・エレメントをさまざまなモニター・インターフェースに入力してセクションに関するデータを取得できます。 パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックを取得するのに使用される MON_GET_PKG_CACHE_STMT 表関数は、実行可能 ID を入力とみなします。

execution_id ユーザー・ログイン ID

ユーザーがオペレーティング・システムにログインするときに指定した ID。この ID は、ユーザーがデータベースに接続するときに指定する auth_id とは異なります。

表 649. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本
DCS アプリケーション	dcs_appl_info	基本

表 650. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_connheader	-

使用法 このエレメントを使用すると、モニター対象のアプリケーションを実行している個人のオペレーティング・システム・ユーザー ID を識別できます。

failed_sql_stmts 失敗したステートメント操作

試行されたが失敗した SQL ステートメントの数。

表 651. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 652. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、データベース・レベルまたはアプリケーション・レベルで成功した SQL ステートメントの合計数を計算できます。

```
  dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= モニター期間中のスループット
```

このカウントには、負の SQLCODE を受信したすべての SQL ステートメントを含みます。

このエレメントは、パフォーマンスが低い場合の原因の判別にも役に立ちます。これは、失敗したステートメントがあると、データベース・マネージャーで余分な時間がかかり、その結果としてデータベースのスループットが落ちるからです。

fcm_congested_sends - FCM 輻輳送信のモニター・エレメント

輻輳 (ふくそう) のために遅延したリモート・メンバーへの送信操作の数。

先に送信したデータがリモート・メンバーで受信されていないために、ネットワーク接続上に送信するデータがブロックされた場合に、輻輳状態となります。輻輳は、受信側での処理の待ち時間、または、パケット・ロスおよびデータ再送信を引き起こすネットワーク上の問題が原因である可能性があります。

表 653. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE

fcm_congestion_time - FCM 輻輳時間のモニター・エレメント

リモート・メンバーに対する FCM ネットワーク接続が輻輳 (ふくそう) 状態であった時間。計測単位はミリ秒です。

先に送信したデータがリモート・メンバーで受信されていないために、ネットワーク接続上に送信するデータがブロックされた場合に、輻輳状態となります。輻輳は、受信側での処理の待ち時間、または、パケット・ロスおよびデータ再送信を引き起こすネットワーク上の問題が原因である可能性があります。

表 654. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE

fcm_num_congestion_timeouts - FCM 輻輳タイムアウト数のモニター・エレメント

FCM ネットワーク接続上の輻輳 (ふくそう) 状態が、許容期間内に自動解決されなかった回数。タイムアウトが発生するのは、リモート・メンバーが一時的に到達不能になったために、送信側が接続をドロップした場合です。

先に送信したデータがリモート・メンバーで受信されていないために、ネットワーク接続上に送信するデータがブロックされた場合に、輻輳状態となります。輻輳は、受信側での処理の待ち時間、または、パケット・ロスおよびデータ再送信を引き起こすネットワーク上の問題が原因である可能性があります。

表 655. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE

fcm_num_conn_lost - FCM 接続損失数のモニター・エレメント

リモート・メンバーに対する FCM ネットワーク接続が予期せず切断された回数。

表 656. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE

fcm_num_conn_timeouts - FCM 接続タイムアウト数のモニター・エレメント

リモート・メンバーに対する FCM ネットワーク接続の確立を試行中にタイムアウトになった回数。

表 657. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE

fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント

FCM 通信層によって分散された内部要求 (RPC など) について受信されたデータの量。この値はバイト単位で示されます。

表 658. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 658. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 659. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、FCM サブシステムを介して送信されているデータ・ボリュームのうちのどれほどの量が、実際の表データではなく、要求メッセージまたは応答メッセージのトラフィックのボリュームであるかを判別します。

fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント

fcm_message_recv_wait_time モニター・エレメントは、以前に送信された FCM 要求メッセージの結果を含む FCM 応答メッセージの待機にエージェントが費やした時間を格納します。

この値は、FCM を使用してパーティション間で応答を送信するのに必要な時間とサブエージェントが要求メッセージを処理するのに必要な時間の両方を反映します。値はミリ秒単位で示されます。

表 660. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 660. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 661. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用すると、複数パーティション・インスタンスにおいて、特定のパーティション上で、他のパーティション上で要求が処理されるのを待機するのにどれだけの時間がかかったかを判別できます。

fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント

以前に送信された FCM 要求メッセージの結果を含む FCM 応答メッセージの一部として受信されるバッファーの合計数。

表 662. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 662. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 663. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このエレメントを使用すると、受信された FCM メッセージごとの平均ボリュームと、単一の FCM メッセージの受信を待機するのにかかった平均時間の両方を判別できます。

fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント

内部 FCM 要求を介して送信されるデータ・ボリュームの量。この値はバイト単位で示されます。

表 664. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 664. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 665. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 665. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このエレメントを使用して、FCM サブシステムを介して送信されているデータ・ボリュームのうちのどれほどの量が、実際の表データを送信するためにではなく、要求メッセージおよび応答メッセージのトラフィックを送信するために使用されているかを判別します。

fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント

FCM メッセージ送信に対するブロックにかかった時間。値はミリ秒単位で示されます。このモニター・エレメントは、データベース・システム上で内部要求を配布するときに、FCM チャネルから FCM バッファーがフラッシュされるためのブロックに費やした時間を反映します。

表 666. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 666. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 667. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このエレメントを使用して、FCM サブシステムを介した FCM 要求メッセージの送信を待機するのにエージェントが費やしている時間を判別します。 FCM デーモンのビジー状況によっては、エージェントはメッセージの送信を試行する際に待機しなければならない場合があります。

fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント

FCM 通信メカニズムを使用して、内部要求の一部として配布されたバッファーの合計数。

表 668. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 668. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 669. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、FCM 要求メッセージごとに送信されたデータの平均の量と、FCM メッセージごとの平均の待機時間の両方を判別します。

fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント

FCM 通信層を介して受信されたデータの合計量。この値はバイト単位で示されます。

表 670. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 670. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 671. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE

表 671. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このパーティション上で FCM を使用して受信されたデータ (メッセージ・トラフィックと表キュー・データの両方を含む) の合計ボリュームを示します。

fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント

FCM を介したデータの受信の待機にかかった合計時間。値はミリ秒単位で示されます。

表 672. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 672. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 673. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、このデータベース・パーティション上で FCM を介したデータの受信の待機にかかった合計時間を判別します。これには、要求メッセージに対する応答からのデータと、表キュー・データの両方が含まれます。

fcm_recvs_total - FCM 受信の合計 : モニター・エレメント

FCM 通信メカニズムを使用して、内部要求のために受信されたバッファーの合計数。fcm_recvs_total モニター・エレメントの値は、fcm_message_recvs_total モニター・エレメントと fcm_tq_recvs_total モニター・エレメントの値の合計です。

表 674. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 674. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 675. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを **fcm_recv_wait_time** モニター・エレメントと一緒に使用して、FCM 受信操作ごとの平均待機時間と、FCM 受信操作から戻される平均ボリュームを判別します。

fcm_send_volume - FCM 送信ボリューム : モニター・エレメント

FCM 通信層によって配布されたデータの合計量。この値はバイト単位で示されます。

表 676. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 676. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 677. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、FCM を使用して送信されたデータ (メッセージ・トラフィックと表キュー・データの両方を含む) の合計量を判別します。

fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント

FCM 送信操作に対するブロックにかかった時間。これには、内部要求用のバッファーがフラッシュされるのを待機するのに要した時間と、表キューでデータを送信するとき、ウィンドウ・カウントの確認応答を待機するのに要した時間が含まれます。値はミリ秒単位で示されます。

表 678. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 678. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 679. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、FCM を介したデータ送信の待機にかかった合計時間を判別します。これには、要求メッセージと表キュー・データの両方が含まれます。

fcm_sends_total - FCM 送信の合計 : モニター・エレメント

内部 FCM 通信層を使用して送信されたバッファーの合計数。

表 680. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 680. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 681. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE

表 681. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、FCM 受信操作ごとの平均待機時間と、FCM 受信操作から戻される平均ボリュームを判別します。

fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント

FCM 通信層によって表キュー上で受信されたデータの量。この値はバイト単位で示されます。

表 682. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 682. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 683. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、表キューを使用して受信されたデータの合計ボリュームを判別します。

fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント

表キューから次のバッファーを受信するのを待機していた時間。値はミリ秒単位で示されます。

表 684. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 684. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 685. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、表キュー上のデータを受信するのをエージェントが待機している時間を判別します。

fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント

内部 FCM 通信メカニズムを使用して表キューから受信されたバッファーの合計数。

表 686. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 686. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 687. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE

表 687. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを **fcm_tq_recv_volume** および **fcm_tq_recv_wait_time** と組み合わせて使用すると、受信した表キュー・バッファーごとの平均待機時間とボリュームを判別できます。

fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント

FCM 通信層によって表キュー上で送信されたデータの量。この値はバイト単位で示されます。

表 688. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 688. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 689. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、FCM を使用して表キュー・バッファー送信によって送信されたデータの合計ボリュームを判別します。

fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント

表キューによって次のバッファーを送信するのを待機していた時間。これは、表キューの受信側の終端からのウィンドウ・カウントの確認応答の待機にかかった時間を反映します。値はミリ秒単位で示されます。

表 690. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 690. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 691. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、FCM を使用した、表キューによるデータ・バッファーの送信を待機するのにかかっている時間を判別します。

fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント

内部 FCM 通信メカニズムを使用して送信された表キュー・データが入っているバッファーの合計数。

表 692. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 692. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 693. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE

表 693. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを **fcm_tq_send_volume** および **fcm_tq_send_wait_time** モニター・エレメントと組み合わせて使用すると、平均データ・ボリューム、および表キューを使用してバッファーが送信されるのを待機した時間を判別できます。

fetch_count 成功したフェッチの数

成功した物理フェッチの数か、または試みられた物理フェッチの数。スナップショット・モニター・レベルに応じて決まります。

- stmt および dynsql スナップショット・モニター・レベルでありイベント・タイプがステートメントの場合は、特定のカーソル上で実行されて成功したフェッチの数。
- dcs_stmt スナップショット・モニター・レベルの場合は、 (アプリケーションによってフェッチされた行数にかかわらず) ステートメントの実行時に試みられた物理フェッチの数。この状態の場合、**fetch_count** は、ステートメントの処理中にサーバーがゲートウェイに対して応答データを送り返す必要があった回数を表します。

表 694. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント
動的 SQL	dynsql	ステートメント

動的 SQL スナップショット・モニターの場合、このカウンターはリセットできます。

表 695. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	常に収集される

使用法

このエレメントを使用すると、データベース・マネージャー内の現在のアクティビティーのレベルがわかります。

ステートメント・イベント・モニターは、パフォーマンス上の理由から、すべての FETCH ステートメントを対象にステートメント・イベント・レコードを生成するわけではありません。レコード・イベントが生成されるのは、FETCH がゼロ以外の SQLCODE を戻した場合だけです。

files_closed - クローズしたデータベース・ファイル : モニター・エレメント

閉じられたデータベース・ファイルの合計数。

表 696. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 697. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 698. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法

データベース・マネージャーは、バッファー・プールへの書き込みおよびバッファー・プールからの読み取りを行うためにファイルを開きます。アプリケーションが同時に開けるデータベース・ファイルの最大数は、 **maxfilop** 構成パラメーターによりコントロールされています。最大値に達すると、新しいファイルを開く前に、ファイルが 1 つ閉じられます。実際に開かれたファイルの数と閉じられたファイルの数は等しくならないことに注意してください。

このエレメントを使用すると、**maxfilop** 構成パラメーターの最適な値を判別するのに役立てることができます。

first_active_log 先頭アクティブ・ログ・ファイル番号

最初のアクティブ・ログ・ファイルのファイル番号。

エレメント ID

first_active_log

エレメント・タイプ

情報

表 699. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 700. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	detail_log	基本

表 701. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *last_active_log* および *current_active_log* エレメントを組み合わせて使用すると、アクティブ・ログ・ファイルの範囲を判別できます。アクティブ・ログ・ファイルの範囲を知っていると、ログ・ファイルに必要なディスク・スペースを判別するのに役立ちます。

また、このエレメントを使用すると、どのログ・ファイルにデータがあるか判別でき、スプリット・ミラー・サポートが必要なログ・ファイルを識別するのに役立ちます。

first_overflow_time 最初のイベント・オーバーフロー時刻

このオーバーフロー・レコードに記録されている最初のオーバーフローの日時。

表 702. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
オーバーフロー・レコード	event_overflow	-

使用法 このエレメントと *last_over_flow time* を組み合わせて使用すると、オーバーフロー・レコードを生成するのに要した経過時間を計算できます。

fs_caching - ファイル・システム・キャッシング : モニター・エレメント

特定の表スペースがファイル・システム・キャッシングを使用するかどうかを示します。**fs_caching** が 0 の場合、ファイル・システム・キャッシングは使用可能です。 **fs_caching** が 1 の場合は、ファイル・システム・キャッシングは使用不可です。

表 703. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 704. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

表 705. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表スペース	event_tablespace	-

fs_id - 固有のファイル・システム識別番号 : モニター・エレメント

このエレメントは、ストレージ・パスまたはコンテナーが指し示すファイル・システムに対してオペレーティング・システムが提供する固有の識別番号を示します。

表 706. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 707. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法

このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するデータを収集します。

- **db_storage_path**
- **sto_path_free_size**
- **fs_used_size**
- **fs_total_size**

fs_total_size - ファイル・システムの合計サイズ : モニター・エレメント

このエレメントは、ストレージ・パスまたはコンテナーが指し示すファイル・システムの容量を (バイト単位で) 示します。

表 708. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 709. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法

このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するデータを収集することができます。

- **db_storage_path**
- **sto_path_free_size**
- **fs_used_size**
- **fs_id**

fs_used_size - ファイル・システム上で使用されるスペースの量 : モニター・エレメント

このエレメントは、ストレージ・パスまたはコンテナーが指し示すファイル・システム上で既に使用されているスペースの量を (バイト単位で) 示します。

表 710. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 711. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法

このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するデータを収集することができます。

- **db_storage_path**
- **sto_path_free_size**
- **fs_total_size**
- **fs_id**

global_transaction_id - グローバル・トランザクション ID のモニター・エレメント

イベントが発生した時刻で使用していた XA トランザクション ID。

表 712. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	ddlstmtexec txncompletion	常に収集される
作業単位	uow	常に収集される

使用法

変更履歴イベント・モニターでは、イベント発生時に使用されていたグローバル・トランザクション ID です。これは、トランザクション・ログの一部となる SQLP_GXID 構造体のデータ・フィールドです。

gw_comm_error_time 通信エラー時刻

DCS アプリケーションがホスト・データベースへの接続を試みたときに、または SQL ステートメントを処理していたときに通信エラー (SQL30081) が最後に発生した日時。

表 713. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ

使用法

このエレメントは、通信エラーおよび管理用通知ログ に記録される通信エラー・ログと組み合わせて、問題判別に使用できます。

gw_comm_errors 通信エラー

DCS アプリケーションがホスト・データベースへの接続を試みたときに、または SQL ステートメントを処理していたときに通信エラー (SQL30081) が発生した回数。

表 714. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 時間をかけて通信エラーの数をモニターすることによって、DB2 Connect ゲートウェイに特定のホスト・データベースとの接続の問題があるかどうかを評価できます。通常のエラーしきい値と考えられるものを設定できるため、エラーの数がこのしきい値を超えるたびに、通信エラーの調査を行うことができます。

管理用通知ログ に記録される通信エラー・ログとともに、このエレメントを問題判別に使用してください。

gw_con_time DB2 Connect ゲートウェイの最初の接続開始

ホスト・データベースへの最初の接続が DB2 Connect ゲートウェイから開始された日時。

表 715. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ
DCS アプリケーション	dcs_appl	タイム・スタンプ

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

gw_connections_top ホスト・データベースへの同時接続の最大数

最初のデータベース接続以降、 DB2 Connect ゲートウェイが処理したホスト・データベースへの同時接続の最大数。

エレメント ID

gw_connections_top

エレメント・タイプ

水準点

表 716. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本

使用法 このエレメントは、DB2 Connect ゲートウェイでのアクティビティーのレベルと、関連したシステム・リソースの使用状況を知るのに役立ちます。

gw_cons_wait_client クライアントの要求送信を待機している接続の数

DB2 Connect ゲートウェイが処理しているホスト・データベースへの接続のうち、クライアントが要求を送信するのを待機している現在の接続数。

エレメント ID

gw_cons_wait_client

エレメント・タイプ

ゲージ

表 717. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
DCS データベース	dcs_dbase	基本

使用法 この値は頻繁に変わることがあります。実際のゲートウェイの使用率を把握するには、長期にわたって周期的にサンプリングを行わなければなりません。

gw_cons_wait_host ホストの応答を待機している接続の数

DB2 Connect ゲートウェイが処理しているホスト・データベースへの接続のうち、ホストからの応答を待機している現在の接続数。

エレメント ID

gw_cons_wait_host

エレメント・タイプ

ゲージ

表 718. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
DCS データベース	dcs_dbase	基本

使用法 この値は頻繁に変わることがあります。実際のゲートウェイの使用率を把握するには、長期にわたって周期的にサンプリングを行わなければなりません。

gw_cur_cons DB2 Connect の現在の接続数

DB2 Connect ゲートウェイが処理しているホスト・データベースへの現在の接続数。

表 719. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
DCS データベース	dcs_dbase	基本

使用法 このエレメントは、DB2 Connect ゲートウェイでのアクティビティーのレベルと、関連したシステム・リソースの使用状況を知るのに役立ちます。

gw_db_alias ゲートウェイでのデータベース別名

ホスト・データベースに接続するために DB2 Connect ゲートウェイで使用される別名。

エレメント ID
gw_db_alias

エレメント・タイプ
情報

表 720. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

gw_exec_time DB2 Connect ゲートウェイ処理の経過時間

DB2 Connect ゲートウェイがアプリケーションの要求を処理するのに要した時間 (接続が確立された以降)、または単一ステートメントを処理するのに要した時間 (秒およびマイクロ秒単位)。

表 721. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用して、全体の処理時間のうちどの部分が DB2 Connect ゲートウェイの処理によるものかを判別します。

gw_total_cons DB2 Connect の接続試行合計回数

最後の `db2start` コマンドまたは最後のリセット以降に、 DB2 Connect ゲートウェイから試行された接続の合計回数。

表 722. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
DCS データベース	dcs_dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントは、DB2 Connect ゲートウェイでのアクティビティーのレベルと、関連したシステム・リソースの使用状況を知るのに役立ちます。

hadr_connect_status HADR 接続状況 : モニター・エレメント

データベースの高可用性災害時リカバリー (HADR) の現在の接続状況。

表 723. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_HADR 表関数 - 高可用性災害時リカバリー (HADR) のモニター情報を戻す	ACTIVITY METRICS BASE

表 724. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、データベースの HADR 接続状況を判別できます。

このエレメントのデータ・タイプは整数です。

データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントの値は、以下のいずれかの定数です。

SQLM_HADR_CONN_CONNECTED
データベースはそのパートナー・ノードに接続しています。

SQLM_HADR_CONN_DISCONNECTED
データベースはそのパートナー・ノードに接続していません。

SQLM_HADR_CONN_CONGESTED
データベースはそのパートナー・ノードに接続していますが、接続が混雑しています。1 次とスタンバイのペアの間に TCP/IP ソケット接続が依然として存在しているものの、一方の終端が他方の終端に送信できない場合に、接続は混雑します。例えば、受信側の終端がソケット接続から受信していないと、 TCP/IP 送信スペースが満杯になります。ネットワーク接続が混雑する理由には、次のものがあります。

- ネットワークを共有するリソースが多すぎるか、ネットワークが 1 次 HADR ノードのトランザクション・ボリュームにとって低速である。
- スタンバイ HADR ノードのあるサーバーが強力でないので、必要な速度で通信サブシステムから情報を取り出せない。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_connect_time HADR 接続時刻 : モニター・エレメント

このモニター・エレメントは、高可用性災害時回復 (HADR) 接続時刻、HADR 輻輳 (ふくそう) 時刻、または HADR 切断時刻のいずれかの値を返します。

表 725. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用すると、現在の HADR 接続状況が始まった時刻を判別できます。

データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントの意味は、 **hadr_connect_status** エレメントの値に従属します。

- **hadr_connect_status** エレメントの値が SQLM_HADR_CONN_CONNECTED の場合は、このエレメントは接続時刻を示す。
- **hadr_connect_status** エレメントの値が SQLM_HADR_CONN_CONGESTED の場合は、このエレメントは輻輳 (ふくそう) の開始時刻を示す。
- **hadr_connect_status** エレメントの値が SQLM_HADR_CONN_DISCONNECTED の場合は、このエレメントは切断時刻を示す。

HADR エンジン・ディスパッチ可能単位 (EDU) の開始以降に接続が行われていない場合、接続状況は「切断」として報告され、切断時刻に HADR EDU 起動時刻が使用されます。 HADR 接続イベントや切断イベントは比較的頻度が低いので、DFT_MON_TIMESTAMP スイッチが OFF の場合でも時刻は収集されて報告されます。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_heartbeat HADR ハートビート : モニター・エレメント

高可用性災害時回復 (HADR) の接続において、連続して受信されなかったハートビートの数。

データベースがハートビートを再び受信すると、この数値はゼロにリセットされます。データベースが HADR プライマリーまたはスタンバイの役割の場合、このエレメントは HADR 接続の正常性を示します。

表 726. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

スナップショット・モニターの場合、このカウンターはリセットできません。

使用上の注意:

このエレメントを使用して、HADR 接続の正常性を判別できます。

ハートビートとは、他の HADR データベースから定期的に送信されるメッセージのことです。このエレメントの値がゼロの場合は、ハートビートは欠落しておらず、接続は健全です。値が大きくなるほど、接続の状態は悪くなります。

切断モードでは、欠落したハートビートは適用されないため、常に 0 として表示されます。

ハートビート間隔は、**hadr_timeout** や **hadr_peer_window** などの構成パラメーターから得られます。最大設定値は 30 秒です。

このエレメントのデータ・タイプは整数です。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR 役割が標準の場合、このエレメントは無視してください。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_local_host - HADR ローカル・ホスト : モニター・エレメント

高可用性災害時リカバリー (HADR) ローカル・ホスト名値は、ホスト名のストリングか、「1.2.3.4」などの IP アドレスのストリングとして表示されます。

表 727. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR ローカル・ホスト名を判別できます。HADR データベース構成パラメーターは静的です。パラメーターに対する変更内容

は、データベースが停止して再始動するまでは有効になりません。このモニター・エレメントは、データベース構成ファイル中の値ではなく、 HADR システムが実際に使用している値を報告します。

このエレメントに対する変更は、データベースを活動化すると有効になります。データベースが既にオンラインの場合は、1 次データベース上の HADR を停止して再始動すると有効になります。

注: 使用する名前は、1 つの IP アドレスに解決されなければなりません。複数のアドレスに解決される名前を使用すると、HADR を始動しようとするときにエラーが発生します。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_local_service HADR ローカル・サービス : モニター・エレメント

ローカル HADR TCP サービス。この値は、サービス名のストリングか、ポート番号のストリングとして表示されます。

表 728. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR ローカル・サービス名を判別できます。

このエレメントに対する変更は、データベースを活動化すると有効になります。データベースが既にオンラインの場合は、1 次データベース上の HADR を停止して再始動すると有効になります。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_log_gap HADR ログ・ギャップ

このエレメントは、 1 次ログ・シーケンス番号 (LSN) とスタンバイ・ログ LSN の間のギャップの現行の平均を示します。ギャップはバイト数で測定されます。

表 729. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_HADR 表関数 - 高可用性災害時リカバリー (HADR) のモニター情報を戻す	ACTIVITY METRICS BASE

表 730. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、 1 次 HADR データベース・ログとスタンバイ HADR データベース・ログの間のギャップを判別できます。

ログ・ファイルが切り捨てられると、次のログ・ファイルの LSN は、直前のファイルが切り捨てられていないかのように始まります。この LSN の穴には、ログ・データは含まれません。この種の穴により、ログ・ギャップが 1 次 HADR データベース・ログとスタンバイ HADR データベース・ログの間の実際のログの差を反映しない可能性があります。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_peer_window - HADR ピア・ウィンドウ : モニター・エレメント

HADR_PEER_WINDOW データベース構成パラメーターの値。

表 731. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用すると、HADR_PEER_WINDOW データベース構成パラメーターの値を判別できます。

hadr_peer_window_end HADR ピア・ウィンドウ終了 : モニター・エレメント

高可用性災害時リカバリー (HADR) 1 次データベースがアクティブである限り、この 1 次データベースがピアまたは切断済みピア状態であることを保証する期間が終了する時点。

表 732. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用すると、1 次データベースがピアまたは切断済みピア状態であることを保証する期間が終了する時点を特定することができます。

1 次データベースによって報告される値と、スタンバイ・データベースによって報告される値が違うことがあります。この違いが生じる理由は、1 次データベースはハートビート・メッセージを送信する際に値を更新しますが、この新しい値がスタンバイ・データベースで表示されるのはこのメッセージがスタンバイ・データベースで受け取られて処理された後に限られるからです。

データベースがピアまたは切断済みピア状態でなくなっても、このモニター・エレメントの値はリセットされません。最後に認識された値が保持され、戻されます。データベースがピア状態に達することがなかった場合は、ゼロの値が戻されます。

ピア・ウィンドウ終了時刻は、1 次データベースによって設定された後、スタンバイ・データベースに送信されます。このため、ピア・ウィンドウ終了時刻の値は 1 次データベースのクロックに基づいています。ピア・ウィンドウ終了時刻と 1 次データベースのダウン時刻を比較する際に、2 つのクロックの同期が十分取れていない場合には、オフセットを追加してタイム・スタンプを 1 次データベースのクロックに変換する必要が生じることがあります。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

hadr_primary_log_file HADR 1 次ログ・ファイル : モニター・エレメント

1 次 HADR データベース上の現行ログ・ファイルの名前。

表 733. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、1 次 HADR データベース上の現行ログ・ファイルを判別できます。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_primary_log_lsn HADR 1 次ログ LSN : モニター・エレメント

1 次 HADR データベースの現在のログの位置。ログ・シーケンス番号 (LSN) とは、データベースのログ・ストリーム中のバイト・オフセットのことです。

表 734. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、1 次 HADR データベース上の現在のログの位置を判別できます。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_primary_log_page HADR 1 次ログ・ページ : モニター・エレメント

1 次 HADR データベース上の現在のログ位置を示す現行ログ・ファイルのページ番号。ページ番号はログ・ファイルに関連しています。例えば、ページ・ゼロはファイルの先頭です。

表 735. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、1 次 HADR データベース上の現行ログ・ページを判別できます。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_remote_host HADR リモート・ホスト : モニター・エレメント

高可用性災害時リカバリー (HADR) リモート・ホスト名値は、ホスト名のストリングか、「1.2.3.4」などの IP アドレスのストリングとして表示されます。

表 736. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR リモート・ホスト名を判別できます。

このエレメントに対する変更は、データベースを活動化すると有効になります。データベースが既にオンラインの場合は、1 次データベース上の HADR を停止して再始動すると有効になります。

注: 使用する名前は、1 つの IP アドレスに解決されなければなりません。複数のアドレスに解決される名前を使用すると、HADR を始動しようとするときにエラーが発生します。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。*hadr_role* モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_remote_instance HADR リモート・インスタンス : モニター・エレメント

リモート HADR インスタンス名。

表 737. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR リモート・インスタンス名を判別できます。

このエレメントに対する変更は、データベースを活動化すると有効になります。データベースが既にオンラインの場合は、1 次データベース上の HADR を停止して再始動すると有効になります。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_remote_service HADR リモート・サービス : モニター・エレメント

リモート HADR TCP サービス。この値は、サービス名のストリングか、ポート番号のストリングとして表示されます。

表 738. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR リモート・サービス名を判別できます。

このエレメントに対する変更は、データベースを活動化すると有効になります。データベースが既にオンラインの場合は、1 次データベース上の HADR を停止して再始動すると有効になります。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_role HADR の役割

データベースの高可用性災害時リカバリー (HADR) の現在の役割。

表 739. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_HADR 表関数 - 高可用性災害時リカバリー (HADR) のモニター情報を戻す	ACTIVITY METRICS BASE

表 740. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、データベースの HADR の役割を判別できます。

このエレメントのデータ・タイプは整数です。

このエレメントの値は、以下のいずれかの定数です。

SQLM_HADR_ROLE_STANDARD
データベースは HADR データベースではありません。

SQLM_HADR_ROLE_PRIMARY
データベースは 1 次 HADR データベースです。

SQLM_HADR_ROLE_STANDBY
データベースはスタンバイ HADR データベースです。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

hadr_standby_log_file HADR スタンバイ・ログ・ファイル : モニター・エレメント

スタンバイ HADR データベース上の現行ログ・ファイルの名前。

表 741. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、スタンバイ HADR データベース上の現行ログ・ファイルを判別できます。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_standby_log_lsn HADR スタンバイ・ログ LSN : モニター・エレメント

スタンバイ HADR データベースの現在のログの位置。ログ・シーケンス番号 (LSN) とは、データベースのログ・ストリーム中のバイト・オフセットのことです。

表 742. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、スタンバイ HADR データベース上の現在のログの位置を判別できます。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_standby_log_page HADR スタンバイ・ログ・ページ : モニター・エレメント

スタンバイ HADR データベース上の現在のログ位置を示す現行ログ・ファイルのページ番号。ページ番号はログ・ファイルに関連しています。例えば、ページ・ゼロはファイルの先頭です。

表 743. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、スタンバイ HADR データベース上の現行ログ・ページを判別できます。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_state HADR の状態 : モニター・エレメント

データベースの高可用性災害時リカバリー (HADR) の現在の状態。

表 744. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_HADR 表関数 - 高可用性災害時リカバリー (HADR) のモニター情報を戻す	ACTIVITY METRICS BASE

表 745. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、データベースの HADR の状態を判別できます。

このエレメントのデータ・タイプは整数です。データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントの値は、以下のいずれかの定数です。

SQLM_HADR_STATE_DISCONNECTED
データベースはそのパートナー・データベースに接続していません。

SQLM_HADR_STATE_LOC_CATCHUP
データベースはローカル・キャッチアップを行っています。

SQLM_HADR_STATE_REM_CATCH_PEND
データベースはそのパートナーに接続してリモート・キャッチアップを行うのを待っています。

SQLM_HADR_STATE_REM_CATCHUP
データベースはリモート・キャッチアップを行っています。

SQLM_HADR_STATE_PEER
1 次データベースとスタンバイ・データベースが接続され、ピア状態になっています。

SQLM_HADR_STATE_DISCONN_PEER
1 次データベースとスタンバイ・データベースが切断済みピア状態になっています。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

hadr_syncmode HADR 同期モード : モニター・エレメント

データベースの高可用性災害時リカバリー (HADR) の現在の同期モード。

表 746. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_HADR 表関数 - 高可用性災害時リカバリー (HADR) のモニター情報を戻す	ACTIVITY METRICS BASE

表 747. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、データベースの HADR 同期モードを判別できます。

このエレメントのデータ・タイプは整数です。

このエレメントに対する変更は、データベースを活動化すると有効になります。データベースが既にオンラインの場合は、1 次データベース上の HADR を停止して再始動すると有効になります。

データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントの値は、以下のいずれかの定数です。

SQLM_HADR_SYNCMODE_SYNC
SYNC モード。

SQLM_HADR_SYNCMODE_NEARSYNC
NEARSYNC モード。

SQLM_HADR_SYNCMODE_ASYNC
ASYNC モード。

SQLM_HADR_SYNCMODE_SUPERASYNC
SUPERASYNC モード。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_timeout HADR タイムアウト : モニター・エレメント

そのパートナーからの通信がなく、パートナーとの間の接続が失敗したと HADR データベース・サーバーが見なすまでに要する秒数。

表 748. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_HADR 表関数 - 高可用性災害時リカバリー (HADR) のモニター情報を戻す	ACTIVITY METRICS BASE

表 749. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR タイムアウト値を判別できます。

このエレメントに対する変更は、データベースを活動化すると有効になります。データベースが既にオンラインの場合は、1 次データベース上の HADR を停止して再始動すると有効になります。

重要: このモニター・エレメントは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『HADR のいくつかのモニター・インターフェースが推奨されなくなった』を参照してください。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hash_join_overflows ハッシュ結合のオーバーフロー

ハッシュ結合データが、使用可能なソート・ヒープ・スペースを超えた回数。

エレメント ID
hash_join_overflows

エレメント・タイプ
カウンター

表 750. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 751. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 データベース・レベルでは、 hash_join_small_overflows の値がこの hash_join_overflows の 10% を超える場合は、ソート・ヒープ・サイズを大きくすることを検討してください。アプリケーション・レベルの値は、個々のアプリケーションについてハッシュ結合のパフォーマンスを評価するときに使用できます。

hash_join_small_overflows ハッシュ結合の短精度オーバーフロー

ハッシュ結合データが、使用可能なソート・ヒープ・スペースを 10% を超えない範囲で超えた回数。

エレメント ID
hash_join_small_overflows

エレメント・タイプ
カウンター

表 752. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 753. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 この値と hash_join_overflows の値が大きい場合は、ソート・ヒープのしきい値を大きくすることを検討してください。この値が hash_join_overflows の 10% を超える場合は、ソート・ヒープ・サイズを大きくすることを検討してください。

histogram_type ヒストグラム・タイプ : モニター・エレメント

ヒストグラムのタイプ (ストリング形式)。

ヒストグラムには、7 つのタイプがあります。

CoordActQueueTime
ネストなしアクティビティーがキュー (しきい値キューなど) に入れられていた時間のヒストグラム (ミリ秒単位)。コーディネーター・メンバー上で測定されます。

CoordActExecTime
ネストなしアクティビティーのコーディネーター・メンバー上での実行時間のヒストグラム (ミリ秒単位)。実行時間には、初期化されている時間またはキューに入れられている時間は含まれません。カーソルの場合、実行時間にはオープン、フェッチ、およびクローズ要求に要する時間のみ含まれます。アクティビティーがサービス・サブクラス間で再マップされると、実行時間ヒストグラムは、アクティビティーが実行を完了するサービス・サブクラスについてのみ更新されます。

CoordActLifetime
ネストなしアクティビティーがデータベース・マネージャーによって識別されてから、そのアクティビティーが実行を完了するまでの経過時間のヒストグラム (ミリ秒単位)。コーディネーター・メンバー上で測定されます。 アクティビティーをサービス・サブクラス間で再マップした場合、存続時間ヒストグラムは、アクティビティーが実行を完了するサービス・サブクラスについてのみ更新されます。

CoordActInterArrivalTime
ネストなしコーディネーター・アクティビティーが到着してから次が到着するまでの時間間隔のヒストグラム (ミリ秒単位)。到着間隔時間の平均値は、アクティビティーがシステムに入るときに使用されるサービス・サブクラスを対象に計算されます。アクティビティーをサービス・サブクラス間で再マップした場合、アクティビティーの再マップ先のサービス・サブクラスの到着間隔時間ヒストグラムは影響を受けません。

CoordActEstCost
ネストなし DML アクティビティーの見積コストのヒストグラム (timeron 単位)。アクティビティーの見積コストは、アクティビティーがシステムに入るときのサービス・サブクラスに関してのみカウントされます。

ReqExecTime
要求の実行時間のヒストグラム (ミリ秒単位)。要求には、コーディネーター・メンバーでの要求と、コーディネーター・メンバーと非コーディネーター・メンバーの両方でのサブリクエスト (RPC 要求、SMP サブエージェント要求など) が含まれます。含まれている要求には、アクティビティーが関連付けられている場合もあれば、そうでない場合もあります。例えば、このヒストグラムには PREPARE 要求と OPEN 要求の両方が含まれていますが、OPEN 要求の方は常にカーソル・アクティビティーに関連付けられているのに対し、PREPARE 要求の方はアクティビティーの一部ではありませ

ん。再マップに関係するサービス・サブクラスの実行時間ヒストグラムでは、そのサービス・サブクラス内で部分要求が費やす実行時間部分がカウントされます。

UowLifetime
データベース・マネージャーによって作業単位が識別されてから、その作業単位の実行を完了 (コミットまたはロールバック) するまでの経過時間のヒストグラム (ミリ秒単位)。

表 754. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	常に収集される

使用法

このエレメントを使用すると、ヒストグラムのタイプを識別できます。複数のヒストグラムが同じ統計レコードに所属する場合がありますが、各タイプごとに 1 つずつしか所属しません。

hld_application_handle - ロックを保持しているアプリケーションの ID : モニター・エレメント

システム全体での、ロックを保持しているアプリケーションのユニーク ID。ロックを保持しているアプリケーションが不明または見つからない場合、NULL 値が戻されます。

表 755. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

hld_member - ロックを保持しているアプリケーションのデータベース・メンバー

ロックを保持しているアプリケーションのデータベース・メンバー。

表 756. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

使用法

ロックがリモート・メンバーで保持されている場合、**hld_member** の値は -2 です。ロックが保持されているメンバーを判別するには、MON_GET_LOCKS 表関数を使用して、検索引数として **lock_name** を指定してください。

host_ccsid ホスト・コード化文字セット ID

ホスト・データベースのコード化文字セット ID (CCSID) です。

エレメント ID
host_ccsid

エレメント・タイプ
情報

表 757. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

host_db_name ホスト・データベース名

情報が収集されているホスト・データベースの実名、またはアプリケーションの接続先のホスト・データベースの実名。これは、このホスト・データベースが作成されたときに付けられた名前です。

表 758. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

hostname - ホスト名 : モニター・エレメント

データベース・メンバーが存在するマシンのホスト名。

表 759. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
DB2_CLUSTER_HOST_STATE 管理ビューおよび DB2_GET_CLUSTER_HOST_STATE 表関数 - ホストに関する情報の取得	ACTIVITY METRICS BASE
MON_GET_CF_CMD 表関数 - クラスター・キャッシング・ファシリティーのコマンド処理時間の取得	ACTIVITY METRICS BASE
MON_GET_CF_WAIT_TIME 表関数 - クラスター・キャッシング・ファシリティーのコマンド待機時間の取得	ACTIVITY METRICS BASE
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE
MON_GET_SERVERLIST 表関数 - メンバーの優先順位の詳細を取得	ACTIVITY METRICS BASE

表 759. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS - サービス・サブクラス・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	ACTIVITY METRICS BASE

host_name - ホスト名 : モニター・エレメント

クラスター・キャッシング・ファシリティープロセスがあるホストの名前。

表 760. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
DB_MEMBERS 表関数	ACTIVITY METRICS BASE
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

host_prdid - ホスト製品/バージョン ID

サーバー上で実行中の製品とバージョン。

表 761. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 これを使用して、DRDA ホスト・データベース製品の製品とコード・バージョンを識別できます。 PPPVVRRM の形式になっています。各部分の定義は次のとおりです。

- PPP は、次のホスト DRDA 製品を示す。
  - ARI の場合 : DB2 Server for VSE & VM
  - DSN の場合 : DB2 for z/OS
  - QSQ の場合 : DB2 for i

– SQL の場合 : 他の DB2 製品
- VV は 2 桁でバージョン番号を示す (バージョン番号が 1 桁の場合には、高位の桁は 0 になります)。
- RR は 2 桁でリリース番号を示す (リリース番号が 1 桁の場合には高位の桁は 0 になります)。
- M は 1 文字で修正レベルを示します (0-9 または A-Z)。

host_response_time ホスト応答時間

DCS ステートメント・レベルでは、ステートメントが DB2 Connect ゲートウェイから処理のためにホストに送信された時刻と、ホストから結果を受信した時刻との間の経過時間です。

DCS データベースおよび DCS アプリケーションのレベルでは、特定のアプリケーションまたはデータベースに対して実行されたすべてのステートメントについての経過時間の合計です。データ伝送レベルでは、この多数のデータ伝送を使用したすべてのステートメントに関するホスト応答時間の合計です。

表 762. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ
データ伝送	stmt_transmissions	ステートメント、タイム・スタンプ

ステートメント・レベルでのスナップショット・モニターの場合、このカウンターはリセットできません。その他のレベルではこのカウンターはリセットできます。

使用法

このエレメントと送信されたアウトバウンド・バイト数および受信されたアウトバウンド・バイト数を組み合わせて使用すると、次のようにしてアウトバウンド応答時間 (転送速度) を計算できます。

```
(送信されたアウトバウンド・バイト数 + 受信されたアウトバウンド・
バイト数) / ホストの応答時間
```

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

id - クラスター・キャッシング・ファシリティー ID モニター・エレメント

db2nodes.cfg ファイルで定義されているクラスター・キャッシング・ファシリティー ID。

表 763. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
DB2_MEMBER および DB2_CF 管理ビューおよび DB2_GET_INSTANCE_INFO 表関数	ACTIVITY METRICS BASE
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CF_CMD 表関数 - クラスター・キャッシング・ファシリティーのコマンド処理時間の取得	ACTIVITY METRICS BASE
MON_GET_CF_WAIT_TIME 表関数 - クラスター・キャッシング・ファシリティーのコマンド待機時間の取得	ACTIVITY METRICS BASE

ida_recv_volume - 受信した合計データ量 : モニター・エレメント

データベース・サーバーがインデータベース分析プロセスから受信した合計データ量 (バイト単位)。

表 764. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
EVMON_FORMAT_UE_TO_XML 表関数 - 未フォーマット・イベントの XML への変換	REQUEST METRICS BASE
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	REQUEST METRICS BASE
MON_FORMAT_XML_WAIT_TIMES_BY _ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 764. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 765. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
パッケージ・キャッシュ	event_pkgcache_metrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
作業単位	event_uow_metrics	REQUEST METRICS BASE

ida_recv_wait_time - データ受信の待機に費やされた時間 : モニター・エレメント

インデータベース分析プロセスからデータを受信するために待機していた時間の合計。

表 766. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
EVMON_FORMAT_UE_TO_XML 表関数 - 未フォーマット・イベントの XML への変換	REQUEST METRICS BASE
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	REQUEST METRICS BASE
MON_FORMAT_XML_WAIT_TIMES_ BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 766. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 767. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
パッケージ・キャッシュ	event_pkgcache_metrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
作業単位	event_uow_metrics	REQUEST METRICS BASE

ida_recvs_total - データ受信回数 : モニター・エレメント

インデータベース分析プロセスからデータを受信した合計回数。

表 768. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
EVMON_FORMAT_UE_TO_XML 表関数 - 未フォーマット・イベントの XML への変換	REQUEST METRICS BASE
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	REQUEST METRICS BASE
MON_FORMAT_XML_WAIT_TIMES _BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

表 768. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 769. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
パッケージ・キャッシュ	event_pkgcache_metrics	ACTIVITY METRICS BASE

表 769. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
作業単位	event_uow_metrics	REQUEST METRICS BASE

ida_send_volume - 送信された合計データ量 : モニター・エレメント

データベース・サーバーからインデータベース分析プロセスに送信された合計データ量 (バイト単位)。

表 770. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
EVMON_FORMAT_UE_TO_XML 表関数 - 未フォーマット・イベントの XML への変換	REQUEST METRICS BASE
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	REQUEST METRICS BASE
MON_FORMAT_XML_WAIT_TIMES _BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 770. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 771. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
パッケージ・キャッシュ	event_pkgcache_metrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
作業単位	event_uow_metrics	REQUEST METRICS BASE

ida_send_wait_time - データ送信の待機に費やされた時間 : モニター・エレメント

インデータベース分析プロセスにデータを送信するために待機していた時間の合計。

表 772. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
EVMON_FORMAT_UE_TO_XML 表関数 - 未フォーマット・イベントの XML への変換	REQUEST METRICS BASE
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	REQUEST METRICS BASE
MON_FORMAT_XML_WAIT_TIMES _BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 772. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 773. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
パッケージ・キャッシュ	event_pkgcache_metrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
作業単位	event_uow_metrics	REQUEST METRICS BASE

ida_sends_total - データ送信回数 : モニター・エレメント

インデータベース分析プロセスにデータが送信された合計回数。

表 774. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
EVMON_FORMAT_UE_TO_XML 表関数 - 未フォーマット・イベントの XML への変換	REQUEST METRICS BASE
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	REQUEST METRICS BASE
MON_FORMAT_XML_WAIT_TIMES _BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

表 774. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 775. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
パッケージ・キャッシュ	event_pkgcache_metrics	ACTIVITY METRICS BASE

表 775. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
作業単位	event_uow_metrics	REQUEST METRICS BASE

idle_agents - アイドル・エージェント数

アプリケーションにまだ割り当てられておらず、『アイドル』状態でエージェント・プール内に存在するエージェントの数。

表 776. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 このエレメントは、 *num_poolagents* 構成パラメーターを設定するときに利用できます。アイドル・エージェントを持つことでエージェントの要求を処理できるので、パフォーマンスが向上します。

iid - 索引 ID : モニター・エレメント

索引の ID。

表 777. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

inbound_bytes_received 受信されたインバウンド・バイト数

DB2 Connect ゲートウェイがクライアントから受信したバイト数。通信プロトコル (TCP/IP など) による使用量は含まれません。

表 778. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	基本
DCS ステートメント	dcs_stmt	ステートメント

アプリケーション・レベルでのスナップショット・モニターの場合、このカウンターはリセットできます。その他のレベルではこのカウンターはリセットできません。

使用法 このエレメントを使用して、クライアントから DB2 Connect ゲートウェイへのスループットを測定します。

inbound_bytes_sent 送信されたインバウンド・バイト数

DB2 Connect ゲートウェイがクライアントに送信したバイト数。通信プロトコル (TCP/IP など) による使用量は含まれません。

表 779. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	基本
DCS ステートメント	dcs_stmt	ステートメント

アプリケーション・レベルでのスナップショット・モニターの場合、このカウンターはリセットできます。その他のレベルではこのカウンターはリセットできません。

使用法 このエレメントを使用して、DB2 Connect ゲートウェイからクライアントへのスループットを測定します。

inbound_comm_address - インバウンド通信アドレス

これはクライアントの通信アドレスです。例えば、TCP/IP 用の IP アドレスとポート番号などです。

表 780. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

include_col_updates - 列の更新の組み込み : モニター・エレメント

列の更新の組み込みの数。

表 781. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

incremental_bind - 追加バインドのモニター・エレメント

パッケージの追加バインドが実行時に行われたかどうかを示します。可能な値は YES または NO です。

表 782. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participant_activities	

index_jump_scans - 索引ジャンプ・スキャンのモニター・エレメント

ジャンプ・スキャンの数。ジャンプ・スキャンとは、索引の開始キーと停止キーとの間にギャップがあり、結果に関係しない索引のセクションがスキップされる索引スキャンのことです。

表 783. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用すると、ジャンプ・スキャンが予期された比率で生じているかどうかを確認できます。このモニター・エレメントの値が予期されたとおりに増えていない場合、以下について確認してください。

- ジャンプ・スキャンが予期される照会を実行している場合、ターゲット表に適切な複合索引があること、および照会述部によって索引ギャップが生じていることを確認してください。 DB2 オプティマイザーは、索引ギャップがない場合にはジャンプ・スキャンを使用したプランを作成しません。
- ジャンプ・スキャンは、以下のタイプの索引をスキャンしません。
  - 範囲がクラスター化された表索引
  - 拡張索引 (例えば、空間索引など)
  - XML 索引
  - テキスト索引 (Text Search 用)

注: DB2 オプティマイザーは、アクセス・プランを作成する際にコスト分析を実行し、ジャンプ・スキャンを実行する必要があるかどうかを判断します。ジャンプ・スキャンを使用しない方が効率的であるとオプティマイザーが判断することもあります。

index_name - 索引名モニター・エレメント

索引の名前。

表 784. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE

index_schema - 索引スキーマのモニター・エレメント

索引スキーマの名前。

表 785. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE

index_object_pages 索引オブジェクト・ページ数

表に対して定義されたすべての索引が使用するディスク・ページの数。

表 786. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 787. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法 このエレメントを使用すると、特定の表に対して定義された索引が使用する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに索引が大きくなる比率を追跡できます。このエレメントは、パーティション表では戻されません。

index_object_l_pages - 索引データ論理ページ : モニター・エレメント

この表に関連付けられているすべての索引によって使用された、ディスク上の論理ページの数。パーティション化された表の場合は NULL 値が返されます。

表 788. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

- この値は、オブジェクトに物理的に割り振られているスペース量より小さい場合があります。そのようになる可能性があるのは、TRUNCATE ステートメントの REUSE STORAGE オプションを使用した場合です。このオプションを指定すると、表に割り振られているストレージは引き続き割り振られますが、ストレージは空と見なされます。さらに、このモニター・エレメントの値は、オブジェクトに論理的に割り振られているスペース量よりも小さくなる可能性があります。その理由は、論理的に割り振られているスペースの合計には、量的にはわずかですが追加のメタデータが含まれているからです。

  オブジェクトの正確な論理サイズまたは物理サイズを取得するには、ADMIN_GET_TAB_INFO_V97 関数を使用します。この関数を使用すると、このモニター・エレメントに関して報告されるページ数とページ・サイズの積で得られる値よりも正確なオブジェクト・サイズに関する情報が得られます。

index_only_scans - 索引のみのスキャン : モニター・エレメント

索引のみのスキャンの数。索引のみのスキャンは、スキャンの結果が索引へのアクセスのみで得られた場合に生じます。

表 789. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

index_scans - 索引スキャン : モニター・エレメント

索引スキャンの数。

表 790. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

index_tbsp_id - 索引表スペース ID : モニター・エレメント

この表で作成された索引を保持する表スペースの ID。

表 791. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

このエレメントの値は、SYSCAT.TABLESPACES のビューの TBSPACEID 列の値と一致します。

input_db_alias 入力データベース別名

スナップショット関数を呼び出すときに指定するデータベースの別名。

表 792. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl_id_info	基本
表スペース	tablespace_list	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
表	table_list	表
ロック	db_lock_list	基本

使用法 このエレメントを使用すると、モニター・データが適用される特定のデータベースを識別できます。特定のデータベースに関連するモニター情報を要求したとき以外は、このエレメントはブランクとなります。

1 つのデータベースが複数の別名を持つことがあるので、このフィールドの値と *client_db_alias* モニター・エレメントの値とが異なる場合があります。複数のアプリケーションやユーザーが異なる別名を使用して、同じデータベースに接続することができます。

insert_sql_stmts 挿入回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースに INSERT ステートメントを発行した合計回数が含まれています。

表 793. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本

表 793. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、フェデレーテッド・サーバーまたはアプリケーションによりこのデータ・ソースに対して行われたデータベース・アクティビティーのレベルを判別できます。

このエレメントを使用すると、次の公式を使用して、フェデレーテッド・サーバーまたはアプリケーションによるこのデータ・ソースへの書き込みアクティビティーのパーセンテージも判別できます。

```
書き込みアクティビティー =
   (INSERT ステートメント + UPDATE ステートメント + DELETE ステートメント) /
   (SELECT ステートメント + INSERT ステートメント + UPDATE ステートメント +
   DELETE ステートメント)
```

insert_time 挿入応答時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの INSERT に対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

このモニターは最新の値を格納します。

応答時間とは、フェデレーテッド・サーバーが INSERT ステートメントをデータ・ソースにサブミットしてからデータ・ソースが INSERT を処理したことをフェデレーテッド・サーバーに応答するまでの時間です。

表 794. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このデータ・ソースに対する INSERT が処理されるのを待機するために生じた実際の時間を判別できます。この情報は、キャパシティー・プランニングおよびチューニングを行うときに便利です。

insert_timestamp - 挿入タイムスタンプ : モニター・エレメント

insert_timestamp モニター・エレメントは、ステートメントまたはセクションがキャッシュに挿入された時刻を格納します。

動的 SQL スナップショットの場合、ステートメントがキャッシュに入った時刻を示します。
MON_GET_PKG_CACHE_STMT、MON_GET_PKG_CACHE_STMT_DETAILS およびパッケージ・キャッシュ・イベント・モニターの場合、この値の粒度はさらに細かくなり、このステートメントの個々のセクションがキャッシュに挿入された時刻を示します。

表 795. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 796. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

表 797. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このエレメントは、ステートメントがキャッシュに入れられた時刻を指定します。これを使用して、キャッシュ内でのステートメントの存続時間を見積もることができます。

int_auto_rebinds 内部自動再バインド

試行された自動再バインド (または再コンパイル) の数。

表 798. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 799. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 自動再バインドは、パッケージが無効にされている場合にシステムが実行する内部バインドです。再バインドは、データベース・マネージャーが初めてパッケージから SQL ステートメントを実行する必要が生じたときに行われます。パッケージは、例えば次の場合に無効にされます。

- プランが従属している表、ビュー、または索引などのオブジェクトのドロップ。
- 外部キーの追加またはドロップ。
- プランが従属しているオブジェクト特権の取り消し。

このエレメントを使用すると、アプリケーション・レベルまたはデータベース・レベルのデータベース・アクティビティーのレベルを判別できます。int_auto_rebinds はパフォーマンスに大きな影響を与えるので、できるだけ最小限に抑える必要があります。

このエレメントを使用すると、次の公式を使用して、再バインド・アクティビティーのパーセンテージも計算できます。

```
int_auto_rebinds / total number of statements
```

この情報は、アプリケーションのアクティビティーおよびスループットの分析に役立ちます。

int_commits - 内部コミット数 : モニター・エレメント

データベース・マネージャーによって内部で開始されたコミットの合計数。

表 800. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 800. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 801. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 802. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

内部コミットは、以下のいずれかのイベントの間に発生する場合があります。

- 再編成
- インポート

- バインドまたはプリコンパイル
- 明示的な SQL COMMIT ステートメントを実行せずにアプリケーションが終了した場合 (UNIX の場合)

この値には明示的な SQL COMMIT ステートメントは含まれず、次の時点以降のこれらの内部コミットの数を表します。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

このエレメントを使用して、作業単位の総数を計算することができます。これには、以下の式を使用して和を計算します。

```
   commit_sql_stmts
 + int_commits
 + rollback_sql_stmts
 + int_rollbacks
```

注: 計算した作業単位に含まれるのは、次の時点以降の作業単位だけです。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

この計算は、アプリケーションまたはデータベース・レベルで実行できます。

int_deadlock_rollbacks デッドロックによる内部ロールバック

デッドロックのためにデータベース・マネージャーによって開始された強制ロールバックの合計数。ロールバックは、デッドロックを解決するためにデータベース・マネージャーが選択したアプリケーション内の現在の作業単位上で実行されます。

エレメント ID

int_deadlock_rollbacks

エレメント・タイプ

カウンター

表 803. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 804. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法 このエレメントは中断されたデッドロックの数を示しており、並行性の問題の標識として使用できます。 int_deadlock_rollbacks は、データベースのスループットが低下するため、この問題は重要です。

この値は、int_rollbacks が示す値に含まれています。

int_node_splits - 中間ノードの分割 : モニター・エレメント

挿入操作時に中間索引ノードが分割された回数。

表 805. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

int_rollbacks - 内部ロールバック数 : モニター・エレメント

データベース・マネージャーによって内部で開始されたロールバックの合計数。

表 806. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 807. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 808. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

内部ロールバックは、以下のいずれかが正常に完了できないときに起こります。

- 再編成
- インポート
- バインドまたはプリコンパイル
- デッドロック状態またはロック・タイムアウト状態によりアプリケーションが終了した場合
- 明示的なコミットまたはロールバック・ステートメントを実行せずにアプリケーションが終了した場合 (Windows の場合)

この値は、次の時点以降のこれらの内部ロールバックの数を表します。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

この値には明示的な SQL ROLLBACK ステートメントは含まれませんが、**int_deadlock_rollbacks** モニター・エレメントからのカウントは含まれます。

このエレメントを使用すると、次の項目を合計して合計作業単位数を計算できます。

```
   commit_sql_stmts
 + int_commits
 + rollback_sql_stmts
 + int_rollbacks
```

注: 計算した作業単位には、次の時点以降の作業単位が含まれます。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

この計算は、アプリケーションまたはデータベース・レベルで実行できます。

int_rows_deleted 削除された内部行数

これは、内部アクティビティーの結果としてデータベースから削除された行の数です。

表 809. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 810. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される

使用法 このエレメントを使用すると、ユーザーが気付かないようなデータベース・マネージャー内の内部アクティビティーを把握できます。このアクティビティーが高い場合は、表の設計内容を検討して、データベースに定義した参照制約やトリガーが必要かどうかを判別してください。

内部の削除アクティビティーは、次の原因により起こります。

- カスケード削除により ON CASCADE DELETE 参照制約が強制された場合。
- トリガーが起動された場合。

int_rows_inserted 挿入された内部行数

トリガーによって行われた内部アクティビティーの結果としてデータベースに挿入された行の数。

表 811. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 812. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される

使用法 このエレメントを使用すると、データベース・マネージャー内の内部アクティビティーを把握できます。このアクティビティーが高い場合は、このアクティビティーを低減するように設計を変更できるかどうか、検討してください。

int_rows_updated 更新された内部行数

これは、内部アクティビティーの結果としてデータベースから更新された行の数です。

表 813. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 814. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される

使用法 このエレメントを使用すると、ユーザーが気付かないようなデータベース・マネージャー内の内部アクティビティーを把握できます。このアクティビティーが高い場合は、表の設計内容を検討して、データベースに定義した参照制約が必要かどうかを判別してください。

内部の更新アクティビティーは、次の原因により起こります。

- *set null* 行更新により ON DELETE SET NULL ルールに定義した参照制約が強制された場合。
- トリガーが起動された場合。

intra_parallel_state - パーティション内並列処理の現行状態モニター・エレメント

ステートメント、アクティビティー、トランザクション、またはワークロードの各レベルでレポートされるパーティション内並列処理の現行状態。値は、パーティション内並列処理が有効でありステートメントをサブエージェントで実行できることを示す「YES」か、パーティション内並列処理が無効であることを示す「NO」です。

表 815. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE

表 816. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	-	常に収集される
ロック		
作業単位		

invocation_id - 呼び出し ID : モニター・エレメント

ルーチンの 1 つの呼び出しを、作業単位内の同じネスト・レベルの他の呼び出しと区別する ID。その ID は特定のネスト・レベルに関して作業単位内で固有です。

invocation_id モニター・エレメントは、 **stmt_invocation_id** モニター・エレメントの別名です。

表 817. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 818. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
作業単位	パッケージ・リストに報告されます。	-

1 このオプションは推奨されなくなりました。このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを使用して、特定の SQL ステートメントが実行された呼び出しを一意に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント

データ・サーバーがクライアントから IPC 経由で受信したデータの量。この値はバイト単位で示されます。

表 819. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 819. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 820. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント

IPC 通信プロトコルを使用して着信クライアント要求を受信するのにエージェントが費やした時間。値はミリ秒単位で報告されます。

表 821. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 821. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 822. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント

データベース・サーバーがクライアント・アプリケーションから IPC を使用してデータを受信した回数。

表 823. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 823. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 824. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント

データ・サーバーによってクライアントに IPC プロトコル経由で送信されたデータの量。この値はバイト単位で示されます。

表 825. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 825. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 826. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント

クライアントへの IPC 送信のブロックにかかった時間。値はミリ秒単位で示されます。

表 827. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 827. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 828. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント

データベース・サーバーによってクライアント・アプリケーションに IPC を使用してデータが送信された回数。

表 829. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 829. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 830. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

is_system_appl システム・アプリケーション : モニター・エレメント

アプリケーションがシステム・アプリケーションであるかどうかを示します。

表 831. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本

使用法

is_system_appl モニター・エレメントは、アプリケーションが内部システム・アプリケーションであるかどうかを示します。可能な値は以下のとおりです。

0 ユーザー・アプリケーション

1 システム・アプリケーション

key_updates - キーの更新 : モニター・エレメント

キーの更新の数。

表 832. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

last_active_log 最終アクティブ・ログ・ファイル番号

最後のアクティブ・ログ・ファイルのファイル番号。

エレメント ID

last_active_log

エレメント・タイプ

情報

表 833. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 834. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	detail_log	基本

表 835. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *first_active_log* および *current_active_log* エレメントを組み合わせて使用すると、アクティブ・ログ・ファイルの範囲を判別できます。アクティブ・ログ・ファイルの範囲を知っていると、ログ・ファイルに必要なディスク・スペースを判別するのに役立ちます。

また、このエレメントを使用すると、どのログ・ファイルにデータがあるか判別でき、スプリット・ミラー・サポートが必要なログ・ファイルを識別するのに役立ちます。

last_backup 最終バックアップ・タイム・スタンプ

データベース・バックアップが最後に完了した日時。

エレメント ID

last_backup

エレメント・タイプ

timestamp

表 836. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	タイム・スタンプ

使用法 このエレメントを使用すると、バックアップをしばらく行っていないデータベースを識別したり、最新のデータベース・バックアップ・ファイルを識別できます。データベースを一度もバックアップしていない場合は、このタイム・スタンプがゼロに初期化されます。

last_executable_id - 最終実行可能 ID : モニター・エレメント

アプリケーションが最後に完了したステートメントの実行可能 ID。

表 837. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

last_extent - 移動した最終エクステント : モニター・エレメント

表スペース・リバランサー・プロセスにより移動された最終エクステントの数値 ID。

表 838. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE

last_metrics_update - メトリック最終更新タイム・スタンプ : モニター・エレメント

このキャッシュ項目で最後にメトリックが更新された時刻を示すタイム・スタンプ。

表 839. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 840. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

last_overflow_time 最後のイベント・オーバーフロー時刻

このオーバーフロー・レコードに記録されている最後のオーバーフローの日時。

表 841. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
オーバーフロー・レコード	event_overflow	-

使用法 このエレメントと *first_overflow_time* を組み合わせて使用すると、オーバーフロー・レコードを生成するのに要した経過時間を計算できます。

last_reference_time - 最終参照時刻 : モニター・エレメント

アクティビティーが要求によって最後にアクセスされた時刻。

表 842. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

last_request_type - 最終要求タイプ : モニター・エレメント

アプリケーションが完了した最後の要求のタイプ。

表 843. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このモニター・エレメントは、アプリケーションのコーディネーター・メンバーでのみ報告されます。

次の値が可能です。

- CLOSE
- COMMIT
- COMPILE
- DESCRIBE
- EXCSQLSET
- EXECIMMD
- EXECUTE
- FETCH
- INTERNAL *number*。ここで *number* は内部定数の値です。
- OPEN
- PREPARE
- REBIND
- REDISTRIBUTE
- REORG
- ROLLBACK
- RUNSTATS

last_reset 最後のリセット・タイム・スタンプ

GET SNAPSHOT を発行するアプリケーションのモニター・カウンターがリセットされた日時を示します。

エレメント ID

last_reset

エレメント・タイプ

timestamp

表 844. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 845. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	タイム・スタンプ
データベース	dbase	タイム・スタンプ
アプリケーション	appl	タイム・スタンプ
表スペース	tablespace_list	バッファー・プール、タイム・スタンプ
表	table_list	タイム・スタンプ
DCS データベース	dcs_dbase	タイム・スタンプ
DCS アプリケーション	dcs_appl	タイム・スタンプ

使用法 このエレメントを使用すると、データベース・システム・モニターによって戻された情報の有効範囲を判別できます。

カウンターがこれまでにリセットされたことがない場合は、このエレメントはゼロになります。

データベース・マネージャーのカウンターがリセットされるのは、ユーザーがすべてのアクティブ・データベースをリセットしたときだけです。

last_updated - 最終更新タイム・スタンプのモニター・エレメント

この項目が最後に更新された時刻を示すタイム・スタンプ。

表 846. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE

表 846. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント

このエレメントは、このタイプの統計イベント・レコードが最後に作成された時刻をローカル・タイム・スタンプの形式で示します。

表 847. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-
統計	event_wcstats	-
統計	event_qstats	-

使用法

wlm_last_reset および **statistics_timestamp** モニター・エレメントを使用すると、イベント・モニター統計レコード中の統計が収集された期間を判別できます。収集間隔の開始時刻は **wlm_last_reset** で、終了時刻は **statistics_timestamp** です。

lib_id - ライブラリー ID のモニター・エレメント

トリガーおよびトリガー・サブルーチンの内部固有 ID。

このエレメントは、モニター対象オブジェクトに適用できない場合には NULL を返します。

表 848. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される

使用法

このエレメントを使用して、トリガーをそのサブルーチンに関連付けます。

lob_object_pages LOB オブジェクト・ページ数

LOB データが使用するディスク・ページの数。

表 849. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 850. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法 このエレメントを使用すると、特定の表中の LOB データが使用する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに LOB データが大きくなる比率を追跡できます。

lob_object_l_pages - LOB データ論理ページ : モニター・エレメント

この表に関連付けられている LOB によって使用された、ディスク上の論理ページの数。

表 851. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

- この値は、オブジェクトに物理的に割り振られているスペース量より小さい場合があります。そのようになる可能性があるのは、TRUNCATE ステートメントの REUSE STORAGE オプションを使用した場合です。このオプションを指定すると、表に割り振られているストレージは引き続き割り振られますが、ストレージは空と見なされます。さらに、このモニター・エレメントの値は、オブジェクトに論理的に割り振られているスペース量よりも小さくなる可能性があります。その理由は、論理的に割り振られているスペースの合計には、量的にはわずかですが追加のメタデータが含まれているからです。

  オブジェクトの正確な論理サイズまたは物理サイズを取得するには、ADMIN_GET_TAB_INFO_V97 関数を使用します。この関数を使用すると、このモニター・エレメントに関して報告されるページ数とページ・サイズの積で得られる値よりも正確なオブジェクト・サイズに関する情報が得られます。

local_cons - ローカル接続

モニター中のデータベース・マネージャー・インスタンス内のデータベースに現在接続しているローカル・アプリケーションの数。

表 852. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

この数は、データベース・マネージャーで発生している並行処理のレベルを判別するのに役立ちます。この値は頻繁に変化するため、システム使用量の現実的な値を得るためには、長期にわたり特定のインターバルを設けてサンプルを採取する必要があります。

この数値に含まれるのは、データベース・マネージャーと同じインスタンスで開始したアプリケーションだけです。アプリケーションは接続されていますが、データベース内で作業単位を実行している場合としていない場合があります。

このエレメントと rem_cons_in モニター・エレメントを組み合わせて使用すると、**max_connections** 構成パラメーターの設定値を調整するときに利用できます。

local_cons_in_exec - データベース・マネージャーで実行中のローカル接続

モニター中のデータベース・マネージャー・インスタンス内のデータベースに現在接続していて、作業単位を現在処理しているローカル・アプリケーションの数。

表 853. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

この数は、データベース・マネージャーで発生している並行処理のレベルを判別するのに役立ちます。この値は頻繁に変化するため、システム使用量の現実的な値を得るためには、長期にわたり特定のインターバルを設けてサンプルを採取する必要があります。この数値に含まれるのは、データベース・マネージャーと同じインスタンスで開始したアプリケーションだけです。

このエレメントと **rem_cons_in_exec** モニター・エレメントを組み合わせて使用すると、**max_coordagents** 構成パラメーターの設定値を調整するときに利用できます。

以下の推奨事項は、非コンセントレーター構成のみに適用されます。コンセントレーターが使用可能になっている場合、DB2 は多数のクライアント接続を 1 つのも

っと小さなコーディネーター・エージェントのプールに多重化します。この場合、普通は **rem_cons_in_exec** および **local_cons_in_exec** の合計が **max_coordagents** 値に近くなってもかまいません。

- **max_coordagents** を AUTOMATIC に設定する場合は、調整を加えないでください。
- **max_coordagents** を AUTOMATIC に設定せず、**max_coordagents** および **local_cons_in_exec** の合計が **max_coordagents** に近い同じ場合は、**max_coordagents** の値を増やしてください。

local_start_time - ローカル開始時刻 : モニター・エレメント

アクティビティーがメンバーで作業を開始した時刻。これはローカル時間です。アクティビティーがシステムに入ったものの、キューに入れられていてまだ実行を開始していない場合は、このフィールドは空ストリングになります。

表 854. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

使用法

local_transaction_id - ローカル・トランザクション ID のモニター・エレメント

イベントが発生した時刻で使用していたローカル・トランザクション ID。

表 855. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	ddlstmtexec txncompletion	常に収集される
作業単位	uow	

使用法

変更履歴イベント・モニターでは、イベント発生時に使用されていたローカル・トランザクション ID です。これは、トランザクション・ログの一部となる SQLU_TID 構造体です。

location - ロケーション

イベントに関連したロケーションを示します。

表 856. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILLOCATION	常に収集される

使用法

変更履歴イベント・モニターでは、ロケーションは UTILITY_TYPE によって異なります (ロード入力ファイルやバックアップ・ターゲット・パス名など)。

location_type - ロケーション・タイプ・

ロケーションの使用目的の説明。

表 857. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILLOCATION	常に収集される

使用法

utility_type エレメントが LOAD の場合は、以下のいずれかです。

C コピー・ターゲット

D 入力データ

L LOB パス

X XML パス

utility_type エレメントが BACKUP の場合は、以下のいずれかです。

B バックアップのターゲット・ロケーション

utility_type エレメントが RESTORE の場合は、以下のいずれかです。

S リストアのソース・ロケーション

utility_type エレメントが ROLLFORWARD の場合は、以下のいずれかです。

O ROLLFORWARD DATABASE コマンドの一部としてキャプチャーされた代替オーバーフロー・ログ・パス。なお、デフォルトのオーバーフロー・ログ・パスを使用する場合、ロケーション・レコードはキャプチャーされません。

上記以外の場合には、ブランク文字になります。

lock_attributes ロック属性 : モニター・エレメント

ロックを現在保持しているアプリケーションのロック属性。

表 858. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE

表 859. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本
ロック	lock_wait	基本

表 860. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

以下の表に、すべてのロック属性設定をリストします。ロック属性の各設定は、`sqlmon.h` で定義されているビット・フラグ値に基づいています。

表関数におけるロック属性値	API 定数	説明
0000000000000001	SQLM_LOCKATTR_WAIT_FOR_AVAIL	使用可能を待機
0000000000000002	SQLM_LOCKATTR_ESCALATED	エスカレーションによる獲得
0000000000000004	SQLM_LOCKATTR_RR_IN_BLOCK	ブロック内の RR ロック
0000000000000008	SQLM_LOCKATTR_INSERT	ロックの挿入
0000000000000010	SQLM_LOCKATTR_RR	RR スキャンによるロック
0000000000000020	SQLM_LOCKATTR_UPDATE_DELETE	行ロックの更新/削除
0000000000000040	SQLM_LOCKATTR_ALLOW_NEW	新規ロック要求を許可
0000000000000080	SQLM_LOCKATTR_NEW_REQUEST	新規ロック・リクエスター

表関数におけるロック属性値	API 定数	説明
0000000000000200	SQLM_LOCKATTR_INDOUBT	未確定トランザクションが保持するロック。
0000000000000400	SQLM_LOCKATTR_LOW_PRIORITY	優先順位の低いアプリケーションが保持するロック。

戻されるビットで、上の表にリストされていないものは、内部使用のために予約されています。

lock_count ロック・カウント : モニター・エレメント

保持されているロックに関するロックの数。

表 861. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

表 862. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本

表 863. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

この値の範囲は 1 から 255 です。新規ロックが獲得されると増分し、ロックが解放されると減分されます。

lock_count モニター・エレメントの値が 255 のときは、トランザクション期間ロックが保持されていることを示します。この時点でロックが獲得または解放されても **lock_count** モニター・エレメントは増分または減分されなくなります。
lock_count モニター・エレメントは次の 2 つのいずれかの場合に値が 255 に設定されます。

1. 獲得されている新規ロックによって **lock_count** モニター・エレメント値が 255 回増分された場合。
2. トランザクション期間ロックが明示的に獲得された場合。例えば LOCK TABLE ステートメントや INSERT が使用された場合です。

lock_current_mode - 変換前の元のロック・モード : モニター・エレメント

ロック変換操作中に、変換が完了する前に、ロックの獲得を待機しているアプリケーションによって保持されるロック・モード。

表 864. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE

表 865. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本
ロック	lock_wait	基本

表 866. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

ロック変換のシナリオの例を挙げます。更新または削除操作のときにターゲット行の X ロックを待機するとします。トランザクションがその行で S または V ロックを保持している場合、変換が必要になります。この時点で、ロックが X ロックに変換されるのを待機している間は、**lock_current_mode** エレメントには S または V の値が割り当てられます。

可能性のあるロック・モードは、次に示す表でリストされています。

モード	ロックのタイプ	API 定数
	ロックなし	SQLM_LNON
IS	意図的共有ロック	SQLM_LOIS
IX	意図的排他ロック	SQLM_LOIX
S	共有ロック	SQLM_LOOS
SIX	意図的排他ロックで共有	SQLM_LSIX
X	排他ロック	SQLM_LOOX
IN	Intent None	SQLM_LOIN
Z	超排他ロック	SQLM_LOOZ
U	更新ロック	SQLM_LOOU
NS	スキャン共有ロック	SQLM_LONS
NW	次キー弱排他ロック	SQLM_LONW

lock_escalation ロック・エスカレーション : モニター・エレメント

このロックを獲得するのを待機しているアプリケーションがロック・エスカレーション要求の結果であったかどうかを示します。可能な値は Y (はい) および N (いいえ) です。

表 867. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

表 868. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	ロック
ロック	lock_wait	ロック

表 869. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを使用すると、デッドロックの原因が分かりやすくなります。アプリケーションがロック・エスカレーションを起こすようなデッドロックが発生した場合は、ロック・メモリーの量を増やすか、または任意のアプリケーションが要求できるロックのパーセンテージを変更してください。

lock_escals - ロック・エスカレーション数 : モニター・エレメント

ロックが複数の行ロックから 1 つの表ロックにエスカレートされた回数。

表 870. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

表 870. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 871. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 872. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
トランザクション	event_xact	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

アプリケーションが保留するロックの合計数がそのアプリケーションで使用可能なロック・リスト・スペースの最大量に達した場合、またはすべてのアプリケーションが使用するロック・リスト・スペースが合計ロック・リスト・スペースに近くな

ると、ロックはエスカレートされます。使用可能なロック・リスト・スペースの量は、 **maxlocks** および **locklist** 構成パラメーターによって決まります。

1 つのアプリケーションが使用可能な最大ロック数に達して、エスカレートするロックがほかにない場合は、ほかのアプリケーションに割り振られているロック・リストのスペースが使用されます。ロック・リスト全体が満杯になるとエラーが起こります。

このデータ項目には、排他ロック・エスカレーションおよび DB2 pureScale 環境におけるロック・エスカレーションも含めて、すべてのロック・エスカレーションのカウントが含まれます。DB2 pureScale 環境におけるロック・エスカレーションだけを判別するには、**lock_escals_global** モニター・エレメントを使用してください。

過剰なロック・エスカレーションが起こる場合は、いくつかの原因が考えられます。

- 同時アプリケーションの数に対してロック・リスト・サイズ (**locklist**) が小さい場合。
- 各アプリケーションが使用できるロック・リストのパーセント値 (**maxlocks**) が小さい場合。
- 1 つ以上のアプリケーションが使用しているロックの数が多すぎる場合。
- DB2 pureScale 環境において、グローバル・ロック・リスト・サイズ (**cf_lock_sz**) が小さい場合。

これらの問題を解決するには、次のようにしてください。

- **locklist** 構成パラメーター値を大きくする。
- **maxlocks** 構成パラメーター値を大きくする。
- 以下のいずれかの公式を使用し、その値を **maxlocks** と比較することによって、ロック数の多いアプリケーション、またはロック・リストの大半を保持しているアプリケーションを識別する。
  - 64 ビット・システムでは (((保持されているロック * 64) / (locklist * 4096)) * 100)
  - 32 ビット・システムでは (((保持されているロック * 48) / (locklist * 4096)) * 100)

  これらのアプリケーションがロック・リストの多くを使用すると、ほかのアプリケーションでロック・エスカレーションを起こします。これらのアプリケーションでは行ロックの代わりに表ロックを使用する必要が生じることがありますが、表ロックが原因で **lock_waits** および **lock_wait_time** モニター・エレメント値が増える可能性があります。

lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント

グローバル・ロック・メモリー使用量が **cf_lock_sz** データベース構成パラメーターで指定した限度に達したために生じた、グローバル・ロックにおけるロック・エスカレーション数。

表 873. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 873. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 874. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このモニター・エレメントを、**lock_escals_maxlocks** モニター・エレメントと **lock_escals_locklist** モニター・エレメントと併せて使用して、データベースでエスカレーションの原因となっているロック・スペース構成パラメーターを判別します。

lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント

ローカル・ロック・メモリー使用量が、**locklist** データベース構成パラメーターで指定した限度に達したために生じたロック・エスカレーション数。

表 875. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 875. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 876. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される

表 876. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される

使用法

このモニター・エレメントを、**lock_escals_maxlocks** モニター・エレメントと **lock_escals_global** モニター・エレメントと併せて使用して、データベースでエスカレーションの原因となっているロック・スペース構成パラメーターを判別します。

lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント

ローカル・ロック・メモリー使用量が、**maxlocks** データベース構成パラメーターで指定した限度に達したために生じたロック・エスカレーション数。

表 877. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 877. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 878. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このモニター・エレメントを、**lock_escals_locklist** モニター・エレメントと **lock_escals_global** モニター・エレメントと併せて使用して、データベースでエスカレーションの原因となっているロック・スペース構成パラメーターを判別します。

lock_hold_count ロック保留カウント : モニター・エレメント

ロックに置かれている保留の数。保留は WITH HOLD 節に登録されたカーソルといくつかの DB2 ユーティリティーによってロック上に置かれます。保留があるロックはトランザクションがコミットされても保留解除されません。

表 879. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本

表 880. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

lock_list_in_use - 使用中のロック・リスト・メモリーの合計 : モニター・エレメント

使用中のロック・リスト・メモリーの合計量 (バイト単位)。

表 881. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法

このエレメントと **locklist** 構成パラメーターを組み合わせて使用すると、ロック・リスト使用率を計算できます。ロック・リスト使用率が高い場合は、そのパラメーターのサイズを増やすことを考慮してください。

注: 使用率を計算する場合、 **locklist** 構成パラメーターが各 4 KB のページ単位で割り振られるのに対し、このモニター・エレメントは結果をバイト単位で示すことに注意してください。

lock_mode - ロック・モード : モニター・エレメント

保持されているロックのタイプ。モードが不明な場合、このモニター・エレメントの値は NULL です。

表 882. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE

表 883. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	lock	ロック
ロック	lock_wait	ロック

表 884. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このモードは、リソースの競合の原因を判別するときに利用できます。

このエレメントは、調査するモニター情報のタイプにより、次の内容を示します。

- 1 つのアプリケーションがロック待ちをしているオブジェクトに対して、別のアプリケーションが保留しているロックのタイプ (アプリケーション・モニターおよびデッドロック・モニターのレベル)。
- このアプリケーションが保持しているオブジェクトのロック・タイプ (オブジェクト・ロック・レベル)。

このフィールドに可能な値は以下のとおりです。

モード	ロックのタイプ	API 定数
	ロックなし	SQLM_LNON
IS	意図的共有ロック	SQLM_LOIS
IX	意図的排他ロック	SQLM_LOIX
S	共有ロック	SQLM_LOOS
SIX	意図的排他ロックで共有	SQLM_LSIX
X	排他ロック	SQLM_LOOX
IN	Intent None	SQLM_LOIN
Z	超排他ロック	SQLM_LOOZ
U	更新ロック	SQLM_LOOU
NS	スキャン共有ロック	SQLM_LONS
NW	次キー弱排他ロック	SQLM_LONW

lock_mode_requested 要求されているロック・モード : モニター・エレメント

ロックを獲得するのを待機しているアプリケーションが要求したロックのモード。

表 885. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

表 886. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock_wait	ロック

表 887. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	event_dlconn	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

アプリケーションが要求したロックのモード。この値は、リソース競合の原因を判別するのに役立ちます。

可能性のあるロック・モードは、次に示す表でリストされています。

モード	ロックのタイプ	API 定数
	ロックなし	SQLM_LNON
IS	意図的共有ロック	SQLM_LOIS
IX	意図的排他ロック	SQLM_LOIX
S	共有ロック	SQLM_LOOS
SIX	意図的排他ロックで共有	SQLM_LSIX
X	排他ロック	SQLM_LOOX
IN	Intent None	SQLM_LOIN
Z	超排他ロック	SQLM_LOOZ
U	更新ロック	SQLM_LOOU
NS	スキャン共有ロック	SQLM_LONS

モード	ロックのタイプ	API 定数
NW	次キー弱排他ロック	SQLM_LONW

lock_name ロック名 : モニター・エレメント

内部バイナリー・ロック名。このエレメントはロックのユニーク ID を示します。

表 888. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE

表 889. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本
ロック	lock_wait	lock_wait

表 890. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

ルーチン MON_FORMAT_LOCK_NAME を使用して内部名をフォーマットし、ロックに関する詳細を取得することができます。例えば、これが表ロックである場合、ロックが参照している表および表スペースを取得することができます。

lock_node ロック・ノード

ロックに関係しているノード。

表 891. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント
デッドロック	event_dlconn	ステートメント
詳細付きデッドロック	event_detailed_dlconn	ステートメント

使用法 これはトラブルシューティングに使用できます。

lock_object_name ロック対象名

このエレメントは情報提供のみを目的としています。アプリケーションがロックを保留するオブジェクトの名前 (オブジェクト・ロック・レベルの情報)、またはアプリケーションがロックの取得を待機しているオブジェクトの名前 (アプリケーション・レベルおよびデッドロック・レベルの情報)。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 892. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	基本

表 893. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	lock	常に収集される
デッドロック	event_dlconn	常に収集される
詳細付きデッドロック	event_detailed_dlconn	常に収集される

使用法 表レベルのロックの場合、SMS および DMS 表スペースのファイル ID (FID)。行レベルのロックの場合のオブジェクト名は行 ID (RID)。表スペース・ロックの場合のオブジェクト名はブランク。バッファー・プール・ロックの場合のオブジェクト名は、バッファー・プールの名前。

ロックを保留する表を判別するときは、ファイル ID は固有のものとは限らないため、ファイル ID ではなく、*table_name* および *table_schema* を使用します。

ロックを保留している表スペースを判別するときは、*tablespace_name* を使用します。

lock_object_type - 待機中のロック対象タイプ : モニター・エレメント

アプリケーションがロックを保持しているオブジェクトのタイプ (オブジェクト・ロック・レベルの情報)、またはアプリケーションがロックの取得を待機しているオブジェクトのタイプ (アプリケーション・レベルおよびデッドロック・レベルの情報)。

表 894. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_FORMAT_LOCK_NAME 表関数 - 内部的なロック名をフォーマットし、詳細を戻す	ACTIVITY METRICS BASE
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE

表 895. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	基本
ロック	lock_wait	ロック

表 896. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントは、リソースの競合の原因を判別するときに役立ちます。

スナップショット・モニターおよびデッドロック[1]・イベント・モニター用に、オブジェクト・タイプ ID が sqlmon.h で定義されています。オブジェクトは、以下のいずれかのタイプです。

- 表スペース (sqlmon.h の SQLM_TABLESPACE_LOCK)

- 表
- バッファー・プール
- ブロック
- レコード (または行)
- データ・パーティション (sqlmon.h の SQLM_TABLE_PART_LOCK)
- 内部 (データベース・マネージャーが内部で保持するほかのタイプのロック)
- 自動サイズ変更
- 自動ストレージ。

ロック・イベント・モニターおよび表 1 のモニター表関数で、**lock_object_type** モニター・エレメントとして戻される値を表 4 に示します。

表 897. lock_object_type モニター・エレメントとして戻される値

戻される値	説明
TABLE	表ロック
ROW	行ロック
TABLESPACE	表スペース・ロック
EOT	end-of-table ロック
KEYVALUE	キー値ロック
SYSBOOT	SYSBOOT ロック
PLAN	プラン・ロック
VARIATION	バリエーション・ロック
SEQUENCE	シーケンス・ロック
BUFFERPOOL	バッファー・プール・ロック
LOB	LOB/LONG 領域ロック
CATALOG	カタログ・キャッシュ・ロック
ONLINE_BACKUP	オンライン・バックアップ・ロック
OBJECT_TABLE	オブジェクト表ロック
ALTER_TABLE	表変更ロック
DMS_SEQUENCE	DMS シーケンス・ロック
REORG	部分再編成ロック
MDC_BLOCK	MDC ブロック・ロック
TABLE_PARTITION	表パーティション・ロック
AUTORESIZE	自動サイズ変更ロック
AUTOSTORAGE	自動ストレージ・ロック
XMLPATH	XML パス・ロック
EXTENT_MOVEMENT	エクステント移動ロック
WORKLOAD	ワークロード許可ロック
FED_SERVER	フェデレーション・サーバー・ロック
FED_USER	フェデレーション・ユーザー・マッピング・ロック
CHUNK	チャンク・ロック
LOAD_PRE_PART	ロードの表事前パーティショニング・ロック

表 897. lock_object_type モニター・エレメントとして戻される値 (続き)

戻される値	説明
LOAD_PART	ロードの表パーティショニング・ロック
LOAD_TS	ロードの表スペース・ロック
LONG_FIELD_ESC	LONG フィールド・エスカレーション・ロック
LONG_FIELD_SPACE	LONG フィールド・バディー・スペース・ロック

lock_release_flags ロック保留解除フラグ : モニター・エレメント

ロック保留解除フラグ。

表 898. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE

表 899. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本
ロック	lock_wait	基本

表 900. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

以下の表に、すべての解除フラグ設定をリストします。各保留解除フラグは sqlmon.h で定義されているビット・フラグ値に基づいています。

API 定数	説明
SQLM_LOCKRELFLAGS_SQLCOMPILER	SQL コンパイラーによるロック
SQLM_LOCKRELFLAGS_UNTRACKED	非ユニーク、非追跡のロック

注: 割り当てられていないビットはすべてアプリケーション・カーソルに使用されます。

lock_status - ロック状況 : モニター・エレメント

ロックの内部状況を示します。

表 901. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE

表 902. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本

表 903. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	lock	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントは、アプリケーションがオブジェクトに対するロックを取得するために待機しているときに起きていることを明らかにするのに役立ちます。アプリケーションがすでに必要なオブジェクトのロックを取得しているように見える場合でも、同じオブジェクトについて異なるタイプのロックの取得を待たなければならないことがあります。

ロックの状況は、次のいずれかになります。

G 付与済みの状態 : アプリケーションは、**lock_mode** モニター・エレメントが指定する状態のロックを保有しています。

C 変換中の状態 : アプリケーションが保持ロックのタイプを変更しようとしています。例えば、共有ロックから排他ロックへの変更などです。

W 待機状態。

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル `sqlmon.h` を参照してください。

lock_timeout_val ロック・タイムアウト値 : モニター・エレメント

アプリケーションが SET CURRENT LOCK TIMEOUT ステートメントを発行した時点のタイムアウト値 (秒単位) を示します。ステートメントが実行されていない場合は、データベース・レベルのロック・タイムアウトが示されます。

表 904. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本
アプリケーション	agent	基本

表 905. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

使用法

SET CURRENT LOCK TIMEOUT ステートメントを使用して、アプリケーション・エージェントが表または索引のロックを待機する最大期間を指定できます。

アプリケーションがロックを待つ時間が長すぎる場合は、アプリケーション中で **lock_timeout_val** モニター・エレメント値を検査して、設定値が大きすぎるかどうか調べることができます。アプリケーション・ロジックにとって適切な場合は、アプリケーションに変更を加え、ロック・タイムアウト値を小さくして、アプリケーションをタイムアウトさせることができます。 SET CURRENT LOCK TIMEOUT ステートメントを使用して、この変更を行えます。

アプリケーションが頻繁にタイムアウトする場合は、ロック・タイムアウトの設定値が小さすぎるかどうかを検査し、該当する場合は大きくすることができます。

lock_timeouts - ロック・タイムアウト数 : モニター・エレメント

オブジェクトをロックするための要求が許可されずにタイムアウトになった回数。

表 906. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 906. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 907. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 908. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントは、 **locktimeout** データベース構成パラメーターの設定値を調整するときに利用できます。通常の操作レベルと比較して、ロックのタイムアウト回数が多くなった場合は、ロックを長期にわたって保有しているアプリケーションがある可能性があります。その場合、このエレメントは、ロックおよびデッドロックに関する他のいくつかのモニター・エレメントを分析して、アプリケーションに問題があるかどうかを判別する必要があることを示している場合があります。

locktimeout データベース構成パラメーターの設定値が高すぎると、ロックのタイムアウト回数が極端に少なくなります。この場合は、アプリケーションがロックを取得するための待機時間が長くなります。

lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント

ロックを保持しているアプリケーションがリモート・メンバー上にある場合のロック・タイムアウト数。

表 909. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 909. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 910. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される

表 910. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される

使用法

このエレメントを、**lock_timeouts** モニター・エレメントと併せて使用してください。**lock_timeouts_global** モニター・エレメントは、別のメンバーで保持されているロックを獲得するために待機中に生じたロック・タイムアウト数を示します。同じメンバーで保持されているロックを獲得するための待機中に生じたロック・タイムアウト数を判別するには、以下の数式を使用してください。

```
lock_timeouts - lock_timeouts_global
```

DB2 pureScale 環境以外では、この値は常時ゼロになります。

lock_wait_end_time - ロック待機終了タイム・スタンプ : モニター・エレメント

現在ロックされているオブジェクトのロックを取得するための待機をアプリケーションが停止した日時。

表 911. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

lock_wait_start_time - ロック待機開始タイム・スタンプ : モニター・エレメント

現在、別のアプリケーションによってロックされているオブジェクトに対するロックを取得するために、このアプリケーションが待機を開始した日時。

表 912. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

表 913. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック、タイム・スタンプ
ロック	lock_wait	ロック、タイム・スタンプ

表 914. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される

表 914. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック[1]	event_dlconn	タイム・スタンプ
詳細付きデッドロック[1]	event_detailed_dlconn	タイム・スタンプ

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントは、リソース競合の重大度を判別するときに役立ちます。

lock_wait_time - ロック待機中の時間 : モニター・エレメント

ロック待機に費やされた合計経過時間。値はミリ秒単位で示されます。

表 915. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 915. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 916. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ロック
アプリケーション	appl	ロック
ロック	appl_lock_list	appl_lock_list

スナップショット・モニターの場合、このカウンターはリセットできます。

表 917. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 917. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
トランザクション	event_xact	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

データベース・レベルでは、このデータベース内ですべてのアプリケーションが 1 つのロックを待機した合計経過時間を示します。この測定された経過時間には、アクティビティーの実行中に取得したロックに関して費やした時間と、コンパイルなどの他の処理中に取得したロックに関して費やした時間が含まれる場合があります。

アプリケーション接続およびトランザクションのレベルでは、この接続またはトランザクションがロックの付与を待機した合計経過時間を示します。

このエレメントの値に、現在もロック待機状態にあるエージェントのロック待機時間は含まれません。これにはすでにロック待機が完了したエージェントのロック待機時間のみが含まれます。

このエレメントと **lock_waits** モニター・エレメントを組み合わせて使用すると、平均ロック待機時間を計算できます。この計算は、データベース・レベルとアプリケーション接続レベルのいずれでも行えます。

経過時間を示すモニター・エレメントを使用するときは、次のことを考慮してください。

- 経過時間は、システム負荷の影響を受けるので、実行する処理数が多くなると、この経過時間の値は大きくなる。
- このエレメントをデータベース・レベルで計算する場合、データベース・システム・モニターはアプリケーション・レベルの時間を合計する。この場合、同時に複数のアプリケーション処理が実行されていることがあるので、データベース・レベルでは時間が二重に計算されます。

  意味のあるデータを提供するためには、上記の説明に従って平均ロック待機時間を計算してください。

lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント

グローバルなロック待機時間。この時間の測定単位はミリ秒です。

表 918. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 918. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 919. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このモニター・エレメントを **lock_wait_time** モニター・エレメント (ロックに費やされた合計待機時間を示します) と併せて使用します。**lock_wait_time_global** モニター・エレメントは、別のメンバー上の競合アプリケーションによって保持されているロックの待機時間を示します。同じメンバー上の競合アプリケーションによって保持されているロックの合計待機時間を判別するには、以下の数式を使用してください。

```
lock_wait_time - lock_wait_time_global
```

DB2 pureScale 環境以外では、この値は常時ゼロになります。

lock_wait_time_global_top - 最長グローバル・ロック待機時間 : モニター・エレメント

別のメンバーで保持されているロックに関して生じた最長ロック待機。この値はミリ秒単位で報告されます。

表 920. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	常に収集される

lock_wait_time_top - ロック待機時間の最上位 : モニター・エレメント

ワークロードでの任意の要求のロック待機時間の最高水準点。単位はミリ秒です。lock_wait_time_top 最高水準点はワークロードに対して常に収集されます。要求メトリックが使用可能になっている場合にのみ、要求はこの最高水準点に寄与します。

表 921. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	常に収集される

使用法

このエレメントを使用して、収集された時間間隔におけるワークロード用のパーティションでの要求の最大ロック待機時間を判別することができます。

lock_wait_val - ロック待機値のモニター・エレメント

mon_lockwait のイベントが生成されるまでのロック待機時間 (ミリ秒単位)。

表 922. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

lock_waits - ロック待機数 : モニター・エレメント

アプリケーションまたは接続がロックを待機した合計回数。

表 923. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 923. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 924. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 925. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

データベース・レベルでは、アプリケーションがデータベース内でロックを待機した合計回数を示します。

アプリケーション接続レベルでは、この接続がロックを要求し、ほかの接続がデータ上でロックを保留していたために待機した合計回数を示します。

このエレメントと **lock_wait_time** を組み合わせて使用すると、データベース・レベルの場合は平均ロック待機時間を計算できます。この計算は、データベース・レベルとアプリケーション接続レベルのいずれでも行えます。

平均ロック待機時間が長い場合は、多数のロックを保留するアプリケーションまたはロック・エスカレーションを起こしているアプリケーションを探します。これにより、必要に応じてアプリケーションをエスカレーションして並行性を改善します。エスカレーションが原因で平均ロック待機時間が長くなっている場合は、**locklist** および **maxlocks** 構成パラメーターのどちらか、または両方の設定値が低すぎることが原因と考えられます。

lock_waits_global - グローバル・ロック待機 : モニター・エレメント

リモート・メンバー上にあるロックをアプリケーションが保持しているために生じたロック待機数。

表 926. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 926. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 927. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このモニター・エレメントを、**lock_waits** モニター・エレメント (すべてのメンバー上の競合アプリケーションによって保持されているロックが原因で生じたロック待機合計数をレポートします) と一緒に使用してください。**lock_waits_global** モニター・エレメントは、別のメンバー上の競合アプリケーションで保持されていたロック待機回数を示します。待機アプリケーションと同じメンバー上の競合アプリケーションによって保持されているロック待機数を判別するには、以下の数式を使用します。

```
lock_waits - lock_waits_global
```

DB2 pureScale 環境以外では、この値は常時ゼロになります。

locks_held - ロック保持数 : モニター・エレメント

現在保持されているロックの数。

表 928. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 928. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 929. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
ロック	db_lock_list	基本
ロック	appl_lock_list	基本

表 930. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック	event_detailed_dlconn	常に収集される

使用法

モニター情報がデータベース・レベルの場合は、データベース内のすべてのアプリケーションが現在保持しているロックの合計数を示します。

モニター情報がアプリケーション・レベルの場合は、アプリケーションのすべてのエージェントが現在保持しているロックの合計数を示します。

locks_held_top - ロック保持最大数 : モニター・エレメント

このトランザクション中に保持されたロックの最大数。

表 931. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	常に収集される

使用法

このエレメントを使用すると、 **maxlocks** 構成パラメーターで定義されている、アプリケーションが使用可能な最大ロック数に近づいているかどうかを判別できます。このパラメーターは、ロック・エスカレーションを起こさずに各アプリケーションが使用できるロック・リストのパーセンテージを示します。ロック・エスカレーションが起こると、データベースに接続されている各アプリケーション間の並行性が低下します。

maxlocks パラメーターがパーセンテージで指定されるのに対して、このエレメントはカウンターであるため、以下のいずれかの公式を使って計算することにより、このエレメントで得られるカウントと、1 つのアプリケーションで保持できる合計ロック数とを比較できます。

- 64 ビット・システムでは (locklist * 4096 / 64) * (maxlocks / 100)
- 32 ビット・システムでは (locklist * 4096 / 48) * (maxlocks / 100)

ロック数が多い場合は、アプリケーション内で実行するコミットの数を多くして、一部のロックを解放する必要があります。

locks_in_list 報告されたロックの回数

イベント・モニターの報告対象の特定のアプリケーションが保留するロック数。

表 932. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック	event_detailed_dlconn	常に収集される

locks_waiting - ロックで待機中の現行エージェント : モニター・エレメント

ロック待機中のエージェントの数を示します。

表 933. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
ロック	db_lock_list	基本

使用法

このエレメントと **appls_cur_cons** と組み合わせて使用すると、ロックを待機中のアプリケーションのパーセンテージが分かります。この値が大きい場合は、アプリケーションに並行性の問題がある可能性があるため、ロックや排他ロックを長時間にわたって保留しているアプリケーションを確認する必要があります。

log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント

ログ・バッファー内のスペースの待機にエージェントが費やした時間。値はミリ秒単位で示されます。

表 934. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 934. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 935. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント

ログ・レコードがディスクにフラッシュされるのをエージェントが待機していた時間。値はミリ秒単位で示されます。

表 936. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 936. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 937. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE

表 937. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント

ログ・データがディスクに書き込まれるのをエージェントが待機しなければならなかった回数。

表 938. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 938. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 939. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

log_held_by_dirty_pages ダーティー・ページによって占有されるログ・スペースの量

データベース中の最も古いダーティー・ページと、アクティブ・ログの先頭との間の差に対応するログの量 (バイト単位)。

エレメント ID

log_held_by_dirty_pages

エレメント・タイプ

水準点

表 940. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 941. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 942. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 この値は、スナップショットをとる際に、そのスナップショットの時点の条件に基づいて計算されます。

このエレメントは、バッファー・プール中の最も古いページに関するページ・クリーニングの有効性を評価するのに使用してください。

バッファー・プール中の古いページのクリーニングは、 *softmax* データベース構成パラメーターによって管理されます。ページ・クリーニングが有効な場合は、 *log_held_by_dirty_pages* がほぼ次の値以下である必要があります。

```
(softmax / 100)  *  logfilsiz * 4096
```

このステートメントが真でない場合は、ページ・クリーナー数 (*num_iocleaners*) 構成パラメーターを大きくしてください。

この条件が真で、ダーティー・ページに保持されるログを少なくすることが必要な場合は、 *softmax* 構成パラメーターを小さくしてください。

log_read_time ログ読み取り時間

ロガーがログ・データをディスクから読み取るのに要した合計経過時間。表に書き込むイベント・モニターの場合、このエレメントの値は、BIGINT データ・タイプを使用して、マイクロ秒単位で示されます。

表 943. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 944. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 945. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *log_reads*、 *num_log_read_io*、および *num_log_data_found_in_buffer* エレメントを組み合わせて使用すると、次のことを判別できます。

- 現行ディスクがロギングに適しているかどうか。
- ログ・バッファー・サイズが適切かどうか。

log_reads 読み取られたログ・ページの数

ロガーがディスクから読み取ったログ・ページの数。

エレメント ID

log_reads

エレメント・タイプ

カウンター

表 946. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 947. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 948. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントとオペレーティング・システム・モニターを組み合わせて使用すると、データベース・アクティビティーによって発生した装置上の入出力の量がわかります。

log_to_redo_for_recovery リカバリーの場合に再実行されるログの量

クラッシュ・リカバリーの場合に再実行する必要のあるログの量 (バイト単位)。

エレメント ID

log_to_redo_for_recovery

エレメント・タイプ

水準点

表 949. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 950. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 951. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 この値は、スナップショットをとる際に、そのスナップショットの時点の条件に基づいて計算されます。値が大きいほど、システムのクラッシュ後のリカバリー時間が長くなることを示します。値が大きすぎるように思える場合は、 *log_held_by_dirty_pages* モニター・エレメントを検査して、ページ・クリーニングを調整する必要があるかどうか調べてください。また、終了する必要のある長期実行トランザクションがあるかどうかも検査してください。

log_write_time ログ書き込み時間

ロガーがログ・データをディスクに書き込むのに要した合計経過時間。表に書き込むイベント・モニターの場合、このエレメントの値は、BIGINT データ・タイプを使用して、マイクロ秒単位で示されます。

表 952. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 953. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 954. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *log_writes* および *num_log_write_io* エレメントを組み合わせて使用すると、現行ディスクがロギングに適しているかどうかを判別できます。

log_writes 書き込まれたログ・ページの数

ロガーがディスクに書き込んだログ・ページの数。

エレメント ID
log_writes

エレメント・タイプ
カウンター

表 955. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 956. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 957. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントとオペレーティング・システム・モニターを組み合わせて使用すると、データベース・アクティビティーによって発生した装置上の入出力の量がわかります。

注: ログ・ページがディスクに書き込まれる際、最後のページが満杯になっていない場合があります。その場合、部分的なログ・ページがログ・バッファー内に残り、さらにログ・レコードがページに書き込まれます。その結果、ログ・ページは、ロガーによってディスクに複数回書き込まれることがあります。このエレメントからは、DB2 が生成したページ数は測定できません。

long_object_pages 長いオブジェクト・ページ数

表中の LONG データが使用するディスク・ページの数。

表 958. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 959. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法 このエレメントを使用すると、特定の表中の LONG データが使用する実際

のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに LONG データが大きくなる比率を追跡できます。

long_object_l_pages - 長いオブジェクト・データ論理ページ : モニター・エレメント

この表に含まれている長いデータによって使用された、ディスク上の論理ページの数。

表 960. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

- この値は、オブジェクトに物理的に割り振られているスペース量より小さい場合があります。そのようになる可能性があるのは、TRUNCATE ステートメントの REUSE STORAGE オプションを使用した場合です。このオプションを指定すると、表に割り振られているストレージは引き続き割り振られますが、ストレージは空と見なされます。さらに、このモニター・エレメントの値は、オブジェクトに論理的に割り振られているスペース量よりも小さくなる可能性があります。その理由は、論理的に割り振られているスペースの合計には、量的にはわずかですが追加のメタデータが含まれているからです。

  オブジェクトの正確な論理サイズまたは物理サイズを取得するには、ADMIN_GET_TAB_INFO_V97 関数を使用します。この関数を使用すると、このモニター・エレメントに関して報告されるページ数とページ・サイズの積で得られる値よりも正確なオブジェクト・サイズに関する情報が得られます。

long_tbsp_id - LONG 表スペース ID : モニター・エレメント

この表で、LONG データ (LONG または LOB タイプの列) を保持する表スペースの ID。

表 961. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

このエレメントの値は、SYSCAT.TABLESPACES のビューの TBSPACEID 列の値と一致します。

machine_identification - ホストのハードウェア ID のモニター・エレメント

プロセッサー・アーキテクチャーを示すストリング。例えば、「x86 64 bit」などです。この ID に戻される値は、ホスト上で実行されているオペレーティング・システムによって決まります。

表 962. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

max_agent_overflows 最大エージェント・オーバーフロー回数

最大エージェント数 (**maxagents**) 構成パラメーターにすでに達しているときに、新規エージェント作成要求を受信した回数。

注: max_agent_overflows モニター・エレメントは、DB2 バージョン 9.5 以降では推奨されません。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 963. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

maxagents 構成パラメーターに達してもエージェント作成要求をまだ受け取る場合は、このノードのワークロードが高すぎることを示している可能性があります。

max_coord_stmt_exec_time - コーディネーターの最大ステートメント実行時間のモニター・エレメント

ステートメントの 1 回の実行にかかった最大コーディネーター実行時間 (ミリ秒)。非コーディネーター・ノード上や、ステートメントが実行されたことがない場合、この値はゼロです。

表 964. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 965. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	pkgcache	ACTIVITY METRICS BASE

max_coord_stmt_exec_time_args - コーディネーターの最大ステートメント実行時間の引数のモニター・エレメント

コーディネーター・メンバー上でステートメントの単一実行の最大実行時間 (**max_coord_stmt_exec_time**) を記録したときに、そのステートメントに与えられていた入力引数を表す XML 文書。

まだ実行されたことがないステートメントや、入力引数を持たないステートメントの場合、この列は NULL です。この文書には、 **max_coord_stmt_exec_time_arg** という名前のエレメント 1 つ以上で構成される、 **max_coord_stmt_exec_time_args** という名前の親エレメントが 1 つ入っています。この XML 文書の構造の例については、 1178 ページの図 19 を参照してください。

表 966. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 967. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	pkgcache_stmt_args	ACTIVITY METRICS BASE

使用法

この文書の内容は、 XMLPARSE スカラー関数を使用して表示することができます。例えば、

```
SELECT XMLPARSE(DOCUMENT MAX_COORD_STMT_EXEC_STMT_ARGS)
  FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL, NULL, NULL, -2));
```

1178 ページの図 19 に、上記のステートメントから戻される XML 文書の内容の例を示します。

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <max_coord_stmt_exec_time_args
  xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="10010000">
   <max_coord_stmt_exec_time_arg>
      <stmt_value_index>1</stmt_value_index>
      <stmt_value_isreopt id="0">no</stmt_value_isreopt>
      <stmt_value_isnull id="0">no</stmt_value_isnull>
      <stmt_value_type>INTEGER</stmt_value_type>
      <stmt_value_data>5</stmt_value_data>
   </max_coord_stmt_exec_time_arg>
   <max_coord_stmt_exec_time_arg>
      <stmt_value_index>2</stmt_value_index>
      <stmt_value_isreopt id="0">no</stmt_value_isreopt>
       <stmt_value_isnull id="0">no</stmt_value_isnull>
       <stmt_value_type>VARCHAR</stmt_value_type>
       <stmt_value_data>78</stmt_value_data>
   </max_coord_stmt_exec_time_arg>
   <max_coord_stmt_exec_time_arg>
      <stmt_value_index>3</stmt_value_index>
      <stmt_value_isreopt id="0">no</stmt_value_isreopt>
       <stmt_value_isnull id="0">no</stmt_value_isnull>
      <stmt_value_type>VARCHAR</stmt_value_type>
      <stmt_value_data>john</stmt_value_data>
   </max_coord_stmt_exec_time_arg>
   <max_coord_stmt_exec_time_arg>
      <stmt_value_index>4</stmt_value_index>
      <stmt_value_isreopt id="0">no</stmt_value_isreopt>
       <stmt_value_isnull id="0">no</stmt_value_isnull>
      <stmt_value_type>VARCHAR</stmt_value_type>
      <stmt_value_data>x</stmt_value_data>
   </max_coord_stmt_exec_time_arg>
   <max_coord_stmt_exec_time_arg>
     <stmt_value_index>5</stmt_value_index>
     <stmt_value_isreopt id="0">no</stmt_value_isreopt>
     <stmt_value_isnull id="0">no</stmt_value_isnull>
     <stmt_value_type>DATE</stmt_value_type>
     <stmt_value_data>2001-02-12</stmt_value_data>

⋮

   <max_coord_stmt_exec_time_arg>
      <stmt_value_index>15</stmt_value_index>
      <stmt_value_isreopt id="0">no</stmt_value_isreopt>
      <stmt_value_isnull id="0">no</stmt_value_isnull>
      <stmt_value_type>DECIMAL</stmt_value_type>
      <stmt_value_data>+0002000.55</stmt_value_data>
    </max_coord_stmt_exec_time_arg>
  </max_coord_stmt_exec_time_args>
```

図 19. **max_coord_stmt_exec_time_args** *の内容例*： この例では、ステートメントに渡された 15 個の引数が文書に示されています。

以下のデータ・タイプの項目はこの XML 文書に記録されますが、これらのタイプの引数の実際の値は STMT_VALUE_DATA エレメントにキャプチャーされません。

- BLOB
- CLOB
- REF
- BOOLEAN
- 構造化データ・タイプ

- DATALINK
- LONG VARGRAPHIC
- LONG VARCHAR
- XML タイプ
- DBCLOB
- ARRAY タイプ
- ROW タイプ
- ROWID
- CURSOR 変数

入力引数は、ステートメント内に最初に出現するものが最初に記録され、その後に出現するものが続いて記録されます。記録できる入力パラメーターの数は、XML 文書を含んでいる BLOB 文書のサイズの上限によってのみ、制約されます。実際には、収集される入力引数の数がこの上限に達することはまずないでしょう。

このエレメントを含む XML 文書のスキーマは、パス `sqllib/misc/DB2EvmonPkgCache.xsd` にあります。

max_coord_stmt_exec_timestamp - コーディネーターの最大ステートメント実行のタイム・スタンプのモニター・エレメント

max_coord_stmt_exec_time 値を作成したステートメントが実行を開始した時刻。

表 968. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 969. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	pkgcache	ACTIVITY METRICS BASE

max_data_received_1024 受信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 513 から 1024 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_1024

エレメント・タイプ

カウンター

表 970. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_128 受信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 1 から 128 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_128

エレメント・タイプ

カウンター

表 971. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_16384 受信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 8193 から 16384 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_16384

エレメント・タイプ

カウンター

表 972. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_2048 受信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 1025 から 2048 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_2048

エレメント・タイプ

カウンター

表 973. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_256 受信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 129 から 256 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_256

エレメント・タイプ

カウンター

表 974. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_31999 受信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数 : モニター・エレメント

このエレメントは、受信アウトバウンド・バイト数が 16385 から 31999 バイトだったステートメントまたはチェーンの数を示します。

表 975. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_4096 受信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 2049 から 4096 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_4096

エレメント・タイプ

カウンター

表 976. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_512 受信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 257 から 512 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_512

エレメント・タイプ

カウンター

表 977. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_64000 受信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数 : モニター・エレメント

このエレメントは、受信アウトバウンド・バイト数が 32000 から 64000 バイトだったステートメントまたはチェーンの数を示します。

表 978. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_8192 受信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 4097 から 8192 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_received_8192

エレメント・タイプ
カウンター

表 979. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_gt64000 受信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数

このエレメントは、受信アウトバウンド・バイト数が 64000 バイトを超えたステートメントまたはチェーンの数を示します。

エレメント ID
max_data_received_gt64000

エレメント・タイプ
カウンター

表 980. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_1024 送信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 513 から 1024 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_1024

エレメント・タイプ
カウンター

表 981. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_128 送信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 1 から 128 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_128

エレメント・タイプ
カウンター

表 982. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_16384 送信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 8193 から 16384 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_16384

エレメント・タイプ
カウンター

表 983. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_2048 送信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 1025 から 2048 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_2048

エレメント・タイプ
カウンター

表 984. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_256 送信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 129 から 256 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_256

エレメント・タイプ
カウンター

表 985. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_31999 送信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 16385 から 31999 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_31999

エレメント・タイプ
カウンター

表 986. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_4096 送信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 2049 から 4096 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_4096

エレメント・タイプ
カウンター

表 987. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_512 送信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 257 から 512 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_512

エレメント・タイプ
カウンター

表 988. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_64000 送信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 32000 から 64000 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_64000

エレメント・タイプ
カウンター

表 989. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_8192 送信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 4097 から 8192 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_8192

エレメント・タイプ
カウンター

表 990. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_gt64000 送信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数

このエレメントは、送信アウトバウンド・バイト数が 64000 バイトを超えたステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_gt64000

エレメント・タイプ
カウンター

表 991. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_100_ms ネットワーク時間が 16 から 100 ミリ秒のステートメント数

このエレメントは、ネットワーク時間が 16 ミリ秒を超えて 100 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

表 992. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_16_ms ネットワーク時間が 4 から 16 ミリ秒のステートメント数

このエレメントは、ネットワーク時間が 4 ミリ秒を超えて 16 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

エレメント ID

max_network_time_16_ms

エレメント・タイプ

カウンター

表 993. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_1_ms ネットワーク時間が 1 ミリ秒以下のステートメント数

このエレメントは、ネットワーク時間が 1 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

表 994. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_4_ms ネットワーク時間が 1 から 4 ミリ秒のステートメント数

このエレメントは、ネットワーク時間が 1 ミリ秒を超えて 4 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

エレメント ID

max_network_time_4_ms

エレメント・タイプ

カウンター

表 995. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_500_ms ネットワーク時間が 100 から 500 ミリ秒のステートメント数

このエレメントは、ネットワーク時間が 100 ミリ秒を超えて 500 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

表 996. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_gt500_ms ネットワーク時間が 500 ミリ秒を超えるステートメント数

このエレメントは、ネットワーク時間が 500 ミリ秒を超えたステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

表 997. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

member - データベース・メンバー・モニター・エレメント

この結果レコードのデータの取得元となったデータベース・メンバーの数値 ID。

表 998. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
ADMIN_GET_MEM_USAGE 表関数 - インスタンスの合計メモリー消費量の取得	ACTIVITY METRICS BASE
AUDIT_ARCHIVE プロシージャーおよび表関数 - 監査ログ・ファイルのアーカイブ	ACTIVITY METRICS BASE
DBCFG 管理ビューおよび DB_GET_CFG 表関数 - データベース構成パラメーター情報の取得	ACTIVITY METRICS BASE
ENV_GET_REG_VARIABLES 表関数 - 使用中の DB2 レジストリー設定の取得	ACTIVITY METRICS BASE
ENV_GET_DB2_SYSTEM_RESOURCES 表関数 - DB2(r) システム情報を返す	ACTIVITY METRICS BASE
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_AUTO_MAINT_QUEUE 表関数 - 自動保守ジョブの情報の取得	ACTIVITY METRICS BASE
MON_GET_AUTO_RUNSTATS_QUEUE 表関数 - 評価用にキューに入れられたオブジェクトの情報の取得	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CF_WAIT_TIME 表関数 - クラスター・キャッシング・ファシリティーのコマンド待機時間の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE

表 998. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_EXTENDED_LATCH_WAIT 表関数 - ラッチ情報を戻す	ACTIVITY METRICS BASE
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続に関する詳細の取得	ACTIVITY METRICS BASE
MON_GET_GROUP_BUFFERPOOL 表関数 - グループ・バッファー・プール・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される

表 998. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_RTS_RQST 表関数 - リアルタイム統計要求の情報の取得	ACTIVITY METRICS BASE
MON_GET_SERVERLIST 表関数 - メンバーの優先順位の詳細を取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得	ACTIVITY METRICS BASE
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE
PDLOGMSGS_LAST24HOURS 管理ビューおよび PD_GET_LOG_MSGS 表関数 - 問題診断メッセージの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE

表 998. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 999. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	-	COLLECT BASE DATA
ロッキング	-	常に収集される
変更履歴	changesummary dbdbmcfg regvar ddlstmtexec txncompletion evmonstart utilstart utillocation utilstop	常に収集される

使用法

DB2 メンバーは、単一ホスト上で DB2 サーバー・ソフトウェアを実行するデータベース・マネージャー・インスタンスであり、DB2 メンバーはそれに接続するアプ

リケーションからのデータベース要求を受け入れて処理します。

memory_free - 物理メモリーの空き容量のモニター・エレメント

実行中のプロセスに割り振られていない、このホスト上の物理メモリーの総量 (MB 単位)。

表 1000. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

memory_pool_used_hwm - メモリー・プール最高水準点 : モニター・エレメント

作成以来このプールに割り当てられた最高メモリー量 (KB 単位)。

表 1001. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_POOL 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

memory_pool_id - メモリー・プール ID : モニター・エレメント

メモリー・プール ID

表 1002. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	常に収集される

memory_pool_type - メモリー・プール名 : モニター・エレメント

メモリー・プールの名前。

表 1003. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE

使用法

memory_pool_type エレメントを使用して、メモリー・プールのタイプを識別します。このモニター・エレメントによって返される可能性のある値が 1198 ページの表 1004 でリストされています。

表 1004. memory_pool_type に返される可能性のある値：

メモリー・プール名 *	説明	追加情報
APM	エージェント・プール管理 (APM) ヒープ	内部メモリー・プール
APPL_SHARED	アプリケーション共有ヒープ	内部メモリー・プール
APPLICATION	アプリケーション・ヒープ	applheapsz - アプリケーション・ヒープ・サイズ構成パラメーターを参照してください。
APS	APS ヒープ	内部メモリー・プール
BSU_CF	基本サービス・ユーティリティー (BSU) CF ヒープ	内部メモリー・プール
BSU	基本サービス・ユーティリティー (BSU) ヒープ	内部メモリー・プール
BP	バッファー・プール・ヒープ	CREATE BUFFERPOOL ステートメントを参照してください。
CAT_CACHE	カタログ・キャッシュ・ヒープ	catalogcache_sz - カタログ・キャッシュ・サイズ構成パラメーターを参照してください。
DATABASE_CF	データベース CF ヒープ	内部メモリー・プール
DATABASE	データベース・ヒープ	dbheap - データベース・ヒープ構成パラメーターを参照してください。
DEBUG	デバッグ・ヒープ	内部メモリー・プール
DROP_INDEX	索引ドロップ・ヒープ	内部メモリー・プール
EDU	エンジン・ディスパッチ可能単位 (EDU) ヒープ	内部メモリー・プール
FCMBP	高速コミュニケーション・マネージャー (FCM) バッファー・ヒープ	fcm_num_buffers - FCM バッファー数構成パラメーターを参照してください。
FCM_CHANNEL	FCM チャネル・ヒープ	fcm_num_channels - FCM チャネル数構成パラメーター を参照してください。
FCM_CONTROL	FCM 制御ヒープ	内部メモリー・プール
FCM_LOCAL	FCM ローカル・ヒープ	内部メモリー・プール
FCM_SESSION	FCM セッション・ヒープ	内部メモリー・プール
FEDERATED	フェデレーテッド・ヒープ	内部メモリー・プール
KERNEL _CONTROL	カーネル制御ブロック・ヒープ	内部メモリー・プール
KERNEL	カーネル・ヒープ	内部メモリー・プール
LOCK_MGR	ロック・マネージャー・ヒープ	locklist - ロック・リスト用最大ストレージ構成パラメーターを参照してください。
MISC	各種ヒープ	DB2_FMP_COMM_HEAPSZ レジストリー変数を参照してください。

表 1004. memory_pool_type に返される可能性のある値 (続き):

メモリー・プール名 *	説明	追加情報
MONITOR	モニター・ヒープ	mon_heap_sz - データベース・システム・モニター・ヒープ・サイズ構成パラメーターを参照してください。
OPTPROF _PARSER	OptProf XML パーサー・ヒープ	内部メモリー・プール
OSS_TRACKER	OSS リソース・トラッキング・ヒープ	内部メモリー・プール
PERSISTENT _PRIVATE	永続専用ヒープ	内部メモリー・プール
PACKAGE _CACHE	パッケージ・キャッシュ・ヒープ	pckcachesz - パッケージ・キャッシュ・サイズ構成パラメーターを参照してください。
PRIVATE	専用	内部メモリー・プール
RESYNC	再同期ヒープ	内部メモリー・プール
SORT	専用ソート・ヒープ	sortheap - ソート・ヒープ・サイズ構成パラメーターを参照してください。
SHARED_SORT	共有ソート・ヒープ	sheapthres_shr - 共有ソートのソート・ヒープのしきい値構成パラメーターを参照してください。
SQL_COMPILER	SQL コンパイラー・ヒープ	内部メモリー・プール
STATEMENT	ステートメント・ヒープ	stmtheap - ステートメント・ヒープ・サイズ構成パラメーターを参照してください。
STATISTICS	統計ヒープ	stat_heap_sz - 統計ヒープ・サイズ構成パラメーターを参照してください。
USER_DATA	ユーザー・データ・ヒープ	内部メモリー・プール
UTILITY	ユーティリティー・ヒープ	util_heap_sz - ユーティリティー・ヒープ・サイズ構成パラメーターを参照してください。
XMLCACHE	XML キャッシュ・ヒープ	内部メモリー・プール
XMLPARSER	XML パーサー・ヒープ	内部メモリー・プール

* これらのプールに関して **db2pd** によって返される名前は省略されることがあります。プールの名前および **db2pd** によって使用されるすべての省略形は sqlopoolinfo.h で定義されています。

memory_pool_used - 使用中のメモリー・プールの量 : モニター・エレメント

このメモリー・プールで使用されているコミット済みのメモリーの量 (KB 単位)。

表 1005. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE

memory_set_committed - 現在コミット済みのメモリー : モニター・エレメント

このメモリー・セットに対する現在コミット済みのメモリーの量 (KB 単位)。

表 1006. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

使用法

コミット済みのメモリーとは、システム上で RAM またはページング・スペース、あるいはその両方に格納されるページのことです。

memory_set_id - メモリー・セット ID : モニター・エレメント

特定のメモリー・セット・タイプにマップする数値 ID。

表 1007. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

memory_set_size - メモリー・セット・サイズ : モニター・エレメント

最大メモリー・コミットメント制限 (KB)。

この値は、メモリー・セットの構成済み設定値か、自動的に管理されるそれらのメモリー・セットの内部計算値のいずれかを表します。

表 1008. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

memory_set_type - メモリー・セット・タイプ : モニター・エレメント

メモリー・セットのタイプ。

表 1009. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

使用法

このモニター・エレメントに対して返される値が表 1010 で説明されています。

表 1010. **memory_set_type** *として可能な値*

メモリー・セット・タイプ	説明	有効範囲
DBMS	データベース・マネージャー・メモリー・セット	インスタンス
FMP	fenced モード・プロセスのメモリー・セット	インスタンス
PRIVATE	専用メモリー・セット	インスタンス
DATABASE	データベース・メモリー・セット	データベース
APPLICATION	アプリケーション・メモリー・セット	データベース
FCM	高速コミュニケーション・マネージャー (FCM) メモリー・セット	インスタンス、ホスト

memory_set_used - このセットによって使用されているメモリー : モニター・エレメント

このセットからメモリー・プールに割り当てられたメモリーの量 (KB 単位)。

表 1011. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

使用法

このモニター・エレメントによって表されるメモリーはすべて、コミット済みのメモリーです。このモニター・エレメントに対して返される値は、MEMORY_SET_COMMITTED に組み込まれます。コミットされたが、使用されて

いない追加メモリーはすべて、パフォーマンスを向上させるためにキャッシュに入れられます。

memory_set_used_hwm - メモリー・セット最高水準点 : モニター・エレメント

メモリー・セット作成以来このセットからメモリー・プールに対して割り当てられた最高メモリー量 (KB 単位)。

表 1012. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

memory_swap_free - フリーのスワップ・スペース合計 : モニター・エレメント

このホスト上の未使用スワップ・スペースの総量 (MB 単位)。

表 1013. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

memory_swap_total - スワップ・スペース合計 : モニター・エレメント

このホスト上のスワップ・スペースの総量 (MB 単位)。

表 1014. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

memory_total - 合計物理メモリーのモニター・エレメント

このホスト上の物理メモリーの総量 (MB 単位)。

表 1015. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

message コントロール表メッセージ

MESSAGE_TIME 列内のタイム・スタンプの性質。このエレメントは、表書き込みイベント・モニターによってコントロール表でのみ使用されます。

表 1016. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
-	-	常に収集される

使用法

値は以下のいずれかです。

DROPPED RECORDS: *n*
MONHEAP を割り振れなかったためにドロップされたアクティビティー・レコードの数。

FIRST_CONNECT
活動化後のデータベースへの最初の接続の時刻。

EVMON_START
EVMONNAME 列にリストされているイベント・モニターが開始された時刻。

OVERFLOWS: *n*
バッファー・オーバーフローのために *n* 個のレコードが廃棄されたことを示します。

LAST DROPPED RECORD
アクティビティー・レコードが最後にドロップされた時刻。

message_time タイム・スタンプ・コントロール表メッセージ

MESSAGE 列に記述されているイベントに対応したタイム・スタンプ。このエレメントは、表書き込みイベント・モニターによってコントロール表でのみ使用されます。

表 1017. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
-	-	常に収集される

metrics - メトリック : モニター・エレメント

統計イベント・モニターによって収集されたシステム・モニター・エレメントの一部が入った XML 文書。

表 1018. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
統計	EVENTS_SCSTATS	常に収集される
統計	EVENT_WLSTATS	常に収集される

使用法

返される XML 文書のスキーマは、ファイル sqllib/misc/DB2MonCommon.xsd に用意されています。最上位エレメントは **system_metrics** です。

このモニター・エレメントと **details_xml** モニター・エレメントに関連付けられている各 XML 文書には、同じシステム・メトリックが入っていますが、重要な相違点が 1 つあります。このモニター・エレメントのメトリックは、統計が最後に収集された時からの各メトリックの値の変化を示すために計算された値です。一方、**details_xml** のメトリックは一般的には 0 から始まり、次にデータベースが活動化されるときまで累算し続けます。

論理データ・グループ EVENT_SCMETRICS および EVENT_WLMETRICS を使用すると、 **metrics** に含まれているモニター・エレメントを個別のエレメントとして直接確認できます。例えば、統計イベント・モニターにより表への書き込みが行われる場合、 SQL 照会を使用して新規論理データ・グループからデータを取得するという方法でメトリックにアクセスできます。この場合、**metrics** モニター・エレメントの XML 文書を後処理する必要も解析する必要もありません。

重要: バージョン 10.1 フィックスパック 1 以降、**details_xml** モニター・エレメントは統計イベント・モニターでは推奨されなくなりました。今後のリリースでは除去される可能性があります。**details_xml** に返される XML メトリック・データを使用している場合は、代わりにこのモニター・エレメントを使用してください。

mon_interval_id - モニター間隔 ID のモニター・エレメント

特定のトランザクションが完了したときの MON_INTERVAL_ID データベース・グローバル変数の値。

表 1019. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

表 1020. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	常に収集される
統計	event_scstats	常に収集される
統計	event_histogrambin	常に収集される
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される
アクティビティー	event_activity	常に収集される
作業単位	uow	常に収集される

nesting_level - ネスト・レベル : モニター・エレメント

このエレメントは、ステートメントが実行されていた間にそのステートメントに対して有効だった、ネストまたは再帰のレベルを示します。ネストの各レベルは、ストアード・プロシージャーまたはユーザー定義関数 (UDF) のネストされた呼び出しまたは再帰的呼び出しに対応します。

nesting_level モニター・エレメントは、**stmt_nest_level** モニター・エレメントの別名です。

表 1021. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1022. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
アクティビティー	event_activitystmt	-
作業単位	パッケージ・リストに報告されます。	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを **stmt_invocation_id** モニター・エレメントと一緒に使用して、特定の SQL ステートメントが実行された呼び出しを一意に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

network_time_bottom ステートメントの最小ネットワーク時間

このエレメントは、 DCS データベースに対してまたはこの DCS アプリケーション内で実行されたステートメントについて、あるいはこの回数のデータ伝送を使用したステートメントについて、最小ネットワーク時間を示します (ネットワーク時間は、1 つのステートメントに対するホスト応答時間と実行経過時間の差です)。

エレメント ID

network_time_bottom

エレメント・タイプ

水準点

表 1023. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント、タイム・スタンプ
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
データ伝送	stmt_transmissions	ステートメント、タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

network_time_top ステートメントの最大ネットワーク時間

このエレメントは、 DCS データベースに対してまたはこの DCS アプリケーション内で実行されたステートメントについて、あるいはこの回数のデータ伝送を使用したステートメントについて、最長ネットワーク時間を示します (ネットワーク時間は、1 つのステートメントに対するホスト応答時間と実行経過時間の差です)。

エレメント ID

network_time_top

エレメント・タイプ

水準点

表 1024. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント、タイム・スタンプ
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
データ伝送	stmt_transmissions	ステートメント、タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。「タイム・スタンプ」スイッチが OFF のときは、このエレメントは収集されません。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

nleaf - リーフ・ページ数 : モニター・エレメント

リーフ・ページのおおよその数。

表 1025. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

nlevels - 索引レベル数 : モニター・エレメント

索引レベルの数。これは近似値です。

表 1026. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

no_change_updates - 無変更の行更新数のモニター・エレメント

結果的に行のいずれの列値も変更されなかった行更新の数。

LOB、XML、または LONG 列が UPDATE ステートメントの SET 節に含まれている場合、そのステートメントの対象となる行はすべて変更されたものと見なされるため、このカウンターには関係しません。

表 1027. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

node_number ノード番号

db2nodes.cfg ファイル内でノードに割り当てられた番号。

表 1028. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本
データベース・マネージャー	memory_pool	基本
データベース・マネージャー	fcm	基本
データベース・マネージャー	fcm_node	基本
データベース・マネージャー	utility_info	基本
データベース	detail_log	基本
バッファー・プール	bufferpool_nodeinfo	バッファー・プール
表スペース	rollforward	基本
ロック	lock	基本
ロック	lock_wait	基本
データベース	db_sto_path_info	バッファー・プール

表 1029. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_connheader	常に収集される
デッドロック	lock	常に収集される
オーバーフロー・レコード	event_overflow	常に収集される
データベース	event_dbmemuse	常に収集される
接続	event_connmemuse	常に収集される

使用法 この値は現在のノード番号を示しており、複数のノードをモニターするときにこの値を利用できます。

nonboundary_leaf_node_splits - 非境界リーフ・ノードの分割 : モニター・エレメント

挿入操作時に非境界リーフ・ノードが分割された回数。

表 1030. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

num_agents ステートメントで作動しているエージェントの数

現在ステートメントまたはサブセクションを実行している並行エージェントの数。

エレメント ID

num_agents

エレメント・タイプ
ゲージ

表 1031. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
アプリケーション	subsection	ステートメント

使用法 照会の並列処理の度合いを示します。スナップショットを連続的にとることによって、照会の実行進行状況を追跡するときに役に立ちます。

num_assoc_agents 関連したエージェント数

これは、アプリケーション・レベルでは、 1 つのアプリケーションに関連付けられているサブエージェントの数です。データベース・レベルでは、すべてのアプリケーション用のサブエージェントの数です。

エレメント ID
num_assoc_agents

エレメント・タイプ
ゲージ

表 1032. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl_info	基本

使用法 このエレメントは、エージェント構成パラメーターの設定を評価するのに役立ちます。

num_compilations - ステートメント・コンパイル数

特定の SQL ステートメントに対する異なるコンパイルの数。

表 1033. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

使用法 異なるスキーマで発行された SQL ステートメント (**SELECT t1 FROM test** など) の中には、それらが異なるアクセス・プランを参照する場合でも DB2 キャッシュ内では同じステートメントと見なされるものがあります。この値と num_executions を組み合わせて使用すると、コンパイル環境に問題があるために動的 SQL スナップショット統計の結果に狂いが生じていないかどうかを判別できます。

num_coord_exec - コーディネーター・エージェントによる実行数 : モニター・エレメント

このセクションがコーディネーター・エージェントによって実行された回数。

表 1034. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1035. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

num_coord_exec_with_metrics - メトリック付きの、コーディネーター・エージェントによる実行数 : モニター・エレメント

このセクションがコーディネーター・エージェントによって実行され、モニター・メトリックがキャプチャーされた回数

表 1036. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1037. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

num_db_storage_paths 自動ストレージ・パスの数

データベースに関連した自動ストレージ・パスの数。

表 1038. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用上の注意

このエレメントを db_storage_path モニター・エレメントと一緒に使用して、このデータベースに関連したストレージ・パスを識別できます。

ストレージ・グループを使用する場合、このエレメントは、デフォルトのデータベース・ストレージ・グループ内の自動ストレージ・パスの数だけを示します。

num_executions - ステートメント実行回数 : モニター・エレメント

SQL ステートメントが実行された回数。

表 1039. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1040. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1041. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このエレメントを使用して、システムで最も頻繁に実行された SQL ステートメントを識別できます。

ステートメントごとに報告されたアクティビティー・メトリックについて、パッケージ・キャッシュ・レベルでの平均を計算するには、このエレメントを使用します。例えば、パッケージ・キャッシュ・レベルで報告されたあるステートメントの実行当たりの平均 CPU 使用量は、次の数式で計算できます。

```
total_cpu_time / num_exec_with_metrics
```

平均を計算する場合は、**num_executions** モニター・エレメントではなく **num_exec_with_metrics** モニター・エレメントを使用してください。**num_executions** モニター・エレメントでは、報告されるアクティビティー・メトリックにステートメントの実行が寄与したかどうかにかかわらず、ステートメントのすべての実行がカウントされるためです。

num_exec_with_metrics メトリックが収集された実行数 : モニター・エレメント

この SQL ステートメント・セクションが、メトリックが収集されて実行された回数。このエレメントを使用してパッケージ・キャッシュのステートメントにあるモニター・エレメントの実行ごとの値を計算することもできます。

表 1042. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1043. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

num_extents_left プロセス残エクステント数 : モニター・エレメント

この表のリバランス・プロセス中に移動するエクステントで、まだ残っているエクステント数。

表 1044. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE

num_extents_moved - 移動したエクステントの数 : モニター・エレメント

このエクステント移動操作中に、これまでに移動されたエクステントの数。

表 1045. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE

num_gw_conn_switches - 接続切り替え回数

ある接続に対してエージェント・プールのエージェントがプライム状態にされ、別の DRDA データベースで使用するために再割り当てされた回数。

表 1046. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

ほとんどのユーザーの場合、**num_poolagents** 構成パラメーターのデフォルト設定で最適なパフォーマンスが確保されます。この構成パラメーターのデフォルト設定では、自動的にエージェント・プールが管理され、エージェントが再割り当てされなくなります。

このモニター・エレメントの値を小さくするには、**num_poolagents** 構成パラメーターの値を調整してください。

num_indoubt_trans 未確定トランザクション数

データベース内に残っている未確定トランザクションの数。

表 1047. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1048. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 未確定トランザクションは非コミット・トランザクションのログ・スペースを保留するため、ログが満杯になる可能性があります。ログが満杯になると、追加のトランザクションは完了できません。この問題を解決するには、手動による試行錯誤によって未確定トランザクションを解決する必要があります。このモニター・エレメントは、試行錯誤的に解決すべき未確定トランザクションの現在の残存数を提供します。

num_log_buffer_full - エージェントがモニター・エレメントを待機する原因となったフル・ログ・バッファーの回数

num_log_buffer_full エレメントは、ログ・レコードをログ・バッファーにコピーする際に、ログ・データをディスクに書き込むのをエージェントが待機しなければならなかった回数を格納します。この値は、問題が生じるたびにエージェント数単位で大きくなります。

例えば、バッファーが満杯の場合に 2 つのエージェントがログ・データをコピーしようとすると、この値は 2 単位ずつ大きくなります。

表 1049. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1050. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1051. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 **logbufsz** データベース構成パラメーターの値を大きくする必要があるかどうかを判別します。

num_log_data_found_in_buffer ログ・データがバッファーにある回数

エージェントがバッファーからログ・データを読み取る回数。バッファーからログ・データを読み取る方が、ディスクから読み取るより速いので望ましいといえます。

表 1052. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1053. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1054. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *num_log_read_io* エレメントを組み合わせて使用すると、LOGBUFSZ データベース構成パラメーターの値を大きくする必要があるかどうかを判別します。

num_log_part_page_io 部分ログ・ページ書き込み数

ロガーが部分的なログ・データをディスクに書き込むのに発行した入出力要求の数。

表 1055. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1056. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1057. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *log_writes*、 *log_write_time*、および *num_log_write_io* エレメントを組み合わせて使用すると、現行ディスクがロギングに適しているかどうかを判別できます。

num_log_read_io ログ読み取り数

ロガーがログ・データをディスクから読み取るのに発行した入出力要求の数。

表 1058. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1059. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1060. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *log_reads* および *log_read_time* エレメントを組み合わせて使用すると、現行ディスクがロギングに適しているかどうかを判別できます。

num_log_write_io ログ書き込み数

ロガーがログ・データをディスクに書き込むのに発行した入出力要求の数。

表 1061. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1062. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1063. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *log_writes* および *log_write_time* エレメントを組み合わせて使用すると、現行ディスクがロギングに適しているかどうかを判別できます。

num_lw_thresh_exceeded - ロック待機の超過されたしきい値の数 : モニター・エレメント

このモニター・エレメントはロック待機しきい値 (**mon_lw_thresh** 構成パラメーターを使用して設定) が超過され、ロック待機イベントがロック・イベント・モニターによってキャプチャーされた回数を記録します。ロック待機イベントが生成されなければ、モニター・エレメントは増分されません。

表 1064. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 1064. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1065. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

表 1065. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

num_nodes_in_db2_instance パーティション内のノード数

スナップショットが取られたインスタンス上のノード数。

表 1066. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

表 1067. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	常に収集される

使用法 このエレメントを使用して、インスタンスにおけるノード数を判別します。非パーティション・システムのデータベースの場合、この値は 1 になります。

num_page_dict_built - 作成または再作成されたページ・レベルのコンプレッション・ディクショナリーの数

データベースが最後に活動化されてから、表に関して作成または再作成されたページ・レベルのコンプレッション・ディクショナリーの数。

表 1068. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

num_ref_with_metrics - メトリックに関する参照回数のモニター・エレメント

セクションがデータベース・オブジェクトを参照した合計回数。データ・オブジェクト用の使用リストが作成され、アクティブになっていなければなりません。また、セクションに関するオブジェクト・メトリックの収集が有効になっていなければなりません。

表 1069. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

num_references - 参照回数モニター・エレメント

リストに追加されて以降、このセクションがこのオブジェクトを参照した回数。

表 1070. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

num_remaps 再マップ数 : モニター・エレメント

このアクティビティーが再マップされた回数のカウント。 num_remaps がゼロより大きい場合、このアクティビティー・レコードの service_class_id は、アクティビティーが最後に再マップされたサービス・クラスの ID です。

表 1071. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

この情報を使用して、予期された回数アクティビティーが再マップされたかどうかを検証します。

num_routines - ルーチン数のモニター・エレメント

セクション実行中に呼び出された可能性があるプロシージャー、外部関数、コンパイル済み関数、およびコンパイル済みトリガーの数。

表 1072. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	常に収集される
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される

表 1073. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
アクティビティー	event_activitystmt	常に収集される
パッケージ・キャッシュ	event_pkgcache	常に収集される

使用法

MON_GET_SECTION_ROUTINE 表関数を使用して、ルーチンとトリガーをリストします。このリストは MON_GET_ROUTINE 表関数と MON_GET_ROUTINE_EXEC_LIST 表関数の出力と比較できます。

num_tbsps - 表スペース数のモニター・エレメント

ログに記録されたイベントに関連する表スペースの数。

表 1074. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	utilstart	常に収集される

num_threshold_violations しきい値違反の回数 : モニター・エレメント

このデータベースが最後にアクティブにされてからそこで発生したしきい値違反の回数。

このモニター・エレメントは、いくつかのモニター (MON_*) 表関数によって戻される 1593 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』 モニター・エレメントの別名です。

表 1075. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1076. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このエレメントを使用すると、この特定のアプリケーションにおいてしきい値が有効であるかどうか、またはしきい値違反が多すぎないかを判別する助けになります。

num_transmissions 伝送回数

DB2 Connect ゲートウェイとこの DCS ステートメントを処理するのに使用されたホストの間の伝送回数。 (1 回のデータ伝送は、送信 1 回または受信 1 回を示します。)

注:

これは、DB2 UDB バージョン 8.1.2 以降には無関係なレガシー・モニター・エレメントです。 DB2 UDB バージョン 8.1.2 以降をご使用の場合は、**num_transmissions_group** モニター・エレメントを参照してください。

エレメント ID

num_transmissions

エレメント・タイプ

カウンター

-->

表 1077. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS ステートメント	dcs_stmt	ステートメント

使用法 このエレメントを使用すると、特定のステートメントの実行時間が長くかかった理由が明確になります。例えば、照会のときに大きな結果セットを戻すと、完了するまでにたくさんのデータ伝送回数が必要になります。

num_transmissions_group 伝送グループの回数

この DCS ステートメントを処理するのに使われた、 DB2 Connect ゲートウェイとホストの間の伝送範囲。 (1 回のデータ伝送は、送信 1 回または受信 1 回を示します。)

表 1078. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS ステートメント	dcs_stmt	ステートメント

使用法 このエレメントを使用すると、特定のステートメントの実行時間が長くかかった理由が明確になります。例えば、照会のときに大きな結果セットを戻すと、完了するまでにたくさんのデータ伝送回数が必要になります。

伝送範囲を表す定数は以下のように記述され、sqlmon.h で定義されています。

API 定数	説明
SQLM_DCS_TRANS_GROUP_2	2 回の伝送
SQLM_DCS_TRANS_GROUP_3TO7	3 回から 7 回までの伝送
SQLM_DCS_TRANS_GROUP_8TO15	8 回から 15 回までの伝送
SQLM_DCS_TRANS_GROUP_16TO64	16 回から 64 回までの伝送
SQLM_DCS_TRANS_GROUP_GT64	64 回を超える伝送

number_in_bin ビン内の数 : モニター・エレメント

このエレメントは、ヒストグラム・ビンの中に入るアクティビティーまたは要求のカウント数を保持します。

表 1079. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントを使用すると、ヒストグラムのビンの高さを示すことができます。

object_data_gbp_indep_pages_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント

エージェントによってローカル・バッファー・プール (LBP) で検出された、グループ・バッファー・プール (GBP) に従属しないデータ・ページの数。

表 1080. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

表 1081. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

object_data_gbp_invalid_pages - 表に関する GBP の無効なデータ・ページのモニター・エレメント

表に関して、データ・ページがグループ・バッファー・プール (GBP) に対して要求された回数。ページが要求された理由は、ローカル・バッファー・プール (LBP) に存在したページのバージョンが無効であったためです。 DB2 pureScale 環境以外では、この値は NULL になります。

表 1082. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求されたデータ・ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_lbp_pages_found + object_xda_lbp_pages_found)
 / (object_data_l_reads + object_xda_l_reads)
```

要求されたデータ・ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_GBP_l_reads + object_xda_l_reads - object_data_GBP_p_reads
 - object_xda_p_reads) /
(object_data_GBP_l_reads + object_xda_l_reads)
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_data_gbp_l_reads - 表の GBP データの論理読み取りモニター・エレメント

表に関して、グループ・バッファー・プール (GBP) 従属のデータ・ページが、GBP に対して要求された回数。ページが要求された理由は、有効なバージョンのページがローカル・バッファー・プール (LBP) に存在しなかったためです。

表 1083. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求されたデータ・ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_lbp_pages_found + object_xda_lbp_pages_found)
 / (object_data_l_reads + object_xda_l_reads)
```

要求されたデータ・ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_GBP_l_reads + object_xda_l_reads - object_data_GBP_p_reads
 - object_xda_p_reads) /
(object_data_GBP_l_reads + object_xda_l_reads)
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_data_gbp_p_reads - 表の GBP データの物理読み取りモニター・エレメント

表に関して、グループ・バッファー・プール (GBP) 従属のデータ・ページが、ディスクからローカル・バッファー・プール (LBP) に読み取られた回数。ページがディスクから LBP に読み取られた理由は、そのページが GBP に存在しなかったためです。

表 1084. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求されたデータ・ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_lbp_pages_found + object_xda_lbp_pages_found)
 / (object_data_l_reads + object_xda_l_reads)
```

要求されたデータ・ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_GBP_l_reads + object_xda_l_reads - object_data_GBP_p_reads
 - object_xda_p_reads) /
(object_data_GBP_l_reads + object_xda_l_reads)
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_data_lbp_pages_found - 表の検出済み LBP データ・ページ・モニター・エレメント

表のデータ・ページが、ローカル・バッファー・プール (LBP) に存在していた回数。

表 1085. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求されたデータ・ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_lbp_pages_found + object_xda_lbp_pages_found)
 / (object_data_l_reads + object_xda_l_reads)
```

要求されたデータ・ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_GBP_l_reads + object_xda_l_reads - object_data_GBP_p_reads
 - object_xda_p_reads) /
(object_data_GBP_l_reads + object_xda_l_reads)
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_data_l_reads - 表のバッファー・プール・データの論理読み取りモニター・エレメント

表に関してバッファー・プールから論理的に読み取られたデータ・ページ数。

表 1086. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

このモニター・エレメントは、以下のデータに対するアクセス数を追跡します。

- データベース・マネージャーがページの処理を必要としたときにバッファー・プールにあるデータ。

- データベース・マネージャーがページを処理する前にバッファー・プールに読み取られたデータ。

モニター・エレメント値を使用する以下の数式によって、データ・ページ・ヒット率を計算してください。

```
(1 - (object_data_p_reads + object_xda_p_reads) /
(object_data_l_reads + object_xda_l_reads))
```

ヒット率が低い場合は、バッファー・プール・ページ数を増やすと、パフォーマンスが向上する場合があります。

object_data_p_reads - 表のバッファー・プールの物理データ読み取りモニター・エレメント

表に関して物理的に読み取られたデータ・ページ数。

表 1087. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

モニター・エレメント値を使用する以下の数式によって、データ・ページ・ヒット率を計算してください。

```
(1 - (object_data_p_reads + object_xda_p_reads) /
(object_data_l_reads + object_xda_l_reads))
```

ヒット率が低い場合は、バッファー・プール・ページ数を増やすと、パフォーマンスが向上する場合があります。

object_index_gbp_indep_pages_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント

エージェントによってローカル・バッファー・プール (LBP) で検出された、グループ・バッファー・プール (GBP) に従属しない索引ページの数。

表 1088. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	DATA OBJECT METRICS EXTENDED

表 1089. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

object_index_gbp_invalid_pages - 索引に関する GBP の無効な索引ページのモニター・エレメント

索引に関して、索引ページがグループ・バッファー・プール (GBP) に対して要求された回数。ページが要求された理由は、ローカル・バッファー・プール (LBP) に存在したページのバージョンが無効であったためです。

表 1090. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求された索引ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
object_index_lbp_pages_found / object_index_l_reads
```

要求された索引ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_index_gbp_l_reads - object_index_gbp_p_reads) / object_index_gbp_l_reads
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_index_gbp_l_reads - 索引の GBP 索引論理読み取りモニター・エレメント

索引に関して、グループ・バッファー・プール (GBP) 従属の索引ページが、 GBP に対して要求された回数。ページが要求された理由は、有効なバージョンのページがローカル・バッファー・プール (LBP) に存在しなかったためです。

表 1091. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	DATA OBJECT METRICS EXTENDED

表 1091. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求された索引ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
object_index_lbp_pages_found / object_index_l_reads
```

要求された索引ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_index_gbp_l_reads - object_index_gbp_p_reads) / object_index_gbp_l_reads
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_index_gbp_p_reads - 索引の GBP 索引物理読み取りモニター・エレメント

索引に関して、グループ・バッファー・プール (GBP) 従属の索引ページが、ディスクからローカル・バッファー・プール (LBP) に読み取られた回数。ページがディスクから LBP に読み取られた理由は、そのページが GBP に存在しなかったためです。

表 1092. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求された索引ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
object_index_lbp_pages_found / object_index_l_reads
```

要求された索引ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_index_gbp_l_reads - object_index_gbp_p_reads) / object_index_gbp_l_reads
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用

すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_index_lbp_pages_found - 索引に関して検出された LBP の索引ページのモニター・エレメント

索引の索引ページが、ローカル・バッファー・プール (LBP) に存在していた回数。

表 1093. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求された索引ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
object_index_lbp_pages_found / object_index_l_reads
```

要求された索引ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_index_gbp_l_reads - object_index_gbp_p_reads) / object_index_gbp_l_reads
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_index_l_reads - 索引のバッファー・プール索引論理読み取りモニター・エレメント

索引に関してバッファー・プールから論理的に読み取られた索引ページ数。

表 1094. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

このモニター・エレメントは、以下のページに対するアクセス数を追跡します。

- データベース・マネージャーが索引ページの処理を必要とした時点でバッファー・プールに存在していた索引ページ。

- データベース・マネージャーが索引ページを処理する前にバッファー・プールに読み取られた索引ページ。

モニター・エレメント値を使用する以下の数式によって、索引ページ・ヒット率を計算してください。

```
(1 - (object_index_p_reads / object_index_l_reads))
```

ヒット率が低い場合は、バッファー・プール・ページ数を増やすと、パフォーマンスが向上する場合があります。

object_index_p_reads - 索引のバッファー・プール索引物理読み取り

索引に関して物理的に読み取られた索引ページ数。

表 1095. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

モニター・エレメント値を使用する以下の数式によって、索引ページ・ヒット率を計算してください。

```
(1 - (object_index_p_reads / object_index_l_reads))
```

ヒット率が低い場合は、バッファー・プール・ページ数を増やすと、パフォーマンスが向上する場合があります。

object_name - オブジェクト名モニター・エレメント

objtype モニター・エレメントの値に応じた表名または索引名。

表 1096. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_AUTO_MAINT_QUEUE 表関数 - 自動保守ジョブの情報の取得	ACTIVITY METRICS BASE
MON_GET_AUTO_RUNSTATS_QUEUE 表関数 - 評価用にキューに入れられたオブジェクトの情報の取得	ACTIVITY METRICS BASE
MON_GET_RTS_RQST 表関数 - リアルタイム統計要求の情報の取得	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

表 1097. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	utilphase utilstart	常に収集される

使用法

変更履歴イベント・モニターでは、 `object_type` エレメントが INDEX、PARTIONGROUP、または TABLE の場合、これは索引、パーティション・グループ、または表の名前です。

object_requested - 要求されたオブジェクトのモニター・エレメント

要求者が所有者からの取得を試みたロックのタイプ。値は、データベース・ロックを表す LOCK、または WLM チケットを表す TICKET のいずれかです。

表 1098. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

object_schema - オブジェクト・スキーマのモニター・エレメント

objtype モニター・エレメントの値に応じて、表のスキーマまたは索引のスキーマ。

表 1099. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_AUTO_MAINT_QUEUE 表関数 - 自動保守ジョブの情報の取得	ACTIVITY METRICS BASE
MON_GET_AUTO_RUNSTATS_QUEUE 表関数 - 評価用にキューに入れられたオブジェクトの情報の取得	ACTIVITY METRICS BASE
MON_GET_RTS_RQST 表関数 - リアルタイム統計要求の情報の取得	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

表 1100. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	utilphase utilstart	常に収集される

使用法

変更履歴イベント・モニターでは、 `object_type` エレメントが INDEX または TABLE の場合、これは索引または表のスキーマであり、それ以外の場合は空のストリングです。

object_xda_gbp_indep_pages_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント

エージェントによってローカル・バッファー・プール (LBP) で検出された、グループ・バッファー・プール (GBP) に従属しない XML ストレージ・オブジェクト (XDA) のデータ・ページ数。

表 1101. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

表 1102. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

object_xda_gbp_invalid_pages - 表に関する GBP の無効な XDA データ・ページのモニター・エレメント

表に関して、XML ストレージ・オブジェクト (XDA) のデータ・ページが、グループ・バッファー・プール (GBP) に対して要求された回数。ページが要求された理由は、ローカル・バッファー・プール (LBP) に存在したページのバージョンが無効であったためです。

表 1103. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求された XDA ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
object_xda_lbp_pages_found / object_xda_l_reads
```

要求された XDA ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_xda_gbp_l_reads - object_xda_gbp_p_reads) / object_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_xda_gbp_l_reads - 表の GBP XDA データの論理読み取り要求モニター・エレメント

表に関して、XML ストレージ・オブジェクト (XDA) のグループ・バッファー・プール (GBP) 従属のデータ・ページが、 GBP に対して要求された回数。ページが要求された理由は、有効なバージョンのページがローカル・バッファー・プール (LBP) に存在しなかったためです。

表 1104. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求された XDA ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
object_xda_lbp_pages_found / object_xda_l_reads
```

要求された XDA ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_xda_gbp_l_reads - object_xda_gbp_p_reads) / object_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_xda_gbp_p_reads - 表の GBP XDA データの物理読み取り要求モニター・エレメント

表に関して、XML ストレージ・オブジェクト (XDA) のグループ・バッファー・プール (GBP) 従属のデータ・ページが、ディスクからローカル・バッファー・プール (LBP) に読み取られた回数。ページがディスクから LBP に読み取られた理由は、そのページが GBP に存在しなかったためです。

表 1105. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED

表 1105. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求された XDA ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
object_xda_lbp_pages_found / object_xda_l_reads
```

要求された XDA ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_xda_gbp_l_reads - object_xda_gbp_p_reads) / object_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_xda_lbp_pages_found - 表の検出済み LBP XDA データ・ページ・モニター・エレメント

表の XML ストレージ・オブジェクト (XDA) のデータ・ページが、ローカル・バッファー・プール (LBP) に存在していた回数。

表 1106. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

要求されたデータ・ページが LBP で検出された頻度を計算するには、モニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_lbp_pages_found + object_xda_lbp_pages_found)
 / (object_data_l_reads + object_xda_l_reads)
```

要求されたデータ・ページが GBP で検出された頻度を計算するには、同じくモニター・エレメントの値を使用する以下の数式を使用します。

```
(object_data_GBP_l_reads + object_xda_l_reads - object_data_GBP_p_reads
 - object_xda_p_reads) /
(object_data_GBP_l_reads + object_xda_l_reads)
```

クラスター・キャッシング・ファシリティー の全体的なパフォーマンスにおいて、LBP および GBP のヒット・レートは重要な要因となります。これらの数式を使用

すると、LBP または GBP によってデータベースのスループットが制限されていないか判別するのに役立ちます。

object_xda_l_reads - 表のバッファー・プール XDA データの論理読み取りモニター・エレメント

表に関してバッファー・プールから論理的に読み取られた XML ストレージ・オブジェクト (XDA) のデータ・ページ数。

表 1107. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

このモニター・エレメントは、以下のデータに対するアクセス数を追跡します。

- データベース・マネージャーがページの処理を必要としたときにバッファー・プールにあるデータ。
- データベース・マネージャーがページを処理する前にバッファー・プールに読み取られたデータ。

モニター・エレメント値を使用する以下の数式によって、データ・ページ・ヒット率を計算してください。

```
(1 - (object_data_p_reads + object_xda_p_reads) /
(object_data_l_reads + object_xda_l_reads))
```

ヒット率が低い場合は、バッファー・プール・ページ数を増やすと、パフォーマンスが向上する場合があります。

object_xda_p_reads - 表のバッファー・プール XDA データの物理読み取りモニター・エレメント

表に関して物理的に読み取られた XML ストレージ・オブジェクト (XDA) のデータ・ページ数。

表 1108. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

モニター・エレメント値を使用する以下の数式によって、データ・ページ・ヒット率を計算してください。

```
(1 - (object_data_p_reads + object_xda_p_reads) /
(object_data_l_reads + object_xda_l_reads))
```

ヒット率が低い場合は、バッファー・プール・ページ数を増やすと、パフォーマンスが向上する場合があります。

objtype - オブジェクト・タイプ : モニター・エレメント

レポート対象のモニター・データのオブジェクト・タイプ。このモニター・エレメントは、object_type モニター・エレメントの別名です。

表 1109. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_TAB_COMPRESS_INFO 表関数 - 圧縮節約量の見積もり	ACTIVITY METRICS BASE
ADMIN_GET_TAB_DICTIONARY_INFO 表関数 - 既存の表ディクショナリーのプロパティーの報告	ACTIVITY METRICS BASE
MON_GET_AUTO_MAINT_QUEUE 表関数 - 自動保守ジョブの情報の取得	ACTIVITY METRICS BASE
MON_GET_AUTO_RUNSTATS_QUEUE 表関数 - 評価用にキューに入れられたオブジェクトの情報の取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_RTS_RQST 表関数 - リアルタイム統計要求の情報の取得	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE

表 1110. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	utilphase utilstart	常に収集される

使用上の注意

- このエレメントが ADMIN_GET_TAB_COMPRESS_INFO 表関数または ADMIN_GET_TAB_DICTIONARY_INFO 表関数から戻された場合、 object_type モニター・エレメントの戻り値は、「XML」または「DATA」のいずれかです。

- このエレメントが MON_GET_RTS_RQST 表関数から戻された場合、 object_type モニター・エレメントの戻り値は、「TABLE」です。
- このエレメントが MON_GET_AUTO_MAINT_QUEUE 表関数から戻された場合、 object_type モニター・エレメントの戻り値は、
「DATABASE」、「TABLE」、「NICKNAME」、または「VIEW」です。
- このエレメントが MON_GET_AUTO_RUNSTATS_QUEUE 表関数から戻された場合、 object_type モニター・エレメントの戻り値は、
「TABLE」、「NICKNAME」、または「VIEW」です。
- このエレメントが変更履歴イベント・モニターから戻された場合、object_type モニター・エレメントの戻り値は、
「DATABASE」、「INDEX」、「PARTITIONGROUP」、「TABLE」、または「TABLESPACE」です。

olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント

OLAP 関数データが、使用可能なソート・ヒープ・スペースを超えた回数。

表 1111. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1112. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法

データベース・レベルでは、このエレメントと total_olap_funcs を組み合わせて使用すると、ディスクにオーバーフローした OLAP 関数のパーセンテージを計算できます。このパーセンテージが高く、OLAP 関数を使用するアプリケーションのパフォーマンスを改善する必要がある場合は、ソート・ヒープ・サイズを大きくすることを検討してください。

アプリケーション・レベルでは、このエレメントを使用すると、個々のアプリケーションにおける OLAP 関数のパフォーマンスを評価できます。

open_cursors オープン・カーソル数

アプリケーションで現在開いているカーソルの数。

表 1113. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	ステートメント

使用法 このエレメントを使用して、割り振られているメモリーの量を評価します。ターゲット・データベース上の DB2 クライアント、DB2 Connect、またはデータベース・エージェントに割り振られるメモリー量は、現在開いているカーソルの数と関連しています。この情報は、キャパシティー・プランニングに利用できます。例えば、ブロッキングを行っている各オープン・カーソルのバッファー・サイズは RQRIOBLK です。 *deferred_prepare* を使用可能にすると、 2 つのバッファーが割り振られます。

このエレメントには、早期クローズによって閉じられたカーソルは組み込まれていません。ホスト・データベースが最後のレコードをクライアントに戻すと、早期クローズが生じます。ホストとゲートウェイではカーソルが閉じられますが、クライアント側では依然として開いています。早期クローズ・カーソルは、DB2 コール・レベル・インターフェースを使用して設定できます。

open_loc_curs 開かれているローカル・カーソル

このアプリケーション用に現在開かれているローカル・カーソルの数。 *open_loc_curs_blk* がカウントするカーソルを含みます。

エレメント ID

open_loc_curs

エレメント・タイプ

ゲージ

表 1114. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントと *open_loc_curs_blk* を組み合わせて使用すると、ローカル・ブロック・カーソルのパーセンテージを計算できます。パーセンテージが低い場合は、アプリケーション内の行ブロッキングを改善するとパフォーマンスが向上します。

リモート・アプリケーションが使用するカーソルについては、『*open_rem_curs*』を参照してください。

open_loc_curs_blk 開かれているローカル・ブロック・カーソル

このアプリケーション用に現在開かれているローカル・ブロック・カーソルの数。

エレメント ID

open_loc_curs_blk

エレメント・タイプ
ゲージ

表 1115. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントと *open_loc_curs* を組み合わせて使用すると、ローカル・ブロック・カーソルのパーセンテージを計算できます。パーセンテージが低い場合は、アプリケーション内の行ブロッキングを改善するとパフォーマンスが向上します。

- 未確定カーソル処理のレコード・ブロッキングについて、プリコンパイル・オプションをチェックする。
- カーソルを再定義してブロッキングを許可する (例えば、可能な場合は、カーソルに FOR FETCH ONLY を指定する)。

rej_curs_blk および acc_curs_blk が提供する情報を使用すると、構成パラメーターを調整して、アプリケーション内の行ブロッキングを改善できます。

リモート・アプリケーションが使用するブロック・カーソルについては、『*open_rem_curs_blk*』を参照してください。

open_rem_curs 開かれているリモート・カーソル

このアプリケーション用に現在開かれているリモート・カーソルの数。*open_rem_curs_blk* がカウントするカーソルを含みます。

エレメント ID
open_rem_curs

エレメント・タイプ
ゲージ

表 1116. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントと *open_rem_curs_blk* を組み合わせて使用すると、リモート・ブロック・カーソルのパーセンテージを計算できます。パーセンテージが低い場合は、アプリケーション内の行ブロッキングを改善するとパフォーマンスが向上します。詳しくは、『*open_rem_curs_blk*』を参照してください。

ローカル・データベースに接続されているアプリケーションが使用するオープン・カーソルの数については、『*open_loc_curs*』を参照してください。

open_rem_curs_blk 開かれているリモート・ブロック・カーソル

このアプリケーション用に現在開かれているリモート・ブロック・カーソルの数。

エレメント ID
open_rem_curs_blk

エレメント・タイプ
ゲージ

表 1117. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントと *open_rem_curs* を組み合わせて使用すると、リモート・ブロック・カーソルのパーセンテージを計算できます。パーセンテージが低い場合は、アプリケーション内の行ブロッキングを改善するとパフォーマンスが向上します。

- 未確定カーソル処理のレコード・ブロッキングについて、プリコンパイル・オプションをチェックする。
- カーソルを再定義してブロッキングを許可する (例えば、可能な場合は、カーソルに FOR FETCH ONLY を指定する)。

rej_curs_blk および *acc_curs_blk* が提供する情報を使用すると、構成パラメーターを調整して、アプリケーション内の行ブロッキングを改善できます。

ローカル・データベースに接続されているアプリケーションが使用するオープン・ブロック・カーソルの数については、『*open_loc_curs_blk*』を参照してください。

os_level - オペレーティング・システムのレベル : モニター・エレメント

このホストで実行されているオペレーティング・システムの修正レベル。 Linux システムの場合のみ報告されます。

表 1118. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

os_name - オペレーティング・システム名 : モニター・エレメント

このホストで実行されているオペレーティング・システムの名前。

表 1119. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

os_release - オペレーティング・システムのリリース : モニター・エレメント

このホストで実行されているオペレーティング・システムのリリース。

表 1120. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

os_version - オペレーティング・システムのバージョン : モニター・エレメント

このホストで実行されているオペレーティング・システムのバージョン。

表 1121. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

outbound_appl_id アウトバウンド・アプリケーション ID

アプリケーションが DRDA ホスト・データベースに接続すると、この ID が生成されます。この ID は、DB2 Connect ゲートウェイをホストに接続するときに使用しますが、**appl_id** モニター・エレメントはクライアントを DB2 Connect ゲートウェイに接続するときに使用します。

エレメント ID
outbound_appl_id

エレメント・タイプ
情報

表 1122. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法

このエレメントと **appl_id** を組み合わせて使用すると、アプリケーション情報のクライアントの部分とサーバーの部分を関連付けることができます。

この ID は、ネットワーク内でユニークな ID です。

ゲートウェイ・コンセントレーターがオンである場合、または DCS アプリケーションが作業論理単位内にない場合に、このエレメントはブランクになります。

形式 `Network.LU Name.Application instance`

例 `CAIBMTOR.OSFDBM0.930131194520`

outbound_bytes_received 受信されたアウトバウンド・バイト数

DB2 Connect ゲートウェイがホストから受信したバイト数。通信プロトコル (TCP/IP など) による使用量は含まれません。データ伝送レベルの場合: このデータ伝送回数を使用したすべてのステートメントの処理で、DB2 Connect ゲートウェイがホストから受信したバイト数。

表 1123. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本
DCS ステートメント	dcs_stmt	ステートメント
データ伝送	stmt_transmissions	ステートメント

ステートメント・レベルでのスナップショット・モニターの場合、このカウンターはリセットできません。その他のレベルではこのカウンターはリセットできます。

使用法

このエレメントを使用して、ホスト・データベースから DB2 Connect ゲートウェイへのスループットを測定します。

outbound_bytes_received_bottom 受信された最小アウトバウンド・バイト数

このデータベースに対するステートメントまたはチェーン、またはこの DCS アプリケーション内のステートメントでまたはチェーン、このデータ伝送回数を使用したすべてのステートメントまたはチェーンを処理している間に、 DB2 Connect ゲートウェイがホストから受信したステートメントまたはチェーン単位の最小バイト数。

表 1124. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	ステートメント

使用法 このエレメントは、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを示すもう 1 つのパラメーターである「受信されたアウトバウンド・バイト数」と組み合わせて使用します。

outbound_bytes_received_top 受信された最大アウトバウンド・バイト数

DB2 Connect ゲートウェイがホストから受信したステートメントまたはチェーン単位の最大バイト数。この DCS データベースに対するステートメントまたはチェーン、またはこの DCS アプリケーション内のステートメントまたはチェーンを処理している間に、このデータ伝送回数を使用したすべてのステートメントまたはチェーンが対象になります。

表 1125. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	ステートメント

使用法 このエレメントは、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを示すもう 1 つのパラメーターである「受信されたアウトバウンド・バイト数」と組み合わせて使用します。

outbound_bytes_sent 送信されたアウトバウンド・バイト数

DB2 Connect ゲートウェイがホストに送信したバイト数。通信プロトコル (TCP/IP など) による使用量は含まれません。データ伝送レベルの場合: このデータ伝送回数を使用したすべてのステートメントの処理で、DB2 Connect ゲートウェイがホストに送信したバイト数。

表 1126. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本
DCS ステートメント	dcs_stmt	ステートメント
データ伝送	stmt_transmissions	ステートメント

ステートメント・レベルでのスナップショット・モニターの場合、このカウンターはリセットできません。その他のレベルではこのカウンターはリセットできます。

使用法 このエレメントを使用して、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを測定します。

outbound_bytes_sent_bottom 送信された最小アウトバウンド・バイト数

DB2 Connect ゲートウェイがホストから受信したステートメントまたはチェーン単位の最小バイト数。この DCS データベースに対するステートメントまたはチェーン、またはこの DCS アプリケーション内のステートメントまたはチェーンを処理している間に、このデータ伝送回数を使用したすべてのステートメントまたはチェーンが対象になります。

表 1127. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	ステートメント

使用法 このエレメントは、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを示すもう 1 つのパラメーターである「送信されたアウトバウンド・バイト数」と組み合わせて使用します。

outbound_bytes_sent_top 送信された最大アウトバウンド・バイト数

このデータベースに対するステートメントまたはチェーン、またはこの DCS アプリケーション内のステートメントまたはチェーンで、このデータ伝送回数を使用したすべてのステートメントまたはチェーンを処理している間に、 DB2 Connect ゲートウェイがホストに送信したステートメントまたはチェーン単位の最大バイト数。

表 1128. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	ステートメント

使用法 このエレメントは、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを示すもう 1 つのパラメーターである「送信されたアウトバウンド・バイト数」と組み合わせて使用します。

outbound_comm_address アウトバウンド通信アドレス

これはターゲット・データベースの通信アドレスです。例えば、TCP/IP 用の IP アドレスとポート番号などです。

エレメント ID

outbound_comm_address

エレメント・タイプ

情報

表 1129. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

outbound_comm_protocol アウトバウンド通信プロトコル

DB2 Connect ゲートウェイとホストの間で使用される通信プロトコル。

表 1130. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法

このエレメントは、DCS アプリケーションに関する問題判別に使用します。有効な値は、次のとおりです。

- SQLM_PROT_TCPIP

outbound_sequence_no アウトバウンド・シーケンス番号

ゲートウェイ・コンセントレーターがオンである場合、または DCS アプリケーションが作業論理単位内にない場合に、このエレメントはブランクになります。

表 1131. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

overflow_accesses - オーバーフロー・レコードへのアクセス : モニター・エレメント

この表のオーバーフローした行へのアクセス (読み取りおよび書き込み) 数。

表 1132. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

表 1133. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1134. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法

オーバーフローした行は、データのフラグメント化が生じたことを示します。この数値が大きい場合は、このフラグメント化をクリーンアップする **REORG** ユーティリティーを使用して表を再編成することによって、表のパフォーマンスを改善できることがあります。

行が更新されて、以前に書き込まれていたデータ・ページに収まらなくなった場合に、行はオーバーフローします。 VARCHAR または ALTER TABLE ステートメントを更新すると、こうしたことが起こります。

overflow_creates - オーバーフローの作成 : モニター・エレメント

この表で作成された、オーバーフローした行の数。

表 1135. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

使用法

package_id - パッケージ ID : モニター・エレメント

パッケージのユニーク ID。

表 1136. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	パッケージ・リストに報告されます。	常に収集される

使用法

このエレメントの値は、SYSCAT.PACKAGES ビューの PKGID 列の値と一致します。

package_elapsed_time - パッケージ経過時間 : モニター・エレメント

パッケージ内のセクションの実行にかかった経過時間。値はミリ秒単位です。

表 1137. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	パッケージ・リストに報告されます。	常に収集される

package_list_count - パッケージ・リスト・カウント : モニター・エレメント

ある特定の作業単位のパッケージ・リストにある項目の数

表 1138. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される

package_list_exceeded - パッケージ・リストの超過 : モニター・エレメント

作業単位内で使用されたパッケージ数が、パッケージ・リストの容量を超過したかどうかを示します。可能な値は YES および NO です。

表 1139. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される

package_list_size - パッケージ・リスト・サイズのモニター・エレメント

パッケージ・リストに含まれているパッケージ ID 数のカウント。

表 1140. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	uow	

package_name - パッケージ名 : モニター・エレメント

SQL ステートメントが含まれているパッケージの名前。

表 1141. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1142. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 1143. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-

表 1143. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを使用すると、実行中のアプリケーション・プログラムや SQL ステートメントを識別できます。

package_schema - パッケージ・スキーマ : モニター・エレメント

SQL ステートメントに関連したパッケージのスキーマ名。

表 1144. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1145. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
パッケージ・キャッシュ	-	COLLECT BASE DATA

package_version_id - パッケージ・バージョン : モニター・エレメント

特定のパッケージ名および作成者について、複数のバージョンが存在する場合があります (DB2 バージョン 8 以降)。パッケージ・バージョンは、現在実行中の SQL ステートメントを含むパッケージのバージョン ID を示します。

パッケージのバージョンは、組み込み SQL プログラムをプリコンパイル (PREP) するときに VERSION キーワードを使用して決めます。プリコンパイル時に指定がない場合は、パッケージ・バージョンが "" (空ストリング) となります。

表 1146. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1147. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 1148. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activitystmt	常に収集される
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このエレメントを使用して、パッケージおよび現在実行中の SQL ステートメントの識別に役立てることができます。

packet_receive_errors - パケット受信エラーのモニター・エレメント

ネットワーク・アダプターの開始以降のパケット受信エラー数。

表 1149. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE

packets_received - 受信パケット数のモニター・エレメント

ネットワーク・アダプターの開始以降に受信したパケット数。

表 1150. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE

packet_send_errors - パケット送信エラーのモニター・エレメント

ネットワーク・アダプターの開始以降のパケット送信エラー数。

表 1151. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE

packets_sent - 送信パケット数のモニター・エレメント

ネットワーク・アダプターの開始以降に送信したパケット数。

表 1152. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE

page_allocations - ページ割り振り : モニター・エレメント

索引に割り振られたページ数。

表 1153. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

page_reorgs ページ再編成 : モニター・エレメント

表のページ再編成が実行された回数。

表 1154. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED

表 1155. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1156. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法

以下の状態では、ページに十分なスペースがあってもフラグメント化されることがあります。

- 新しい行が挿入される場合
- 既存の行が更新され、その結果レコード・サイズが大きくなる場合

ページがフラグメント化されている場合は、ページの再編成が必要になることがあります。再編成すると、フラグメント化されたすべてのスペースを連続したエリアに移動して、そこに新しいレコードが書き込まれます。このようなページの再編成 (page reorg) の実行には、数千の指示が必要になることがあります。また、操作のログ・レコードも生成されます。

ページ再編成の回数が多すぎると、最適な挿入パフォーマンスを達成できないことがあります。 REORG TABLE ユーティリティーを使用すると、表を再編成してフラグメント化をなくすことができます。さらに、ALTER TABLE ステートメントで APPEND パラメーターを使用すると、表の末尾にすべての挿入を付加するように指定でき、ページの再編成を回避できます。

行の更新をすると行の長さが増えるような場合は、そのページに新しい行を挿入するだけの場所があっても、そのスペースのフラグメント化を解消するためにページ REORG が必要になる場合があります。ページに新しい大きな行を挿入するだけの場所がない場合は、オーバーフロー・レコードが作成されて、読み取りのときに *overflow_accesses* の原因となります。どちらの状態も、可変長列の代わりに固定長列を使用することで回避できます。

page_reclaims_x - ページ再利用の排他的アクセス : モニター・エレメント

オブジェクトに関連したページが、DB2 pureScale インスタンス内の別のメンバーによって計画的な解放より前に再利用された回数。その場合、そのページを再利用したメンバーは排他的アクセスを必要としました。

表 1157. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

内部的に保持されているオブジェクト・スペース・マップに関連するページ再利用は、別個にカウントされます。

page_reclaims_s - ページ再利用の共有アクセス : モニター・エレメント

オブジェクトに関連したページが、DB2 pureScale インスタンス内の別のメンバーによって計画的な解放より前に再利用された回数。その場合、そのページを再利用したメンバーは共有アクセスを必要としました。

表 1158. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

内部的に保持されているオブジェクト・スペース・マップに関連するページ再利用は、別個にカウントされます。

page_reclaims_initiated_x - 排他的アクセスで開始されたページ再利用 : モニター・エレメント

ページが別のメンバーから再利用される原因となった、排他モードでのそのページのアクセス回数。内部的に保持されているオブジェクト・スペース・マップに関連するページ再利用は、別個にカウントされます。

表 1159. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

page_reclaims_initiated_s - 共有アクセスで開始されたページ再利用 : モニター・エレメント

ページが別のメンバーから再利用される原因となった、共有モードでのページ・アクセス回数。内部的に保持されているオブジェクト・スペース・マップに関連するページ再利用は、別個にカウントされます。

表 1160. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

pages_from_block_ios - ブロック入出力によって読み取られたページ数の合計 : モニター・エレメント

ブロック入出力でバッファー・プールのページ・エリアに読み取られたページ数の合計。

表 1161. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1162. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール

使用法

ブロック・ベースのバッファー・プールが使用可能になっている場合、このエレメントは、ブロック入出力によって読み取られたページ数の合計を報告します。そうでない場合、このエレメントは 0 を戻します。

ブロック・ベースの入出力ごとに順次プリフェッチされたページ数の平均を計算するには、**pages_from_block_ios** モニター・エレメントの値を **block_ios** モニター・エレメントの値で除算します。この値が、CREATE BUFFERPOOL または ALTER BUFFERPOOL ステートメントでブロック・ベースのバッファー・プールについて定義した BLOCKSIZE オプションより大幅に小さい場合は、ブロック・ベースの入出力の利点が最大限に活用されていません。考えられる原因の 1 つは、順次プリフェッチされている表スペースのエクステント・サイズと、ブロック・ベースのバッファー・プールのブロック・サイズが一致していないことです。

pages_from_vectored_ios - ベクトル化入出力によって読み取られたページ数の合計 : モニター・エレメント

ベクトル化入出力でバッファー・プールのページ・エリアに読み取られたページ数の合計。

表 1163. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 1163. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1164. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール

pages_merged - マージされたページ : モニター・エレメント

マージされた索引ページの数。

表 1165. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

pages_read - 読み取られたページの数 : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースの物理表スペース・コンテナーから読み取られたページ (データ、索引、および XML) の数。

表 1166. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

使用法

pages_written - 書き込まれたページの数 : モニター・エレメント

表スペース・コンテナーに物理的に書き込まれたページ (データ、索引、および XML) の数。

表 1167. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

使用法

parent_activity_id 親アクティビティー ID : モニター・エレメント

アクティビティーの親アクティビティーの作業単位内における、その親アクティビティーのユニーク ID。親アクティビティーがない場合、このモニター・エレメントの値は 0 です。

表 1168. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 1169. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを **parent_uow_id** エレメントおよび **appl_id** エレメントと組み合わせて使用すると、このアクティビティー・レコードで記述されているアクティビティーの親アクティビティーを一意的に識別できます。

parent_uow_id 親作業単位 ID : モニター・エレメント

アプリケーション・ハンドル内の固有の作業単位 ID。アクティビティーの親アクティビティーが発生する作業単位の ID。親アクティビティーがない場合、値は 0 です。

表 1170. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 1171. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを **parent_activity_id** エレメントおよび **appl_id** エレメントと組み合わせて使用すると、このアクティビティー・レコードで記述されているアクティビティーの親アクティビティーを一意的に識別できます。

partial_record 部分レコード : モニター・エレメント

イベント・モニター・レコードが部分レコードでしかないことを示します。

表 1172. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表	event_table	-
表スペース	event_tablespace	-
バッファー・プール	event_bufferpool	-
接続	event_conn	-
ステートメント	event_stmt	-
ステートメント	event_subsection	-
トランザクション	event_xact	-
アクティビティー	event_activity	-

使用法

ほとんどのイベント・モニターは、データベースが非活動状態になるまでそれぞれの結果を出力しません。 FLUSH EVENT MONITOR <monitorName> ステートメントを使用すると、モニター値をイベント・モニター出力ライターに強制的に渡すことができます。これにより、イベント・モニターを停止して再始動する必要なしに、イベント・モニター・レコードをライターに強制的に渡すことができます。このエレメントは、イベント・モニター・レコードがフラッシュ操作の結果であり、したがってそれが部分レコードであるかどうかを示します。

イベント・モニターのフラッシュが原因で値がリセットされることはありません。これは、イベント・モニターが起動するときに、完全イベント・モニター・レコードが生成されることを意味します。

event_activity 論理データ・グループで、**partial_record** モニター・エレメントの有効値は以下のとおりです。

0 アクティビティーの終わりに通常どおりアクティビティー・レコードが生成されました。

1 WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーの呼び出しの結果としてアクティビティー・レコードが生成されました。

2 使用可能なストレージが足りずにレコードを作成できなかったため、このアクティビティーに関する情報がありません。
event_activity、event_activitystmt、または event_activityvals レコードの情報が失われている可能性があります。

participant_no デッドロック内の参加者

このデッドロック内のこの参加者を一意的に識別するシーケンス番号。

エレメント ID
participant_no

エレメント・タイプ
情報

表 1173. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	常に収集される
詳細付きデッドロック	event_detailed_dlconn	常に収集される

使用法 モニター・アプリケーションでこのエレメントを使用すると、デッドロック接続イベント・レコードとデッドロック・イベント・レコードを関連付けることができます。

participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者

このアプリケーションがロックの取得を待機しているオブジェクトのロックを保留しているアプリケーションの参加者番号。

エレメント ID
participant_no_holding_lk

エレメント・タイプ
情報

表 1174. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	常に収集される
詳細付きデッドロック	event_detailed_dlconn	常に収集される

使用法 このエレメントは、リソースの競合状態にあるアプリケーションの判別に利用できます。

participant_type - 参加者タイプのモニター・エレメント

ロック参加者のタイプ。これは要求者または所有者のいずれかです。

表 1175. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

partition_key - パーティション・キーのモニター・エレメント

イベント・モニター表のパーティション・キー。各イベント・モニター処理は、この値を選択し、処理を実行中のデータベース・パーティションにデータを挿入します。つまり、挿入操作はイベント・モニター処理を実行しているデータベース・パーティションでローカルに実行します。

表 1176. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
しきい値違反	event_thresholdviolations	
しきい値違反	control	
統計	event_qstats	
統計	event_scstats	
統計	event_histogrambin	
統計	event_wcstats	
統計	event_wlstats	
統計	control	
ロッキング	lock	
ロッキング	lock_participants	
ロッキング	lock_participant_activities	
ロッキング	lock_activity_values	
ロッキング	control	
パッケージ・キャッシュ	pkgcache_metrics	
パッケージ・キャッシュ	pkgcache_stmt_args	
パッケージ・キャッシュ	control	
作業単位	uow	
作業単位	uow_metrics	
作業単位	uow_package_list	
作業単位	uow_executable_list	
作業単位	control	
アクティビティー	event_activity	
アクティビティー	event_activitystmt	
アクティビティー	event_activityvals	
アクティビティー	event_activitymetrics	
アクティビティー	control	

表 1176. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
変更履歴	changesummary dbdbmcfg ddlstmtexec evmonstart regvar txncompletion utillocation utilphase utilstart utilstop	常に収集される

partition_number パーティション番号

このエレメントは、パーティション・データベース環境または DB2 pureScale 環境で、表書き込みイベント・モニターによってターゲット SQL 表でのみ使用されます。この値は、イベント・モニターのデータが挿入されたメンバーの番号を示します。

表 1177. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
DB_MEMBERS 表関数	ACTIVITY METRICS BASE

表 1178. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
-	-	常に収集される

passthru_time パススルー時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの PASSTHRU ステートメントに対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

このモニターは最新の値を格納します。応答時間は、フェデレーテッド・サーバーが PASSTHRU ステートメントをデータ・ソースにサブミットしてからデータ・ソースが PASSTHRU ステートメントを処理したことを応答するまでの時間です。

表 1179. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このデータ・ソースでステートメントをパススルー・モードで処理するのに要した実際の時間を判別できます。

passthrus パススルー数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースに直接渡した SQL ステートメントの合計数が含まれています。

表 1180. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、フェデレーテッド・サーバーがネイティブに処理できる SQL ステートメントのパーセンテージ、およびパススルー・モードが必要なパーセンテージを判別できます。この値が大きい場合は、原因を突き止めて、ネイティブ・サポートをさらに効率的に使用する方法を検討してください。

past_activities_wrapped - 過去のアクティビティー・リストの折り返しモニター・エレメント

アクティビティー・リストが折り返したかどうかを示します。 1 つのアプリケーションが保持する過去のアクティビティーの数のデフォルトの限度は 250 です。このデフォルトは、*DB2_MAX_INACT_STMTS* レジストリー変数を使用してオーバーライドできます。

この限度に対して異なる値を選択することによって、非アクティブ・ステートメントの情報のために用いられるシステム・モニター・ヒープの量を増減できます。

表 1181. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

phase_start_event_id - フェーズ開始イベント ID

対応する UTILPHASESTART の EVENT_ID。

表 1182. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILPHASE	常に収集される

使用法

変更履歴イベント・モニターの場合:

- イベントが、ユーティリティーのフェーズまたは処理段階の停止 (UTILPHASESTOP) である場合、これは、対応するユーティリティーのフェーズ開始 (UTILPHASESTART) の EVENT_ID です。それ以外の場合は -1 です。

PHASE_START_EVENT_TIMESTAMP およびメンバー・エレメントと併用して、フェーズ停止レコードをフェーズ開始レコードに関連付けます。

phase_start_event_timestamp - フェーズ開始イベントのタイム・スタンプ

対応する UTILPHASESTART の時刻

表 1183. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILPHASE	常に収集される

使用法

変更履歴イベント・モニターの場合:

- イベントが、ユーティリティーのフェーズまたは処理段階の停止 (UTILPHASESTOP) である場合、これは、対応するユーティリティーのフェーズ開始 (UTILPHASESTART) の時刻です。それ以外の場合は空です。

PHASE_START_EVENT_ID およびメンバー・エレメントと併用して、フェーズ停止レコードをフェーズ開始レコードに関連付けます。

piped_sorts_accepted 受け入れられたパイプ・ソート

受け入れられたパイプ・ソートの数。

表 1184. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 システム上のアクティブ・ソートには、それぞれメモリーが割り振られるため、使用可能なシステム・メモリーからソートのためのメモリー量を多く取り過ぎることがあります。

パイプ・ソートを要求した数に対して、受け入れられたパイプ・ソートの数が少ない場合は、次の構成パラメーターのいずれかまたは両方を調整するとソート・パフォーマンスを改善できます。

- sortheap
- sheapthres

パイプ・ソートがリジェクトされる場合は、ソート・ヒープを少なくするか、またはソート・ヒープしきい値を大きくすることを考慮してください。ただし、これらのオプションが与えるそれぞれの影響を知っておく必要があります。ソート・ヒープしきい値を高くすると、ソート処理に割り振られるメモリーの量が多くなる可能性があります。その場合、メモリーがディスクにページングされることがあります。ソート・ヒープを少なくすると、マージ・フェーズが増えてソート処理が遅くなることがあります。

piped_sorts_requested 要求されたパイプ・ソート数

要求されたパイプ・ソートの数。

表 1185. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 システム上のアクティブ・ソートには、それぞれメモリーが割り振られるため、使用可能なシステム・メモリーからソートのためのメモリー量を多く取り過ぎることがあります。

ソート・リスト・ヒープ (*sortheap*) およびソート・ヒープしきい値 (*sheapthres*) の構成パラメーターは、ソート操作に使用するメモリー量をコントロールするのに利用できます。また、これらのパラメーターを使用すると、ソートをパイプ化するかどうかを決定することもできます。

ソートをパイプ化するとディスク入出力が少なくなり、パイプ・ソートの数を増やせるので、ソート操作のパフォーマンスを改善し、さらにはシステム全体のパフォーマンスもよくなります。ただし、ソート・ヒープをソートに割り振る時にソート・ヒープしきい値を超えてしまうと、パイプ・ソートは受け入れてもらえません。パイプ・ソートがリジェクトされる場合については、『*piped_sorts_accepted*』を参照してください。

SQL EXPLAIN 出力には、オプティマイザーがパイプ・ソートを要求するかどうかが示されます。

pkg_cache_inserts - パッケージ・キャッシュ挿入 : モニター・エレメント

要求したセクションが使用できなかったためにパッケージ・キャッシュへのロードが必要になった回数の合計。このカウントには、システムが暗黙に準備した数が含まれます。

表 1186. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1187. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1188. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このモニター・エレメントと **pkg_cache_lookups** モニター・エレメントとを組み合わせ、次の公式を使用してパッケージ・キャッシュ・ヒット率を計算します。

```
1 - (パッケージ・キャッシュ挿入 / パッケージ・キャッシュ検索)
```

pkg_cache_lookups - パッケージ・キャッシュ検索 : モニター・エレメント

パッケージ・キャッシュ検索モニター・エレメントは、アプリケーションがパッケージ・キャッシュ内のセクションやパッケージを検索した回数をカウントします。データベース・レベルでは、データベースの開始以降、またはモニター・データのリセット以降の参照の合計数を示します。

このカウンターには、セクションをキャッシュにすでにロードしてある場合と、セクションをキャッシュにロードする必要がある場合が含まれます。エージェントがさまざまなアプリケーションと関連付けられているようなコンセントレーター環境では、新しいエージェントに必要なセクションやパッケージがローカル・ストレージ内にない場合に、パッケージ・キャッシュの検索がさらに必要になります。

表 1189. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1189. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1190. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1191. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

パッケージ・キャッシュ・ヒット率の計算には次の公式を使用します。

```
1 - (パッケージ・キャッシュ挿入 / パッケージ・キャッシュ検索)
```

パッケージ・キャッシュ・ヒット率は、パッケージ・キャッシュが効果的に利用されているかどうかを示します。このヒット率が高い場合 (0.8 を超える値)、キャッシュは効果的に動作しています。このヒット率が低い場合は、パッケージ・キャッシュを大きくする必要があることを示します。

パッケージ・キャッシュのサイズを変えて試すことにより、 **pckcachesz** 構成パラメーターに最適な値を見つける必要があります。例えば、キャッシュのサイズを小さくしても **pkg_cache_inserts** エレメントが増えない場合は、パッケージ・キャッシュのサイズをさらに小さくできます。パッケージ・キャッシュのサイズを小さくすれば、その分のシステム・リソースを他の作業のために使えるようになります。または、パッケージ・キャッシュのサイズを大きくすることにより (それによって **pkg_cache_inserts** が少なくなるのであれば)、システム全体のパフォーマンスを向上させることもできます。この実験は、フル・ワークロードの条件で行うのが最善です。

このエレメントと **ddl_sql_stmts** を組み合わせて使用すると、 DDL ステートメントを実行したときにパッケージ・キャッシュのパフォーマンスに影響を与えるかどうかを判別できます。 DDL ステートメントを実行すると、動的 SQL ステートメントの一部のセクションが無効になる場合があります。無効なセクションは、次に使用されるときにシステムが暗黙的に準備します。 DDL ステートメントを実行すると、多数のセクションが無効になり、その結果、それらのセクションの準備に余分な処理時間が必要になるため、パフォーマンスに大きく影響を及ぼすことがあります。この場合のパッケージ・キャッシュ・ヒット率は、無効なセクションの暗黙的な再コンパイルを反映します。キャッシュに挿入される新しいセクションは反映されないので、パッケージ・キャッシュのサイズを大きくしても総合的なパフォーマンスは改善できません。フル環境を対象に作業する前に、アプリケーション自体のキャッシュを調整すれば、混乱を避けることができます。

実行すべきアクションを考える前に、パッケージ・キャッシュ・ヒット率の値に DDL ステートメントがどのような役割を果たしているのかを明確にする必要があります。 DDL ステートメントがあまり発生しない場合は、キャッシュのサイズを大きくするとキャッシュのパフォーマンスを改善できる場合があります。 DDL ステートメントが頻繁に使用される場合は、 DDL ステートメントを制限する (時間を限定するなど) と改善できる場合があります。

static_sql_stmts および **dynamic_sql_stmts** のカウントは、キャッシュに入れるセクションの数量とタイプに関する情報を提供するときに利用できます。

注: この情報をデータベース・レベルで使用すると、すべてのアプリケーションについて個別の平均パッケージ・キャッシュ・ヒット率を計算できます。特定のアプリケーションのパッケージ・キャッシュ・ヒット率が知りたいときには、この情報をアプリケーション・レベルで調べてください。実行頻度の少ないアプリケーションのキャッシュ要件を満たすためにパッケージ・キャッシュのサイズを大きくしてもあまり意味がありません。

pkg_cache_num_overflows パッケージ・キャッシュ・オーバーフロー

割り振られたメモリーの境界からパッケージ・キャッシュがオーバーフローした回数。

表 1192. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1193. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このエレメントと **pkg_cache_size_top** モニター・エレメントを組み合わせて使用すると、オーバーフローを回避するのにパッケージ・キャッシュのサイズを大きくする必要があるかどうかを判別できます。

pkg_cache_size_top - パッケージ・キャッシュの最高水準点

パッケージ・キャッシュが到達した最大サイズ。

注: pkg_cache_size_top モニター・エレメントは、DB2 バージョン 9.5 以降では非推奨になっています。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1194. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 1195. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

パッケージ・キャッシュがオーバーフローした場合、このエレメントは、オーバーフロー時にパッケージ・キャッシュが到達した最大サイズになります。

このような状態が発生したかどうかを確認するには、 **pkg_cache_num_overflows** モニター・エレメントをチェックしてください。

ワークロードに必要なパッケージ・キャッシュの最小サイズは次のように決定できます。

```
パッケージ・キャッシュの最大サイズ / 4096
```

この結果を切り上げた整数が、オーバーフローを避けるためにパッケージ・キャッシュが必要とする 4K ページの最小数になります。

pool_async_data_gbp_indep_pages_found_in_lbp - 非同期 EDU によってローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント

非同期 EDU によってローカル・バッファー・プールで検出された、グループ・バッファー・プール (GBP) に従属しないデータ・ページの数。

表 1196. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1197. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

pool_async_data_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント

ローカル・バッファー・プール内のページが無効であったために、プリフェッチャーが、グループ・バッファー・プールからデータ・ページを読み取ろうとした回数。

表 1198. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) のデータ・ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads )
 / pool_async_data_gbp_l_reads
```

IBM DB2 pureScale Feature の全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。この数式を使用すると、データベースのスループットでグループ・バッファー・プールがパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_async_data_gbp_l_reads - 非同期のグループ・バッファー・プール・データの論理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属のデータ・ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、プリフェッチャーがグループ・バッファー・プールから読み取ろうとした回数。

表 1199. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) のデータ・ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads )
 / pool_async_data_gbp_l_reads
```

IBM DB2 pureScale Feature の全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。この数式を使用すると、データベースのスループットでグループ・バッファー・プールがパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_async_data_gbp_p_reads - 非同期のグループ・バッファー・プール・データの物理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属のデータ・ページが、GBP 内で検出されなかったために、プリフェッチャーによってディスクからローカル・バッファー・プールに読み込まれた回数。

表 1200. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) のデータ・ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads )
 / pool_async_data_gbp_l_reads
```

IBM DB2 pureScale Feature の全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。この数式を使用すると、データベースのスループットでグループ・バッファー・プールがパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_async_data_lbp_pages_found - 非同期のローカル・バッファー・プールの検出データ・ページ : モニター・エレメント

プリフェッチャーがデータ・ページにアクセスしようとして、そのデータ・ページがローカル・バッファー・プールに存在した回数。

表 1201. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) のデータ・ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads )
 / pool_async_data_gbp_l_reads
```

IBM DB2 pureScale Feature の全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。この数式を使用すると、データベースのスループットでグループ・バッファー・プールがパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント

プリフェッチャーによるオペレーティング・システムに対する非同期読み取り要求の数。これらの要求は一般に、複数ページから成るラージ・ブロック入出力です。

表 1202. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 1202. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1203. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1204. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法

読み取り要求ごとのデータ・ページの平均数を計算するには、次の数式を使用します。

```
pool_async_data_reads / pool_async_data_read_reqs
```

この平均値は、プリフェッチャーで使用される読み取り入出力平均サイズを判別するのに役立ちます。また、測定対象ワークロードのラージ・ブロック入出力要件を理解するうえでもこのデータは役立ちます。

プリフェッチャー読み取り入出力の最大サイズは、関係する表スペースの CREATE TABLESPACE ステートメントの EXTENTSIZE オプションで指定された値ですが、次のようないくつかの状況下ではそれよりも小さくなることがあります。

- エクステントのいくつかのページがすでにバッファー・プールに入っている場合
- オペレーティング・システムの能力を超えている場合
- EXTENTSIZE オプションの値が非常に大きく、ラージ入出力を実行すると全体のパフォーマンスに悪影響が出る場合

pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント

すべてのタイプの表スペースについて、非同期エンジン・ディスパッチ可能単位 (EDU) によって表スペース・コンテナー (物理) から読み取られた、データ・ページの数を示します。

表 1205. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1206. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1207. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法

このエレメントと **pool_data_p_reads** を組み合わせて使用すると、同期的に実行された物理読み取り数を計算できます (つまり、データベース・マネージャーのエージェントが実行したデータ・ページ物理読み取り数)。次の数式を使用します。

```
1-((pool_data_p_reads+pool_index_p_reads)-(pool_async_data_reads+pool_async_index_reads))
/(pool_data_l_reads+pool_index_l_reads)
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。このエレメントは、**num_ioservers** 構成パラメーターを調整するときに役に立ちます。

非同期読み取りは、データベース・マネージャーのプリフェッチャーが実行します。

pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント

非同期ページ・クリーナーまたはプリフェッチャーによりバッファー・プール・データ・ページがディスクに物理的に書き込まれた回数。プリフェッチャーは、プリフェッチするページの場所を作るために、ダーティー・ページをディスクに書き込む場合があります。

表 1208. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1209. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1210. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法 このエレメントを **pool_data_writes** モニター・エレメントと一緒に使用すると、同期的に実行された物理書き込み要求の数 (つまり、データベース・マネージャーのエージェントが実行したデータ・ページの物理書き込み数) を計算できます。次の数式を使用します。

```
pool_data_writes - pool_async_data_writes
```

非同期読み取り数と同期読み取り数を比較すると、バッファー・プール・ページ・クリーナーの動作状態がわかります。この比率は、 **num_iocleaners** 構成パラメーターを調整するときに役に立ちます。

pool_async_index_gbp_indep_pages_found_in_lbp - 非同期 EDU によってローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント

非同期 EDU によってローカル・バッファー・プールで検出された、グループ・バッファー・プール (GBP) に従属しない索引ページの数。

表 1211. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1212. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

pool_async_index_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効な索引データ・ページ : モニター・エレメント

ローカル・バッファー・プール内のページが無効であったために、プリフェッチャーが、グループ・バッファー・プールから索引ページを読み取ろうとした回数。

表 1213. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の索引ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads )
 / pool_async_index_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_index_gbp_l_reads - 非同期のグループ・バッファー・プール索引の論理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属の索引ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、プリフェッチャーがグループ・バッファー・プールから読み取ろうとした回数。

表 1214. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の索引ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads )
 / pool_async_index_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_index_gbp_p_reads - 非同期のグループ・バッファー・プール索引の物理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属の索引ページが、GBP 内で検出されなかったために、プリフェッチャーによってディスクからローカル・バッファー・プールに読み込まれた回数。

表 1215. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の索引ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads )
 / pool_async_index_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グルー

プ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_index_lbp_pages_found - 非同期のローカル・バッファー・プールの検出索引ページ : モニター・エレメント

プリフェッチャーが索引ページにアクセスしようとして、その索引ページがローカル・バッファー・プールに存在した回数。

表 1216. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の索引ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads )
 / pool_async_index_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント

索引ページの非同期読み取り要求の数。

表 1217. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1218. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1219. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法 非同期要求当たりの索引ページ読み取りの数を計算するには、次の数式を使用します。

```
pool_async_index_reads / pool_async_index_read_reqs
```

この平均値は、プリフェッチャーとのそれぞれの対話で索引ページに関して行われる非同期入出力量を判別するのに役立ちます。

pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント

すべてのタイプの表スペースについて、非同期エンジン・ディスパッチ可能単位 (EDU) によって表スペース・コンテナー (物理) から読み取られた、索引ページの数を示します。

表 1220. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1221. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1222. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法

このエレメントを **pool_index_p_reads** モニター・エレメントと一緒に使用すると、同期的に実行された物理読み取りの数 (つまり、データベース・マネージャーのエージェントが実行した索引ページ物理読み取り数) を計算できます。次の数式を使用します。

```
1 - ((pool_data_p_reads + pool_index_p_reads) - (pool_async_data_reads + pool_async_index_reads))
 / (pool_data_l_reads + pool_index_l_reads)
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。このエレメントは、**num_ioservers** 構成パラメーターを調整するときに役に立ちます。

非同期読み取りは、データベース・マネージャーのプリフェッチャーが実行します。

pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント

非同期ページ・クリーナーまたはプリフェッチャーによりバッファー・プール索引ページがディスクに物理的に書き込まれた回数。プリフェッチャーは、プリフェッチするページの場所を作るために、ダーティー・ページをディスクに書き込む場合があります。

表 1223. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1224. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1225. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法

このエレメントを **pool_index_writes** モニター・エレメントと一緒に使用すると、同期的に実行された物理索引書き込み要求の数 (つまり、データベース・マネージャーのエージェントが実行した索引ページ物理書き込み数) を計算できます。次の数式を使用します。

```
pool_index_writes - pool_async_index_writes
```

非同期読み取り数と同期読み取り数を比較すると、バッファー・プール・ページ・クリーナーの動作状態がわかります。この比率は、 **num_iocleaners** 構成パラメーターを調整するときに役に立ちます。

pool_async_read_time バッファー・プール非同期読み取り時間

すべてのタイプの表スペースについて、非同期エンジン・ディスパッチ可能単位 (EDU) によって表スペース・コンテナー (物理) からデータ・ページおよび索引ページを読み取るために費やされた合計時間を示します。この値はミリ秒単位で示されます。

エレメント ID

pool_async_read_time

エレメント・タイプ

カウンター

表 1226. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1227. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1228. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法 このエレメントを使用すると同期読み取りの経過時間を計算できます。次の公式を使用します。

```
pool_read_time - pool_async_read_time
```

このエレメントを使用すると、平均非同期読み取り時間も計算できます。次の公式を使用します。

```
pool_async_read_time / pool_async_data_reads
```

これらの計算は、実行中の入出力操作を把握するときに使用します。

pool_async_write_time - バッファー・プール非同期書き込み時間 : モニター・エレメント

データベース・マネージャーのページ・クリーナーがデータ・ページまたは索引ページをバッファー・プールからディスクに書き込むのに要した合計経過時間。この値はミリ秒単位で報告されます。

表 1229. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1230. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1231. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法

同期によるページ書き込みでの経過時間を計算するには、次の公式を使用します。

```
pool_write_time - pool_async_write_time
```

このエレメントを使用すると、次の数式を使用して、平均非同期書き込み時間も計算できます。

```
pool_async_write_time
 / (pool_async_data_writes
   + pool_async_index_writes)
```

これらの計算は、実行中の入出力操作を把握するときに使用します。

pool_async_xda_gbp_indep_pages_found_in_lbp - 非同期 EDU によってローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属 XML ストレージ・オブジェクト (XDA) ページのモニター・エレメント

非同期 EDU によってローカル・バッファー・プールで検出された、グループ・バッファー・プール (GBP) に従属しない XML ストレージ・オブジェクト (XDA) ページの数。

表 1232. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1233. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

pool_async_xda_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・ページが、ローカル・バッファー・プールで無効のマークが付けられているため、プリフェッチャーによってグループ・バッファー・プールから要求された回数。

表 1234. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の XDA ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads )
 / pool_async_xda_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グルー

プ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_xda_gbp_l_reads - グループ・バッファー・プール XDA データの非同期論理読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) の GBP 従属のデータ・ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、プリフェッチャーがグループ・バッファー・プールから読み取ろうとした回数。

表 1235. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の XDA ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads )
 / pool_async_xda_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_xda_gbp_p_reads - グループ・バッファー・プール XDA データの非同期物理読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) の GBP 従属のデータ・ページが、GBP で検出されなかったために、プリフェッチャーがディスクからローカル・バッファー・プールに読み込んだ回数。

表 1236. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の XDA ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads )
 / pool_async_xda_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_xda_lbp_pages_found - 非同期のローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・ページが、プリフェッチャーによって要求されてローカル・バッファー・プールで検出された回数。

表 1237. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の XDA ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads )
 / pool_async_xda_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) データの非同期読み取り要求の数。

表 1238. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1239. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1240. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法 非同期要求当たりの XML ストレージ・オブジェクト・データ・ページ読み取りの平均数を計算するには、次の数式を使用します。

```
pool_async_xda_reads / pool_async_xda_read_reqs
```

この平均値は、各プリフェッチャーとのやりとりでの非同期入出力量を判別するのに利用します。

pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント

すべてのタイプの表スペースについて、非同期エンジン・ディスパッチ可能単位 (EDU) によって表スペース・コンテナー (物理) から読み取られた、XML ストレージ・オブジェクト (XDA) データ・ページの数を示します。

表 1241. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1242. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1243. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

表 1243. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表スペース	event_tablespace	常に収集される

使用法

pool_async_xda_reads および **pool_xda_p_reads** モニター・エレメントを使用して、XML ストレージ・オブジェクト・データ・ページ上で同期的に実行された物理読み取りの数 (つまり、データベース・マネージャーのエージェントが XML データに対して実行したデータ・ページの物理読み取り数) を計算します。次の数式を使用します。

```
pool_xda_p_reads - pool_async_xda_reads
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。このエレメントは、**num_ioservers** 構成パラメーターを調整するときに役に立ちます。

非同期読み取りは、データベース・マネージャーのプリフェッチャーが実行します。

pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント

非同期ページ・クリーナーまたはプリフェッチャーにより、XML ストレージ・オブジェクト (XDA) のバッファー・プール・データ・ページがディスクに物理的に書き込まれた回数。プリフェッチャーは、プリフェッチするページの場所を作るために、ダーティー・ページをディスクに書き込む場合があります。

表 1244. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1245. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1246. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法 このエレメントを **pool_xda_writes** モニター・エレメントと一緒に使用すると、XML ストレージ・オブジェクト・データ・ページ上で同期的に実行された物理書き込み要求の数 (つまり、データベース・マネージャーのエージェントが XML データに対して実行したデータ・ページの物理書き込み数) を計算できます。次の数式を使用します。

```
pool_xda_writes - pool_async_xda_writes
```

非同期読み取り数と同期読み取り数を比較すると、バッファー・プール・ページ・クリーナーの動作状態がわかります。この比率は、 **num_iocleaners** 構成パラメーターを調整するときに役に立ちます。

pool_config_size メモリー・プールの構成済みサイズ

DB2 データベース・システム内のメモリー・プールの内部構成済みのサイズです。値はバイト単位で示されます。

表 1247. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本
アプリケーション	memory_pool	基本

表 1248. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	常に収集される
接続	event_connmemuse	常に収集される

使用法 システム・メモリーの使用量を追跡するときに、この値と **pool_cur_size**、**pool_id**、および **pool_watermark** を組み合わせて使用します。

メモリー・プールがほぼ満杯状態になっているかどうかを確認するには、**pool_config_size** と **pool_cur_size** を比較します。例えば、ユーティリティー・ヒープが小さすぎるとします。この問題は、一定間隔でスナップショットを取り、スナップショット出力のユーティリティー・ヒープ・セクションを調べることによって診断できます。必要な場合は、メモリー不足の障害が起きないように、 **pool_cur_size** を **pool_config_size** より大きくすることもできます。大きくなる頻度が非常に少ない場合は、おそらく追加のアクションは必要ありません。しかし、常に **pool_cur_size** の値が **pool_config_size** の値に近いか大きい場合は、ユーティリティー・ヒープのサイズを大きくすることを考慮してください。

pool_cur_size メモリー・プールの現行サイズ

メモリー・プールの現行サイズ。値はバイト単位で示されます。

表 1249. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本
アプリケーション	memory_pool	基本

表 1250. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	常に収集される
接続	event_connmemuse	常に収集される

使用法 システム・メモリーの使用量を追跡するときに、この値と *pool_config_size*、 *pool_id*、および *pool_watermark* を組み合わせて使用します。

メモリー・プールがほぼ満杯状態になっているかどうかを確認するには、*pool_config_size* と *pool_cur_size* を比較します。例えば、ユーティリティー・ヒープが小さすぎるとします。この問題は、一定間隔でスナップショットを取り、スナップショット出力のユーティリティー・ヒープ・セクションを調べることによって診断できます。 *pool_cur_size* の値が *pool_config_size* の値に近い場合は、ユーティリティー・ヒープのサイズを大きくすることを考慮してください。

pool_data_gbp_indep_pages_found _in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント

エージェントによってローカル・バッファー・プール (LBP) で検出された、グループ・バッファー・プール (GBP) に従属しないデータ・ページの数。

表 1251. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 1251. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1252. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント

データ・ページがローカル・バッファー・プールで無効であったため、代わりにグループ・バッファー・プールから読み取られた回数。 DB2 pureScale 環境以外では、この値は NULL になります。

表 1253. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1253. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1254. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求されたデータ・ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_DATA_LBP_PAGES_FOUND - POOL_ASYNC_DATA_LBP_PAGES_FOUND) / POOL_DATA_L_READS
```

要求されたデータ・ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_DATA_GBP_L_READS - POOL_DATA_GBP_P_READS) / POOL_DATA_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属のデータ・ページが、ローカル・バッファー・プール (LBP) で無効であったか、存在しなかったため、グループ・バッファー・プールから読み取ろうとされた回数。

表 1255. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1255. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1256. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求されたデータ・ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_DATA_LBP_PAGES_FOUND - POOL_ASYNC_DATA_LBP_PAGES_FOUND) / POOL_DATA_L_READS
```

要求されたデータ・ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_DATA_GBP_L_READS - POOL_DATA_GBP_P_READS) / POOL_DATA_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属のデータ・ページが、GBP 内で検出されなかったためにディスクからローカル・バッファー・プールに読み込まれた回数。

表 1257. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1257. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1258. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求されたデータ・ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_DATA_LBP_PAGES_FOUND - POOL_ASYNC_DATA_LBP_PAGES_FOUND) / POOL_DATA_L_READS
```

要求されたデータ・ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_DATA_GBP_L_READS - POOL_DATA_GBP_P_READS) / POOL_DATA_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント

データ・ページがローカル・バッファー・プールで検出された回数。

表 1259. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1259. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1260. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求されたデータ・ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_DATA_LBP_PAGES_FOUND - POOL_ASYNC_DATA_LBP_PAGES_FOUND) / POOL_DATA_L_READS
```

要求されたデータ・ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_DATA_GBP_L_READS - POOL_DATA_GBP_P_READS) / POOL_DATA_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースのバッファー・プール (論理) から要求されたデータ・ページの数。

表 1261. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1261. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1262. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1263. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このカウントには、次のデータへのアクセスが含まれます。

- データベース・マネージャーがページの処理を必要としたときにバッファー・プールにすでにあるデータ。
- データベース・マネージャーがページを処理する前にバッファー・プールに読み取られたデータ。

pool_data_l_reads モニター・エレメントと **pool_data_p_reads** モニター・エレメントを使用すると、データ・ページのヒット率を計算できます。例えば、以下の数式によって、IBM DB2 pureScale Feature を使用しない DB2 環境におけるデータ・ページのヒット率が戻ります。

```
(pool_data_lbp_pages_found
- pool_async_data_lbp_pages_found - pool_temp_data_l_reads) / (pool_data_l_reads)
× 100
```

詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』 を参照してください。

バッファー・プール・サイズを大きくすると、一般的にヒット率は高くなりますが、ある点を超えると逆に低くなります。理想的には、データベース全体を保管できるような大きなバッファー・プールを割り振ることができれば、システムが稼働中のヒット率は 100% になります。しかし、現実的にはそうしたことは起こりません。実際には、使用するデータのサイズとそのデータへのアクセス方法によってヒット率の重要度は異なります。非常に大きなデータベースでアクセスが均等な場合は、ヒット率が低くなります。表が非常に大きな場合は、対応する方法はほとんどありません。

頻繁にアクセスされる小さめの表と索引のヒット率を上げるには、そうした表と索引を個別のバッファー・プールに割り当ててください。

pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースの表スペース・コンテナー (物理) から読み取られた、データ・ページの数を示します。

表 1264. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1264. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1265. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1266. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを **pool_data_l_reads** および **pool_async_data_reads** モニター・エレメントと一緒に使用して、同期的に実行された物理読み取りの数 (つまり、データベース・マネージャーのエージェントが実行したデータ・ページの物理読み取り数) を計算します。次の数式を使用します。

```
1 - ((pool_data_p_reads + pool_index_p_reads) - (pool_async_data_reads + pool_async_index_reads))
 / (pool_data_l_reads + pool_index_l_reads)
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。この情報は、**num_ioservers** 構成パラメーターを調整するときに役立つ場合があります。

pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント

バッファー・プールのデータ・ページが物理的にディスクに書き込まれた回数。

表 1267. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1267. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1268. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1269. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

pool_data_p_reads モニター・エレメントの値のパーセンテージが高いためにバッファー・プールのデータ・ページがディスクに書き込まれる場合は、データベースで利用可能なバッファー・プール・ページ数を増やすとパフォーマンスを改善できる可能性があります。

バッファー・プール・データ・ページをディスクに書き込む理由は次のとおりです。

- バッファー・プール内のページを解放して、次のページを読み取れるようにする。
- バッファー・プールを空にする。

システムがあるページを書き込むときに、新しいページのためのスペースを用意するとは限りません。そのページが更新されていなければ、単純に置換されます。このエレメントでは、このような置換はカウントされません。

データ・ページは、バッファー・プール・スペースが必要になる前に、非同期ページ・クリーナー・エージェントによって書き込まれることがあります。それについ

ては、**pool_async_data_writes** モニター・エレメントによって報告されます。これらの非同期ページ書き込みは、同期ページ書き込みと合わせて、このエレメントの値に含まれます。

このパーセンテージを計算するときは、バッファー・プールを最初に埋めるために必要となる物理読み取り数は無視してください。書き込みページ数は、次のように求めます。

1. アプリケーションを実行します (バッファーをロードするため)。
2. このエレメントの値を書き取ります。
3. アプリケーションを再び実行します。
4. このエレメントの新しい値からステップ 2 で記録した値を引きます。

アプリケーションを終了してから次に実行するまでのあいだにバッファー・プールの割り振りが解除されるのを防止するには、次のいずれかを行います。

- **ACTIVATE DATABASE** コマンドを使用してデータベースを活動化する。
- アイドル状態のアプリケーションをデータベースに接続する。

すべてのアプリケーションがデータベースを更新するような場合は、ほとんどのバッファー・プール・ページが更新されたデータを含んでおり、これをディスクに書き込む必要があるので、バッファー・プールのサイズを大きくしてもパフォーマンスはあまり改善されません。ただし、更新されたページを書き出す前に、ほかの作業単位がこのページを使用できる場合は、バッファー・プールが書き込み操作と読み取り操作を節約できるので、パフォーマンスが向上する場合があります。

pool_drty_pg_steal_clns - 起動されたバッファー・プール・ビクティム・ページ・クリーナー : モニター・エレメント

データベースのビクティム・バッファー置換の際に同期書き込みが必要になりページ・クリーナーが呼び出された回数。

表 1270. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 1271. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1272. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

次の数式を使用すると、クリーナー呼び出しの合計に占めるこのエレメントのパーセンテージを計算できます。

```
   pool_drty_pg_steal_clns
/ (pool_drty_pg_steal_clns
 + pool_drty_pg_thrsh_clns
 + pool_lsn_gap_clns)
```

この比率が低い場合は、定義したページ・クリーナー数が多すぎることを示します。 **chngpgs_thresh** 構成パラメーターをあまり低く設定すると、後にダーティー・ページになるページを書き出す可能性があります。クリーニングをあまり積極的にすると、バッファー・プールの 1 つの目的である、書き込みをできるだけ遅らせることができなくなります。

この比率が高い場合は、定義したページ・クリーナーの数が十分でないことを示している可能性があります。十分なページ・クリーナーがないと、障害時のリカバリー時間が長くなります。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が OFF の場合は、以下のようになります。

- **pool_drty_pg_steal_clns** モニター・エレメントがモニター・ストリーム中に挿入される。
- **pool_drty_pg_steal_clns** モニター・エレメントが、データベースのビクティム・バッファー置換の際に同期書き込みが必要になり、ページ・クリーナーが呼び出された回数をカウントする。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が ON の場合は、以下のようになります。

- **pool_drty_pg_steal_clns** モニター・エレメントが、モニター・ストリーム中に 0 を挿入する。
- ビクティム・バッファー置換の際に同期書き込みが必要でも、ページ・クリーナーは明示的に起動されない。データベースまたは特定のバッファー・プール用に構成されているページ・クリーナーの数が適切かどうかを判別するには、**pool_no_victim_buffer** モニター・エレメントを参照してください。

注: ダーティー・ページはディスクに書き出されますが、バッファー・プールに新しいページを読み取るためのスペースが必要な場合を除いて、このページはバッファー・プールからすぐには除去されません。

pool_drty_pg_thrsh_clns - 起動されたバッファー・プールしきい値クリーナー : モニター・エレメント

バッファー・プールがデータベースのダーティー・ページしきい値基準に達したためにページ・クリーナーが呼び出された回数。

表 1273. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 1274. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1275. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 しきい値は **chngpgs_thresh** 構成パラメーターによって設定されます。この値は、バッファー・プール・サイズに適用されるパーセンテージです。プール内のダーティー・ページ数がこの値を超えると、クリーナーを起動します。

chngpgs_thresh 構成パラメーターの設定値が低すぎると、ページの書き出しが早すぎて、再読み取りが必要になる可能性があります。設定値が高すぎると、累積されるページ数が多くなり、ページを同期的に書き出す必要が生じる場合があります。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が OFF の場合は、以下のようになります。

- **pool_drty_pg_thrsh_clns** モニター・エレメントがモニター・ストリーム中に挿入される。
- **pool_drty_pg_thrsh_clns** モニター・エレメントが、バッファー・プールがデータベースのダーティー・ページしきい値基準に達したためにページ・クリーナーが呼び出された回数をカウントする。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が ON の場合は、以下のようになります。

- **pool_drty_pg_thrsh_clns** モニター・エレメントがモニター・ストリーム中に 0 を挿入する。
- ページ・クリーナーは、基準値によって起動されるのを待たず、常時アクティブで、使用可能なビクティム用の空きバッファーが十分あるか確認する。

pool_failed_async_data_reqs - 失敗したデータ・プリフェッチ要求のモニター・エレメント

データ・プリフェッチ要求をキューに入れようとして失敗した回数。原因の 1 つとしては、プリフェッチ・キューが満杯であることが考えられます。

表 1276. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1276. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1277. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_failed_async_..._reqs** エレメントと一緒に、プリフェッチ・キューに追加できなかったプリフェッチ要求数を示します。プリフェッチ・キューが小さすぎる場合や、プリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。プリフェッチ・キューに要求を追加できない場合、データベース・エージェントは、通常、ディスク入出力を同期的に実行しますが、これはプリフェッチよりも効率が劣ります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
```

```
  )
+
  (
  POOL_QUEUED_ASYNC_DATA_REQS +
  POOL_QUEUED_ASYNC_INDEX_REQS +
  POOL_QUEUED_ASYNC_XDA_REQS +
  POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
  POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
  POOL_QUEUED_ASYNC_TEMP_XDA_REQS
  )
) × 100
```

この数式は、生成された要求の総数に対する、成功したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。多数の要求が作成された場合、または、不適切な構成やチューニングが原因でプリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。成功した要求の割合が低い場合、プリフェッチ・メカニズムのボトルネックを示している可能性があります。場合によっては、構成パラメーター **num_ioservers** の値を変更して、より多くのプリフェッチャーを構成する必要があります。プリフェッチ・キューが満杯になるという状況は、エージェントが小さな要求を大量に実行している場合にも発生する可能性があります。関連のモニター・エレメント **pool_queued_async_..._pages** および **pool_queued_async_..._reqs** を使用すると、プリフェッチ要求の平均サイズを調べることができます。

pool_failed_async_index_reqs - 失敗した索引プリフェッチ要求のモニター・エレメント

索引プリフェッチ要求をキューに入れようとして失敗した回数。原因の 1 つとしては、プリフェッチ・キューが満杯で、要求によりフリー・リストから結果を取得できなかったことが考えられます。

表 1278. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1278. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1279. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_failed_async_..._reqs** エレメントと一緒に、プリフェッチ・キューに追加できなかったプリフェッチ要求数を示します。プリフェッチ・キューが小さすぎる場合や、プリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。プリフェッチ・キューに要求を追加できない場合、データベース・エージェントは、通常、ディスク入出力を同期的に実行しますが、これはプリフェッチよりも効率が劣ります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS
   )
 ) × 100
```

この数式は、生成された要求の総数に対する、成功したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。多数の要求が作成された場合、または、不適切な構成やチューニングが原因でプリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。成功した要求の割合が低い場合、プリフェッチ・メカニズムのボトルネックを示している可能性があります。場合によっては、構成パラメーター **num_ioservers** の値を変更して、より多くのプリフェッチャーを構成する必要があります。プリフェッチ・キューが満杯になるという状況は、エージェントが小さな要求を大量に実行している場合にも発生する可能性があります。関連のモニター・エレメント **pool_queued_async_..._pages** および **pool_queued_async_..._reqs** を使用すると、プリフェッチ要求の平均サイズを調べることができます。

pool_failed_async_other_reqs - 失敗した非プリフェッチの要求のモニター・エレメント

非プリフェッチの要求をキューに入れようとして失敗した回数。このエレメントは、プリフェッチャーによって行われる非プリフェッチの処理に関するエレメントです。要求が失敗した原因の 1 つとして、プリフェッチ・キューが満杯であったことが考えられます。

表 1280. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1280. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1281. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントは、アクセス・プランから指示されたプリフェッチとは関係がない処理のための要求のうち、プリフェッチ・キューに追加できなかった要求の数を報告します。バックアップ・ユーティリティーなどのユーティリティーは、プリフェッチャーのメカニズムを使用してユーティリティーのタスクを実行します。ただし、その使用方法は、SQL ステートメントのアクセス・プランによる使用方法とは異なっています。要求をプリフェッチ・キューに追加できなかった原因は、キューが満杯だったからである可能性があります。

pool_failed_async_temp_data_reqs - 失敗した TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント

TEMPORARY 表スペースのデータ・プリフェッチ要求をキューに入れようとして失敗した回数。原因の 1 つとしては、プリフェッチ・キューが満杯で、要求によりフリー・リストから結果を取得できなかったことが考えられます。

表 1282. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1282. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1283. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_failed_async_..._reqs** エレメントと一緒に、プリフェッチ・キューに追加できなかったプリフェッチ要求数を示します。プリフェッチ・キューが小さすぎる場合や、プリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。プリフェッチ・キューに要求を追加できない場合、データベース・エージェントは、通常、ディスク入出力を同期的に実行しますが、これはプリフェッチよりも効率が劣ります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
```

```
    POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
    POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) × 100
```

この数式は、生成された要求の総数に対する、成功したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。多数の要求が作成された場合、または、不適切な構成やチューニングが原因でプリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。成功した要求の割合が低い場合、プリフェッチ・メカニズムのボトルネックを示している可能性があります。場合によっては、構成パラメーター **num_ioservers** の値を変更して、より多くのプリフェッチャーを構成する必要があります。プリフェッチ・キューが満杯になるという状況は、エージェントが小さな要求を大量に実行している場合にも発生する可能性があります。関連のモニター・エレメント **pool_queued_async_..._pages** および **pool_queued_async_..._reqs** を使用すると、プリフェッチ要求の平均サイズを調べることができます。

pool_failed_async_temp_index_reqs - 失敗した TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント

TEMPORARY 表スペースの索引プリフェッチ要求をキューに入れようとして失敗した回数。原因の 1 つとしては、プリフェッチ・キューが満杯で、要求によりフリー・リストから結果を取得できなかったことが考えられます。

表 1284. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1284. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1285. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_failed_async_..._reqs** エレメントと一緒に、プリフェッチ・キューに追加できなかったプリフェッチ要求数を示します。プリフェッチ・キューが小さすぎる場合や、プリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。プリフェッチ・キューに要求を追加できない場合、データベース・エージェントは、通常、ディスク入出力

を同期的に実行しますが、これはプリフェッチよりも効率が劣ります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS
   )
 ) × 100
```

この数式は、生成された要求の総数に対する、成功したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。多数の要求が作成された場合、または、不適切な構成やチューニングが原因でプリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。成功した要求の割合が低い場合、プリフェッチ・メカニズムのボトルネックを示している可能性があります。場合によっては、構成パラメーター **num_ioservers** の値を変更して、より多くのプリフェッチャーを構成する必要があります。プリフェッチ・キューが満杯になるという状況は、エージェントが小さな要求を大量に実行している場合にも発生する可能性があります。関連のモニター・エレメント **pool_queued_async_..._pages** および **pool_queued_async_..._reqs** を使用すると、プリフェッチ要求の平均サイズを調べることができます。

pool_failed_async_temp_xda_reqs - 失敗した TEMPORARY 表スペースの XDA プリフェッチ要求のモニター・エレメント

TEMPORARY 表スペースの XML ストレージ・オブジェクト (XDA) のデータ・プリフェッチ要求をキューに入れようとして失敗した回数。原因の 1 つとしては、プリフェッチ・キューが満杯で、要求によりフリー・リストから結果を取得できなかったことが考えられます。

表 1286. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1287. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_failed_async_..._reqs** エレメントと一緒に、プリフェッチ・キューに追加できなかったプリフェッチ要求数を示します。プリフェッチ・キューが小さすぎる場合や、プリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。プリフェッチ・キューに要求を追加できない場合、データベース・エージェントは、通常、ディスク入出力を同期的に実行しますが、これはプリフェッチよりも効率が劣ります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
```

```
    POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
    POOL_QUEUED_ASYNC_TEMP_XDA_REQS
    )
  ) × 100
```

この数式は、生成された要求の総数に対する、成功したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。多数の要求が作成された場合、または、不適切な構成やチューニングが原因でプリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。成功した要求の割合が低い場合、プリフェッチ・メカニズムのボトルネックを示している可能性があります。場合によっては、構成パラメーター **num_ioservers** の値を変更して、より多くのプリフェッチャーを構成する必要があります。プリフェッチ・キューが満杯になるという状況は、エージェントが小さな要求を大量に実行している場合にも発生する可能性があります。関連のモニター・エレメント **pool_queued_async_..._pages** および **pool_queued_async_..._reqs** を使用すると、プリフェッチ要求の平均サイズを調べることができます。

pool_failed_async_xda_reqs - 失敗した XDA プリフェッチ要求のモニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・プリフェッチ要求をキューに入れようとして失敗した回数。原因の 1 つとしては、プリフェッチ・キューが満杯で、要求によりフリー・リストから結果を取得できなかったことが考えられます。

表 1288. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1288. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1289. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_failed_async_..._reqs** エレメントと一緒に、プリフェッチ・キューに追加できなかったプリフェッチ要求数を示します。プリフェッチ・キューが小さすぎる場合や、プリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。プリフェッチ・キューに要求を追加できない場合、データベース・エージェントは、通常、ディスク入出力

を同期的に実行しますが、これはプリフェッチよりも効率が劣ります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS
   )
 ) × 100
```

この数式は、生成された要求の総数に対する、成功したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。多数の要求が作成された場合、または、不適切な構成やチューニングが原因でプリフェッチャーの実行が遅すぎる場合に、プリフェッチ・キューに要求を追加できないことがあります。成功した要求の割合が低い場合、プリフェッチ・メカニズムのボトルネックを示している可能性があります。場合によっては、構成パラメーター **num_ioservers** の値を変更して、より多くのプリフェッチャーを構成する必要があります。プリフェッチ・キューが満杯になるという状況は、エージェントが小さな要求を大量に実行している場合にも発生する可能性があります。関連のモニター・エレメント **pool_queued_async_..._pages** および **pool_queued_async_..._reqs** を使用すると、プリフェッチ要求の平均サイズを調べることができます。

pool_id メモリー・プール ID

メモリー・プールのタイプ。

表 1290. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本

表 1290. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	memory_pool	基本

表 1291. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	-
接続	event_connmemuse	-

使用法

システム・メモリーの使用量を追跡するときに、この値と **pool_config_size**、**pool_cur_size**、および **pool_watermark** を組み合わせて使用します。

pool_id を使用すると、システム・モニター出力に示されているメモリー・プールを識別できます。sqlmon.h に、さまざまなメモリー・プール ID があります。通常の操作条件では、次に示すプールの 1 つ以上が該当します。

API 定数	説明
SQLM_HEAP_APPLICATION	アプリケーション・ヒープ
SQLM_HEAP_DATABASE	データベース・ヒープ
SQLM_HEAP_LOCK_MGR	ロック・マネージャー・ヒープ
SQLM_HEAP_UTILITY	バックアップ/リストア/ユーティリティー・ヒープ
SQLM_HEAP_STATISTICS	統計ヒープ
SQLM_HEAP_PACKAGE_CACHE	パッケージ・キャッシュ・ヒープ
SQLM_HEAP_CAT_CACHE	カタログ・キャッシュ・ヒープ
SQLM_HEAP_MONITOR	データベース・モニター・ヒープ
SQLM_HEAP_STATEMENT	ステートメント・ヒープ
SQLM_HEAP_FCMBP	FCMBP ヒープ
SQLM_HEAP_IMPORT_POOL	インポート・プール
SQLM_HEAP_OTHER	その他のメモリー
SQLM_HEAP_BP	バッファー・プール・ヒープ
SQLM_HEAP_APPL_SHARED	アプリケーション共有ヒープ
SQLM_HEAP_SHARED_SORT	ソート共有ヒープ

pool_index_gbp_indep_pages_found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント

エージェントによってローカル・バッファー・プール (LBP) で検出された、グループ・バッファー・プール (GBP) に従属しない索引ページの数。

表 1292. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1292. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1293. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント

ローカル・バッファー・プール内のページが無効であったために、グループ・バッファー・プールから索引ページを読み取ろうとした回数。

表 1294. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 1294. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1295. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求された索引ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_INDEX_LBP_PAGES_FOUND - POOL_ASYNC_INDEX_LBP_PAGES_FOUND) / POOL_INDEX_L_READS
```

要求された索引ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_INDEX_GBP_L_READS - POOL_INDEX_GBP_P_READS) / POOL_INDEX_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属の索引ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、グループ・バッファー・プールから読み取ろうとされた回数。

表 1296. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1296. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1297. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求された索引ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_INDEX_LBP_PAGES_FOUND - POOL_ASYNC_INDEX_LBP_PAGES_FOUND) / POOL_INDEX_L_READS
```

要求された索引ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_INDEX_GBP_L_READS - POOL_INDEX_GBP_P_READS) / POOL_INDEX_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属の索引ページが、GBP 内で検出されなかったためにディスクからローカル・バッファー・プールに読み込まれた回数。

表 1298. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1298. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1299. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求された索引ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_INDEX_LBP_PAGES_FOUND - POOL_ASYNC_INDEX_LBP_PAGES_FOUND) / POOL_INDEX_L_READS
```

要求された索引ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_INDEX_GBP_L_READS - POOL_INDEX_GBP_P_READS) / POOL_INDEX_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント

索引ページがローカル・バッファー・プールに存在していた回数。

表 1300. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1300. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1301. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求された索引ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_INDEX_LBP_PAGES_FOUND - POOL_ASYNC_INDEX_LBP_PAGES_FOUND) / POOL_INDEX_L_READS
```

要求された索引ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_INDEX_GBP_L_READS - POOL_INDEX_GBP_P_READS) / POOL_INDEX_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースのバッファー・プール (論理) から要求された、索引ページの数を示します。

表 1302. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1302. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1303. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1304. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このカウントには、次の索引ページへのアクセスが含まれます。

- データベース・マネージャーがページの処理を必要としたときにバッファー・プールにすでにあるデータ。
- データベース・マネージャーがページを処理する前にバッファー・プールに読み取られたデータ。

pool_index_p_reads と **pool_async_index_reads** を組み合わせて使用すると、**pool_index_l_reads** によってバッファー・プールの索引ページ・ヒット率を計算できます。例えば、以下の数式によって、IBM DB2 pureScale Feature を使用しない DB2 環境における索引ページ・ヒット率が戻ります。

```
((pool_index_lbp_pages_found
- pool_async_index_lbp_pages_found - pool_temp_index_l_reads)
/ (pool_index_l_reads) × 100
```

詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

ヒット率が低い場合は、バッファー・プール・ページ数を増やすと、パフォーマンスが向上する場合があります。

pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースの表スペース・コンテナー (物理) から読み取られた、索引ページの数を示します。

表 1305. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1305. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1306. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1307. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

pool_index_l_reads と **pool_async_index_reads** を組み合わせて使用すると、**pool_index_p_reads** によってバッファー・プールの索引ページ・ヒット率を計算できます。例えば、以下の数式によって、IBM DB2 pureScale Feature を使用しない DB2 環境における索引ページ・ヒット率が戻ります。

```
((pool_index_lbp_pages_found
- pool_async_index_lbp_pages_found - pool_temp_index_l_reads)
/ (pool_index_l_reads) × 100
```

詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント

バッファー・プール索引ページがディスクに物理的に書き込まれた回数。

表 1308. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1308. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1309. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1310. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

データ・ページと同様に、バッファー・プール索引ページは以下の理由でディスクに書き込まれます。

- バッファー・プール内のページを解放して、次のページを読み取れるようにする。
- バッファー・プールを空にする。

システムがあるページを書き込むときに、新しいページのためのスペースを用意するとは限りません。そのページが更新されていなければ、単純に置換されます。このエレメントでは、このような置換はカウントされません。

索引ページは、バッファー・プール・スペースが必要になる前に、非同期ページ・クリーナー・エージェントにより書き込まれます。非同期索引ページの書き込みは、同期索引ページの書き込みと合わせて、このエレメントの値に含まれます (**pool_async_index_writes** モニター・エレメントを参照)。

pool_index_p_reads モニター・エレメントの値のパーセンテージが高いためにバッファー・プールの索引ページがディスクに書き込まれる場合は、データベースで利用可能なバッファー・プール・ページ数を増やすとパフォーマンスを改善できる可能性があります。

このパーセンテージを計算するときは、バッファー・プールを最初に埋めるために必要となる物理読み取り数は無視してください。書き込みページ数は、次のように求めます。

1. アプリケーションを実行します (バッファーをロードするため)。
2. このエレメントの値を書き取ります。
3. アプリケーションを再び実行します。
4. このエレメントの新しい値からステップ 2 で記録した値を引きます。

アプリケーションを終了してから次に実行するまでのあいだにバッファー・プールの割り振りが解除されるのを防止するには、次のいずれかを行います。

- **ACTIVATE DATABASE** コマンドを使用してデータベースを活動化する。
- アイドル状態のアプリケーションをデータベースに接続する。

すべてのアプリケーションがデータベースを更新するような場合は、ほとんどのページが更新されたデータを含んでおり、これをディスクに書き込む必要があるので、バッファー・プールのサイズを大きくしてもパフォーマンスはあまり改善されません。

pool_lsn_gap_clns - 起動されたバッファー・プール・ログ・スペース・クリーナー : モニター・エレメント

使用されているロギング・スペースがデータベースの定義済み基準に達したためにページ・クリーナーが呼び出された回数。

表 1311. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 1312. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1313. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このエレメントは、ロギングに十分なスペースがあるかどうか、またログ・ファイルやさらに大きなログ・ファイルの追加が必要かどうかを判別するときに利用できます。

ページ・クリーニングの基準は、 **softmax** 構成パラメーターの設定値により決定されます。バッファー・プール内の最も古いページに含まれている更新内容が現行ログ位置と比較して基準値よりも古いログ・レコードにより記述されている場合に、ページ・クリーナーが起動される。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が OFF の場合は、以下のようになります。

- **pool_lsn_gap_clns** モニター・エレメントがモニター・ストリーム中に挿入される。
- バッファー・プール内の最も古いページに含まれている更新内容が現行ログ位置と比較して基準値よりも古いログ・レコードにより記述されている場合に、ページ・クリーナーが起動される。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が ON の場合は、以下のようになります。

- **pool_lsn_gap_clns** モニター・エレメントがモニター・ストリーム中に 0 を挿入する。
- ページ・クリーナーが、基準値によって起動されるのを待たずに、先行してページを書き込む。

pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント

エージェントに、事前選択された使用可能なビクティム・バッファーがなかった回数。

表 1314. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 1315. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1316. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

使用法 このエレメントは、ページ・クリーニングを先行して行う際に、特定のバッファー・プールに十分なページ・クリーナーがあるかどうかを判断するときに利用できます。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が ON の場合、pool_no_victim_buffer エレメントは、エージェントが即時使用に対応する事前選択されたビクティム・バッファーを見つけることができずに、バッファー・プールで適切なビクティム・バッファーを検索することを余儀なくされた回数をカウントします。

pool_no_victim_buffer エレメントの値が、バッファー・プールへの論理読み取りの数と比べて大きい場合、DB2 データベース・システムは、十分な数の適切なビクティムを確実に使用可能にしておくのが難しくなります。ページ・クリーナーの数を増やすと、DB2 が事前選択されたビクティム・バッファーを提供する能力も向上します。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が OFF の場合、pool_no_victim_buffer エレメントは予測値を持つわけではないので、無視しても問題ありません。この構成の場合、DB2 データベース・システムは、事前選択されたビクティム・バッファーをエージェントに対して使用可能にしておこうとしないため、バッファー・プールにアクセスするときには、大抵の場合、エージェントがバッファー・プールでビクティム・バッファーを検索する必要が生じます。

pool_queued_async_data_pages - データ・ページ・プリフェッチ要求のモニター・エレメント

正常にプリフェッチが要求されたデータ・ページの数。

表 1317. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1317. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1318. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 1318. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントは、他の **pool_queued_async_..._pages** エレメントと一緒に、プリフェッチ要求によって取り出されたデータ・ページ数を示します。この情報を使用すると、システム上でプリフェッチ要求が効率的に実行されているかどうかを調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ要求あたりの平均ページ数を計算できます。

```
  (POOL_QUEUED_ASYNC_DATA_PAGES +
   POOL_QUEUED_ASYNC_INDEX_PAGES +
   POOL_QUEUED_ASYNC_XDA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_DATA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES +
   POOL_QUEUED_ASYNC_TEMP_XDA_PAGES)
÷
  (POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS)
```

要求あたりの平均ページ数が低く、かつ、システム上で大量のプリフェッチが行われている場合は、必要以上の入出力操作がシステムで実行されている可能性があります。一般的に、要求サイズはプリフェッチ・サイズに基づいており、プリフェッチ・サイズは、少なくともエクステント・サイズと同じサイズでなければなりません。このため、平均要求サイズが小さいということは、プリフェッチ・サイズの設定が小さすぎること、また、プリフェッチ・サイズをエクステント・サイズの倍数に増やせばパフォーマンスを改善できることを示している可能性があります。また、平均要求サイズが小さいということは、プリフェッチ・キューがすぐに満杯になることを示している可能性もあります。そのため、関連する **pool_failed_async_..._reqs** モニター・エレメントもモニターすると役に立ちます。

pool_queued_async_data_reqs - データ・プリフェッチ要求のモニター・エレメント

プリフェッチ・キューに正常に追加されたデータ・プリフェッチ要求の数。

表 1319. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1319. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1320. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE

表 1320. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_queued_async_*_reqs** エレメントと一緒に、プリフェッチ・キューに追加されたプリフェッチ要求数を示します。この情報を使用すると、データベース・マネージャーでプリフェッチが行われる頻度が分かります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS
   )
 ) * 100
```

この数式は、生成された要求の総数に対する、失敗したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。このパーセンテージが低い場合は、時により、

num_ioservers 構成パラメーターを変更して、より多くのプリフェッチャーを構成する必要があります。

pool_queued_async_index_pages - 索引ページ・プリフェッチ要求のモニター・エレメント

正常にプリフェッチが要求された索引ページの数。

表 1321. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1321. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1322. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントは、他の **pool_queued_async_..._pages** エレメントと一緒に、プリフェッチ要求によって取り出されたデータ・ページ数を示します。この情報を使用すると、システム上でプリフェッチ要求が効率的に実行されているかどうかを調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ要求あたりの平均ページ数を計算できます。

```
  (POOL_QUEUED_ASYNC_DATA_PAGES +
   POOL_QUEUED_ASYNC_INDEX_PAGES +
   POOL_QUEUED_ASYNC_XDA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_DATA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES +
   POOL_QUEUED_ASYNC_TEMP_XDA_PAGES)
÷
  (POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS)
```

要求あたりの平均ページ数が低く、かつ、システム上で大量のプリフェッチが行われている場合は、必要以上の入出力操作がシステムで実行されている可能性があります。一般的に、要求サイズはプリフェッチ・サイズに基づいており、プリフェッチ・サイズは、少なくともエクステント・サイズと同じサイズでなければなりません。このため、平均要求サイズが小さいということは、プリフェッチ・サイズの設定が小さすぎること、また、プリフェッチ・サイズをエクステント・サイズの倍数

に増やせばパフォーマンスを改善できることを示している可能性があります。また、平均要求サイズが小さいということは、プリフェッチ・キューがすぐに満杯になることを示している可能性もあります。そのため、関連する **pool_failed_async_..._reqs** モニター・エレメントもモニターすると役に立ちます。

pool_queued_async_index_reqs - 索引プリフェッチ要求のモニター・エレメント

プリフェッチ・キューに正常に追加された索引プリフェッチ要求の数。

表 1323. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 1323. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1324. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_queued_async_*_reqs** エレメントと一緒に、プリフェッチ・キューに追加されたプリフェッチ要求数を示します。この情報を使用すると、データベース・マネージャーでプリフェッチが行われる頻度が分かります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
```

```
  POOL_FAILED_ASYNC_TEMP_DATA_REQS +
  POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
  POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
+
  (
  POOL_QUEUED_ASYNC_DATA_REQS +
  POOL_QUEUED_ASYNC_INDEX_REQS +
  POOL_QUEUED_ASYNC_XDA_REQS +
  POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
  POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
  POOL_QUEUED_ASYNC_TEMP_XDA_REQS
  )
) * 100
```

この数式は、生成された要求の総数に対する、失敗したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。このパーセンテージが低い場合は、時により、**num_ioservers** 構成パラメーターを変更して、より多くのプリフェッチャーを構成する必要があります。

pool_queued_async_other_reqs - プリフェッチャーが処理したその他の要求モニター・エレメント

プリフェッチ・キューに正常に追加された非プリフェッチ処理の要求の数。これは、プリフェッチャーによって行われたその他の処理に関するものです。

表 1325. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1325. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1326. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントは、アクセス・プランから指示されたプリフェッチとは関係がない入出力処理用に、プリフェッチ・キューに追加された要求数を報告します。バックアップ・ユーティリティーなどのユーティリティーは、プリフェッチャーのメカニズムを使用してユーティリティーのタスクを実行します。ただし、その使用方法は、SQL ステートメントのアクセス・プランによる使用方法とは異なっています。

pool_queued_async_temp_data_pages - TEMPORARY 表スペースのデータ・ページ・プリフェッチ要求のモニター・エレメント

正常にプリフェッチが要求された TEMPORARY 表スペースのデータ・ページの数。

表 1327. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1327. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1328. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントは、他の **pool_queued_async_..._pages** エレメントと一緒に、プリフェッチ要求によって取り出されたデータ・ページ数を示します。この情報を使用すると、システム上でプリフェッチ要求が効率的に実行されているかどうかを調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ要求あたりの平均ページ数を計算できます。

```
  (POOL_QUEUED_ASYNC_DATA_PAGES +
   POOL_QUEUED_ASYNC_INDEX_PAGES +
   POOL_QUEUED_ASYNC_XDA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_DATA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES +
   POOL_QUEUED_ASYNC_TEMP_XDA_PAGES)
÷
  (POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
```

```
POOL_QUEUED_ASYNC_XDA_REQS +
POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
POOL_QUEUED_ASYNC_TEMP_XDA_REQS)
```

要求あたりの平均ページ数が低く、かつ、システム上で大量のプリフェッチが行われている場合は、必要以上の入出力操作がシステムで実行されている可能性があります。一般的に、要求サイズはプリフェッチ・サイズに基づいており、プリフェッチ・サイズは、少なくともエクステント・サイズと同じサイズでなければなりません。このため、平均要求サイズが小さいということは、プリフェッチ・サイズの設定が小さすぎること、また、プリフェッチ・サイズをエクステント・サイズの倍数に増やせばパフォーマンスを改善できることを示している可能性があります。また、平均要求サイズが小さいということは、プリフェッチ・キューがすぐに満杯になることを示している可能性もあります。そのため、関連する **pool_failed_async_..._reqs** モニター・エレメントもモニターすると役に立ちます。

pool_queued_async_temp_data_reqs - TEMPORARY 表スペースのデータ・プリフェッチ要求のモニター・エレメント

プリフェッチ・キューに正常に追加された、 TEMPORARY 表スペースのデータ・プリフェッチ要求の数。

表 1329. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1329. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1330. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_queued_async_*_reqs** エレメントと一緒に、プリフェッチ・キューに追加されたプリフェッチ要求数を示します。この情報を使用すると、データベース・マネージャーでプリフェッチが行われる頻度が分かります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができま

す。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS
   )
 ) * 100
```

この数式は、生成された要求の総数に対する、失敗したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。このパーセンテージが低い場合は、時により、**num_ioservers** 構成パラメーターを変更して、より多くのプリフェッチャーを構成する必要があります。

pool_queued_async_temp_index_pages - TEMPORARY 表スペースの索引ページ・プリフェッチ要求のモニター・エレメント

正常にプリフェッチが要求された TEMPORARY 表スペースの索引ページの数。

表 1331. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 1331. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1332. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

表 1332. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントは、他の **pool_queued_async_..._pages** エレメントと一緒に、プリフェッチ要求によって取り出されたデータ・ページ数を示します。この情報を使用すると、システム上でプリフェッチ要求が効率的に実行されているかどうかを調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ要求あたりの平均ページ数を計算できます。

```
  (POOL_QUEUED_ASYNC_DATA_PAGES +
   POOL_QUEUED_ASYNC_INDEX_PAGES +
   POOL_QUEUED_ASYNC_XDA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_DATA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES +
   POOL_QUEUED_ASYNC_TEMP_XDA_PAGES)
÷
  (POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS)
```

要求あたりの平均ページ数が低く、かつ、システム上で大量のプリフェッチが行われている場合は、必要以上の入出力操作がシステムで実行されている可能性があります。一般的に、要求サイズはプリフェッチ・サイズに基づいており、プリフェッチ・サイズは、少なくともエクステント・サイズと同じサイズでなければなりません。このため、平均要求サイズが小さいということは、プリフェッチ・サイズの設定が小さすぎること、また、プリフェッチ・サイズをエクステント・サイズの倍数に増やせばパフォーマンスを改善できることを示している可能性があります。また、平均要求サイズが小さいということは、プリフェッチ・キューがすぐに満杯になることを示している可能性もあります。そのため、関連する **pool_failed_async_..._reqs** モニター・エレメントもモニターすると役に立ちます。

pool_queued_async_temp_index_reqs - TEMPORARY 表スペースの索引プリフェッチ要求のモニター・エレメント

プリフェッチ・キューに正常に追加された、 TEMPORARY 表スペースの索引プリフェッチ要求の数。

表 1333. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1333. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1334. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_queued_async_*_reqs** エレメントと一緒に、プリフェッチ・キューに追加されたプリフェッチ要求数を示します。この情報を使用すると、データベース・マネージャーでプリフェッチが行われる頻度が分かります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS
   )
 ) * 100
```

この数式は、生成された要求の総数に対する、失敗したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。このパーセンテージが低い場合は、時により、**num_ioservers** 構成パラメーターを変更して、より多くのプリフェッチャーを構成する必要があります。

pool_queued_async_temp_xda_pages - TEMPORARY 表スペースの XDA データ・ページ・プリフェッチ要求のモニター・エレメント

正常にプリフェッチが要求された TEMPORARY 表スペースの XML ストレージ・オブジェクト (XDA) のデータ・ページの数。

表 1335. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1335. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1336. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントは、他の **pool_queued_async_..._pages** エレメントと一緒に、プリフェッチ要求によって取り出されたデータ・ページ数を示します。この情報を使用すると、システム上でプリフェッチ要求が効率的に実行されているかどうかを調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ要求あたりの平均ページ数を計算できます。

```
  (POOL_QUEUED_ASYNC_DATA_PAGES +
   POOL_QUEUED_ASYNC_INDEX_PAGES +
   POOL_QUEUED_ASYNC_XDA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_DATA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES +
   POOL_QUEUED_ASYNC_TEMP_XDA_PAGES)
÷
  (POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS)
```

要求あたりの平均ページ数が低く、かつ、システム上で大量のプリフェッチが行われている場合は、必要以上の入出力操作がシステムで実行されている可能性があります。一般的に、要求サイズはプリフェッチ・サイズに基づいており、プリフェッチ・サイズは、少なくともエクステント・サイズと同じサイズでなければなりません。このため、平均要求サイズが小さいということは、プリフェッチ・サイズの設定が小さすぎること、また、プリフェッチ・サイズをエクステント・サイズの倍数に増やせばパフォーマンスを改善できることを示している可能性があります。また、平均要求サイズが小さいということは、プリフェッチ・キューがすぐに満杯になることを示している可能性もあります。そのため、関連する **pool_failed_async_..._reqs** モニター・エレメントもモニターすると役に立ちます。

pool_queued_async_temp_xda_reqs - TEMPORARY 表スペースの XDA データ・プリフェッチ要求のモニター・エレメント

プリフェッチ・キューに正常に追加された、 TEMPORARY 表スペースの XML ストレージ・オブジェクト (XDA) のデータ・プリフェッチ要求の数。

表 1337. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 1337. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1338. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_queued_async_*_reqs** エレメントと一緒に、プリフェッチ・キューに追加されたプリフェッチ要求数を示します。この情報を使用すると、データベース・マネージャーでプリフェッチが行われる頻度が分かります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
```

```
POOL_FAILED_ASYNC_XDA_REQS +
POOL_FAILED_ASYNC_TEMP_DATA_REQS +
POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
POOL_FAILED_ASYNC_TEMP_XDA_REQS
)
÷
(
  (
  POOL_FAILED_ASYNC_DATA_REQS +
  POOL_FAILED_ASYNC_INDEX_REQS +
  POOL_FAILED_ASYNC_XDA_REQS +
  POOL_FAILED_ASYNC_TEMP_DATA_REQS +
  POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
  POOL_FAILED_ASYNC_TEMP_XDA_REQS
  )
+
  (
  POOL_QUEUED_ASYNC_DATA_REQS +
  POOL_QUEUED_ASYNC_INDEX_REQS +
  POOL_QUEUED_ASYNC_XDA_REQS +
  POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
  POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
  POOL_QUEUED_ASYNC_TEMP_XDA_REQS
  )
) * 100
```

この数式は、生成された要求の総数に対する、失敗したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。このパーセンテージが低い場合は、時により、**num_ioservers** 構成パラメーターを変更して、より多くのプリフェッチャーを構成する必要があります。

pool_queued_async_xda_pages - XDA ページ・プリフェッチ要求のモニター・エレメント

正常にプリフェッチが要求された XML ストレージ・オブジェクト (XDA) のデータ・ページの数。

表 1339. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1339. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1340. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 1340. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントは、他の **pool_queued_async_..._pages** エレメントと一緒に、プリフェッチ要求によって取り出されたデータ・ページ数を示します。この情報を使用すると、システム上でプリフェッチ要求が効率的に実行されているかどうかを調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ要求あたりの平均ページ数を計算できます。

```
  (POOL_QUEUED_ASYNC_DATA_PAGES +
   POOL_QUEUED_ASYNC_INDEX_PAGES +
   POOL_QUEUED_ASYNC_XDA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_DATA_PAGES +
   POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES +
   POOL_QUEUED_ASYNC_TEMP_XDA_PAGES)
÷
  (POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS)
```

要求あたりの平均ページ数が低く、かつ、システム上で大量のプリフェッチが行われている場合は、必要以上の入出力操作がシステムで実行されている可能性があります。一般的に、要求サイズはプリフェッチ・サイズに基づいており、プリフェッチ・サイズは、少なくともエクステント・サイズと同じサイズでなければなりません。このため、平均要求サイズが小さいということは、プリフェッチ・サイズの設定が小さすぎること、また、プリフェッチ・サイズをエクステント・サイズの倍数に増やせばパフォーマンスを改善できることを示している可能性があります。また、平均要求サイズが小さいということは、プリフェッチ・キューがすぐに満杯になることを示している可能性もあります。そのため、関連する **pool_failed_async_..._reqs** モニター・エレメントもモニターすると役に立ちます。

pool_queued_async_xda_reqs - XDA プリフェッチ要求のモニター・エレメント

プリフェッチ・キューに正常に追加された、XML ストレージ・オブジェクト (XDA) のデータ・プリフェッチ要求の数。

表 1341. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1341. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1342. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

使用法

このエレメントは、他の **pool_queued_async_*_reqs** エレメントと一緒に、プリフェッチ・キューに追加されたプリフェッチ要求数を示します。この情報を使用すると、データベース・マネージャーでプリフェッチが行われる頻度が分かります。これらのエレメントを他のプリフェッチャー・モニター・エレメントと組み合わせて使用すると、システムで行われているプリフェッチの効率を調べることができます。例えば、以下に示すような数式を使用して、プリフェッチ・キューに正常に追加された要求の割合を求めることができます。

```
1 -
 (
 POOL_FAILED_ASYNC_DATA_REQS +
 POOL_FAILED_ASYNC_INDEX_REQS +
 POOL_FAILED_ASYNC_XDA_REQS +
 POOL_FAILED_ASYNC_TEMP_DATA_REQS +
 POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
 POOL_FAILED_ASYNC_TEMP_XDA_REQS
 )
 ÷
 (
   (
   POOL_FAILED_ASYNC_DATA_REQS +
   POOL_FAILED_ASYNC_INDEX_REQS +
   POOL_FAILED_ASYNC_XDA_REQS +
   POOL_FAILED_ASYNC_TEMP_DATA_REQS +
   POOL_FAILED_ASYNC_TEMP_INDEX_REQS +
   POOL_FAILED_ASYNC_TEMP_XDA_REQS
   )
 +
   (
   POOL_QUEUED_ASYNC_DATA_REQS +
   POOL_QUEUED_ASYNC_INDEX_REQS +
   POOL_QUEUED_ASYNC_XDA_REQS +
   POOL_QUEUED_ASYNC_TEMP_DATA_REQS +
   POOL_QUEUED_ASYNC_TEMP_INDEX_REQS +
   POOL_QUEUED_ASYNC_TEMP_XDA_REQS
   )
 ) * 100
```

この数式は、生成された要求の総数に対する、失敗したプリフェッチ要求の割合を計算するものです。失敗したプリフェッチ要求とは、プリフェッチ・キューに追加できなかった要求のことです。このパーセンテージが低い場合は、時により、**num_ioservers** 構成パラメーターを変更して、より多くのプリフェッチャーを構成する必要があります。

pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント

すべてのタイプの表スペースについて、表スペース・コンテナー (物理) からデータ・ページおよび索引ページを読み取るために費やされた合計時間を示します。この値はミリ秒単位で示されます。

表 1343. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1343. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1344. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1345. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	常に収集される

表 1345. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントと **pool_data_p_reads** および **pool_index_p_reads** モニター・エレメントを一緒に使用して、ページ読み取りの平均時間を計算します。この平均値は、入出力待ちがあるかどうかを示すので重要です。これにより、データをほかの装置に移動すべきかどうかがわかります。

データベースおよび表スペースのレベルでは、このエレメントには **pool_async_read_time** モニター・エレメントの値が含まれます。

pool_secondary_id メモリー・プール 2 次 ID

モニター・データを戻す対象となるメモリー・プールを判別するために役立つ追加 ID。

エレメント ID

pool_secondary_id

エレメント・タイプ

情報

表 1346. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本
アプリケーション	memory_pool	基本

表 1347. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	常に収集される
接続	event_connmemuse	常に収集される

使用法 モニター・データを戻す対象となるメモリー・プールを判別するために pool_id と共に使用します。 pool_secondary_id のデータは必要な場合のみ表示されます。例えば、示された pool_id が、モニター・データがどのバッファー・プールに関連するかを判別するためのバッファー・プール・ヒープである場合に pool_secondary_id のデータは表示されます。

データベースの作成時に、(IBMDEFAULTBP という) デフォルトのバッファー・プールがデータベースに作成され、そのサイズはプラットフォームによって決まります。このバッファー・プールは「1」という 2 次 ID を持ちます。このバッファー・プール、およびユーザーが作成するすべてのバッフ

ァー・プールに加えて、それぞれ異なるページ・サイズに対応するいくつかのシステム・バッファー・プールがデフォルトで作成されます。これらのバッファー・プールの ID は、pool_secondary_id のスナップショットに次のように表示される可能性があります。

- System 32k buffer pool
- System 16k buffer pool
- System 8k buffer pool
- System 4k buffer pool

pool_sync_data_gbp_reads - 同期グループ・バッファー・プール・データ読み取りモニター・エレメント

DB2 pureScale 環境において、データ・ページがバッファー・プールに存在すると予期されたが、そうではなくグループ・バッファー・プールから取得された回数。DB2 pureScale 環境以外の環境では、この値は 0 です。

pool_sync_data_reads - 同期バッファー・プール・データ読み取りモニター・エレメント

データ・ページがバッファー・プールに存在すると予期されたが、そうではなくディスクから読み取られた回数。

pool_sync_index_gbp_reads - 同期グループ・バッファー・プール索引読み取りモニター・エレメント

DB2 pureScale 環境において、索引ページがバッファー・プールに存在すると予期されたが、そうではなくグループ・バッファー・プールから取得された回数。 DB2 pureScale 環境以外の環境では、この値は 0 です。

pool_sync_index_reads - 同期バッファー・プール索引読み取りモニター・エレメント

索引ページがバッファー・プールに存在すると予期されたが、そうではなくディスクから読み取られた回数。

pool_sync_xda_gbp_reads - 同期グループ・バッファー・プール XDA データ読み取りモニター・エレメント

DB2 pureScale 環境において、 XML ページがバッファー・プールに存在すると予期されたが、そうではなくグループ・バッファー・プールから取得された回数。DB2 pureScale 環境以外の環境では、この値は 0 です。

pool_sync_xda_reads - 同期バッファー・プール XDA データ読み取りモニター・エレメント

XML ページがバッファー・プールに存在すると予期されたが、そうではなくディスクから読み取られた回数。

pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント

TEMPORARY 表スペースのバッファー・プール (論理) から要求された、データ・ページの数を示します。

表 1348. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 1348. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1349. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1350. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される

表 1350. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

pool_temp_data_p_reads エレメントと組み合わせて使用すると、TEMPORARY 表スペースにあるバッファー・プールのデータ・ページ・ヒット率を計算できます。

詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント

TEMPORARY 表スペースの表スペース・コンテナー (物理) から読み取られた、データ・ページの数を示します。

表 1351. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1351. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1352. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1353. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE

表 1353. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

ステートメント・レベルでバッファー・プール情報を記録する機能は、 API および CLP スナップショット要求用にサポートされています。

pool_temp_data_l_reads エレメントと組み合わせて使用すると、TEMPORARY 表スペースにあるバッファー・プールのデータ・ページ・ヒット率を計算できます。詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント

TEMPORARY 表スペースのバッファー・プール (論理) から要求された、索引ページの数を示します。

表 1354. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 1354. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1355. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

表 1355. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1356. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを **pool_temp_index_p_reads** エレメントと組み合わせて使用すると、TEMPORARY 表スペースにあるバッファー・プールの索引ページ・ヒット率を計算できます。詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

pool_temp_index_p_reads - バッファー・プールー時索引の物理読み取り : モニター・エレメント

TEMPORARY 表スペースの表スペース・コンテナー (物理) から読み取られた、索引ページの数を示します。

表 1357. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1357. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1358. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1359. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このエレメントを **pool_temp_index_l_reads** エレメントと組み合わせて使用すると、TEMPORARY 表スペースにあるバッファー・プールの索引ページ・ヒット率を計算できます。詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント

TEMPORARY 表スペースのバッファー・プール (論理) から要求された、XML ストレージ・オブジェクト (XDA) データのページの数を示します。

表 1360. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1360. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1361. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1362. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

pool_temp_xda_l_reads モニター・エレメントを **pool_temp_xda_p_reads**、**pool_temp_data_l_reads**、および **pool_temp_data_p_reads** モニター・エレメント

と組み合わせて使用すると、TEMPORARY 表スペースにあるバッファー・プールのデータ・ページ・ヒット率を次の数式で計算できます。

```
1 - ((pool_temp_data_p_reads + pool_temp_xda_p_reads)
      / (pool_temp_data_l_reads + pool_temp_xda_l_reads))
```

pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント

TEMPORARY 表スペースの表スペース・コンテナー (物理) から読み取られた、XML ストレージ・オブジェクト (XDA) データのページの数を示します。

表 1363. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1363. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1364. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1365. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

pool_temp_xda_p_reads モニター・エレメントを **pool_temp_xda_l_reads**、**pool_temp_data_l_reads**、および **pool_temp_data_p_reads** モニター・エレメントと組み合わせて使用すると、TEMPORARY 表スペースにあるバッファー・プールのデータ・ページ・ヒット率を次の数式で計算できます。

```
1 - ((pool_temp_data_p_reads + pool_temp_xda_p_reads) /
(pool_temp_data_l_reads + pool_temp_xda_l_reads))
```

pool_watermark メモリー・プール水準点

メモリー・プール作成後のその最大サイズ。値はバイト単位で示されます。

エレメント ID

pool_watermark

エレメント・タイプ

情報

表 1366. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本
アプリケーション	memory_pool	基本

表 1367. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	常に収集される
接続	event_connmemuse	常に収集される

使用法 継続的に稼働しているシステムの場合は、 *pool_watermark* と *pool_config_size* のエレメントを組み合わせて使用すると、メモリーに関する潜在的な問題を予測できます。

例えば、一定間隔 (例えば 1 日に 1 回) でスナップショットを取り、*pool_watermark* と *pool_config_size* の値を調べます。 *pool_watermark* の値が *pool_config_size* の値に近づく場合は (メモリー関連のトラブルが起こる可能性を示しています)、メモリー・プールのサイズを大きくする必要があることを示します。

pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント

データ・ページまたは索引ページをバッファー・プールからディスクに物理的に書き込むのに要した合計時間を示します。経過時間はミリ秒単位で示されます。

表 1368. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 1368. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1369. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1370. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される

表 1370. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを **pool_data_writes** および **pool_index_writes** モニター・エレメントと一緒に使用して、ページ書き込みの平均時間を計算します。この平均値は、入出力待ちがあるかどうかを示すので重要です。これにより、データをほかの装置に移動すべきかどうかがわかります。

データベースおよび表スペースのレベルでは、このエレメントには **pool_async_write_time** モニター・エレメントの値が含まれます。

pool_xda_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント

エージェントによってローカル・バッファー・プール (LBP) で検出された、グループ・バッファー・プール (GBP) に従属しない XML ストレージ・オブジェクト (XDA) のデータ・ページ数。

表 1371. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1371. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1372. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE

pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・ページが、ローカル・バッファー・プールで無効のマークが付けられているため、グループ・バッファー・プールから要求された回数。

表 1373. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1373. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1374. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

要求された XDA ページがローカル・バッファー・プールで検出された頻度を判別するには、次の数式を使用します。

```
(pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found) / pool_xda_l_reads
```

要求された XDA ページがグループ・バッファー・プールで検出された回数を判別するには、次の数式を使用します。

```
(pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) の GBP 従属のデータ・ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、グループ・バッファー・プールから読み取ろうとされた回数。

表 1375. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1375. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1376. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

要求された XDA ページがローカル・バッファー・プールで検出された頻度を判別するには、次の数式を使用します。

```
(pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found) / pool_xda_l_reads
```

要求された XDA ページがグループ・バッファー・プールで検出された回数を判別するには、次の数式を使用します。

```
(pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) の GBP 従属のデータ・ページが、グループ・バッファー・プールで検出されなかったためにディスクからローカル・バッファー・プールに読み込まれた回数。

表 1377. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1377. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1378. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

要求された XDA ページがローカル・バッファー・プールで検出された頻度を判別するには、次の数式を使用します。

```
(pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found) / pool_xda_l_reads
```

要求された XDA ページがグループ・バッファー・プールで検出された回数を判別するには、次の数式を使用します。

```
(pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースのバッファー・プール (論理) から要求された、XML ストレージ・オブジェクト (XDA) のデータ・ページの数を示します。

表 1379. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 1379. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1380. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1381. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	常に収集される
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このカウントには、次のデータへのアクセスが含まれます。

- データベース・マネージャーがページの処理を必要としたときにバッファー・プールにすでにあるデータ。
- データベース・マネージャーがページを処理する前にバッファー・プールに読み取られたデータ。

pool_xda_l_reads、**pool_xda_p_reads**、 **pool_data_l_reads**、および **pool_data_p_reads** モニター・エレメントを使用して、バッファー・プールのデータ・ページ・ヒット率を計算します。詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

例えば、全体のバッファー・プール・ヒット率は、次のように計算できます。

```
     ((pool_data_lbp_pages_found + pool_index_lbp_pages_found
 + pool_xda_lbp_pages_found
-
pool_async_data_lbp_pages_found - pool_async_index_lbp_pages_found
- pool_async_xda_lbp_pages_found)
/ (pool_data_l_reads
+ pool_index_l_reads + pool_xda_l_reads + pool_temp_data_l_reads +
pool_temp_xda_l_reads + pool_temp_index_l_reads)) × 100
```

この計算には、バッファー・プールによってキャッシュされているすべてのページ (索引とデータ) が含まれます。

バッファー・プール・サイズを大きくすると、一般的にヒット率は高くなりますが、ある点を超えると逆に低くなります。理想的には、データベース全体を保管で

きるような大きなバッファー・プールを割り振ることができれば、システムが稼働中のヒット率は 100% になります。しかし、現実的にはそうしたことは起こりません。使用するデータのサイズとそのデータへのアクセス方法によってヒット率の重要度は異なります。非常に大きなデータベースでアクセスが均等な場合は、ヒット率が低くなります。表が非常に大きな場合は、対応する方法はほとんどありません。このような場合、より小さく頻繁にアクセスがある表、および索引に着目するのが賢明です。

pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・ページが、ローカル・バッファー・プールから要求されて検出された回数。

表 1382. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 1382. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1383. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

要求された XDA ページがローカル・バッファー・プールで検出された頻度を判別するには、次の数式を使用します。

```
(pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found) / pool_xda_l_reads
```

要求された XDA ページがグループ・バッファー・プールで検出された回数を判別するには、次の数式を使用します。

```
(pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースの表スペース・コンテナー (物理) から読み取られた、XML ストレージ・オブジェクト (XDA) のデータ・ページの数を示します。

表 1384. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1384. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1385. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1386. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される

表 1386. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	バッファー・プール、ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

pool_async_xda_reads および **pool_xda_p_reads** モニター・エレメントを使用して、XML ストレージ・オブジェクト・データ・ページ上で同期的に実行された物理読み取りの数 (つまり、データベース・マネージャーのエージェントが XML データに対して実行したデータ・ページの物理読み取り数) を計算します。次の数式を使用します。

```
pool_xda_p_reads - pool_async_xda_reads
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。このエレメントは、**num_ioservers** 構成パラメーターを調整するときに役に立ちます。

pool_xda_l_reads、**pool_xda_p_reads**、 **pool_data_l_reads**、および **pool_data_p_reads** モニター・エレメントを使用して、バッファー・プールのデータ・ページ・ヒット率を計算します。詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のバッファー・プール・データ・ページがディスクに物理的に書き込まれた回数を示します。

表 1387. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 1387. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1388. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1389. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このモニター・エレメントは、データベースで使用可能なバッファー・プール・ページの数を増やすことによりパフォーマンスが向上するかを評価する助けとなります。 XML データを含むデータベースの場合、バッファー・プール・ページの読み取りに対する書き込みの比率を、XML データおよびリレーショナル・データの両方のタイプについて考慮する必要があります。XML データの場合は **pool_xda_writes** および **pool_xda_p_reads** モニター・エレメントを使用し、リレーショナル・データの場合は **pool_data_writes** および **pool_data_p_reads** モニター・エレメントを使用します。

pool_xda_l_reads、**pool_xda_p_reads**、 **pool_data_l_reads**、および **pool_data_p_reads** モニター・エレメントを使用して、バッファー・プールのデータ・ページ・ヒット率を計算します。詳しくは、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

port_number - ポート番号のモニター・エレメント

メンバーがクライアント接続を listen している TCP/IP ポート。

表 1390. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
DB_MEMBERS 表関数	常に収集される
MON_GET_SERVERLIST 表関数 - メンバーの優先順位の詳細を取得	常に収集される

post_shrthreshold_hash_joins ポストしきい値ハッシュ結合

ソート・メモリー・スロットル・アルゴリズムによってスロットルして戻されたハッシュ結合の合計数。スロットルされたハッシュ結合は、ソート・メモリー・マネージャーが要求するメモリーよりも少ないメモリーが付与されたハッシュ結合です。

表 1391. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1392. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

共有ソート・ヒープから割り振られるメモリーがデータベース構成パラメーター *sheapthres_shr* により設定された制限に近づくと、ハッシュ結合はスロットルして戻されます。システムが適切に構成されていない場合、このスロットルによって、*sheapthres_shr* 制限を超えたオーバーフロー数が大幅に削減されます。このエレメントで報告されるデータは、共有ソート・ヒープから割り振られるメモリーを使用したハッシュ結合のみを反映します。

post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント

ソート・メモリー・スロットル・アルゴリズムによってスロットルして戻されたソートの合計数。スロットルされたソートは、ソート・メモリー・マネージャーが要求するメモリーよりも少ないメモリーが付与されたソートです。

表 1393. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1393. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1394. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	Sort

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1395. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 1395. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

ソートに対するメモリーの割り振りがデータベース構成パラメーター **sheapthres_shr** により設定された制限に近づくと、ソートはスロットルして戻されます。システムが適切に構成されていない場合、このスロットルによって、**sheapthres_shr** 制限を超えたオーバーフロー数が大幅に削減されます。このエレメントで報告されるデータは、共有ソート・ヒープから割り振られるメモリーを使用したソートのみを反映します。

post_threshold_hash_joins ハッシュ結合のしきい値

共有または専用のソート・ヒープ・スペースが同時使用されていたためにハッシュ結合ヒープ要求が制限された合計回数。

エレメント ID

post_threshold_hash_joins

エレメント・タイプ

カウンター

表 1396. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 この値が大きい (hash_join_overflows の 5% より大きい) 場合は、ソート・ヒープのしきい値を大きくしてください。

post_threshold_olap_funcs OLAP 関数のしきい値 : モニター・エレメント

ソート・ヒープしきい値を超えた後にソート・ヒープを要求した OLAP 関数の数。

表 1397. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

ソート、ハッシュ結合、および OLAP 関数は、ソート・ヒープを使用する操作の例です。通常の条件では、データベース・マネージャーは、sortheap 構成パラメーターによって指定された値を使用して、ソート・ヒープを割り振ります。ソート・ヒープに割り振られたメモリー量がソート・ヒープのしきい値を超えると (sheapthres 構成パラメーター)、データベース・マネージャーは、sortheap 構成パラメーターが指定する値よりも低い値を使用して以降のソート・ヒープを割り振ります。

ソート・ヒープのしきい値に達すると、その後に開始した OLAP 関数では実行するための十分なメモリー量を得られないことがあります。

ソート、ハッシュ結合、OLAP 関数のパフォーマンス、およびシステム全体のパフォーマンスを改善するには、ソート・ヒープしきい値およびソート・ヒープ・サイズの構成パラメーターを変更します。

このエレメントの値が高い場合は、ソート・ヒープしきい値 (sheapthres) を上げます。

post_threshold_peas - partial early aggregation しきい値のモニター・エレメント

ソート・ヒープしきい値を超過するため、要求した量より少ないメモリーを partial early aggregation 操作が受け取った回数。

表 1398. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 1398. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	REQUEST METRICS BASE

表 1399. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 1399. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1400. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
接続	event_conn	-
ステートメント	event_stmt	-
トランザクション	event_xact	-
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントと **total_peas** モニター・エレメントを組み合わせて使用すると、partial early aggregation 操作に十分なソート・ヒープ・メモリーが付与されているかを大まかに調べることができます。**total_peas** モニター・エレメントに対する **post_thresold_peas** モニター・エレメントの比率が高い場合、データベースのパフォーマンスは最適ではない可能性があります。ソート・ヒープ・サイズかソート・ヒープしきい値のいずれか一方または両方を大きくすることを検討してください。

post_threshold_peds - partial early distinct しきい値のモニター・エレメント

ソート・ヒープしきい値を超過するため、要求した量より少ないメモリーを partial early distinct 操作が受け取った回数。

表 1401. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 1402. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1402. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1403. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
接続	event_conn	-
ステートメント	event_stmt	-
トランザクション	event_xact	-
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントと **total_peds** モニター・エレメントを組み合わせて使用すると、partial early distinct 操作に十分なソート・ヒープ・メモリーが付与されているかを大まかに調べることができます。**total_peds** モニター・エレメントに対する **post_thresold_peds** モニター・エレメントの比率が高い場合、データベースのパフォーマンスは最適ではない可能性があります。ソート・ヒープ・サイズかソート・ヒープしきい値のいずれか一方または両方を大きくすることを検討してください。

post_threshold_sorts - ポストしきい値ソート : モニター・エレメント

ソート・ヒープしきい値に達した後でヒープを要求したソートの数。

表 1404. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1405. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	Sort

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1406. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

通常の条件では、データベース・マネージャーは、 **sortheap** 構成パラメーターによって指定された値を使用して、ソート・ヒープを割り振ります。ソート・ヒープに割り振られたメモリー量がソート・ヒープのしきい値を超えると (**sheapthres** 構成パラメーター)、データベース・マネージャーは、 **sortheap** 構成パラメーターが指定する値よりも低い値を使用してソート・ヒープを割り振ります。

システム上のアクティブ・ソートには、それぞれメモリーが割り振られるので、利用可能なシステム・メモリーからソートのためのメモリー量を多く取り過ぎることがあります。ソート・ヒープのしきい値を超えると、その後に開始したソートでは実行するための十分なメモリー量を得られないことがありますが、システム全体としてはそのほうが利点があります。ソート・ヒープしきい値およびソート・ヒープ・サイズの構成パラメーターを変更すると、ソート操作のパフォーマンスとシステム全体のパフォーマンスを改善できます。エレメントの値が高い場合は、次のいずれかを行うことができます。

- ソート・ヒープしきい値 (**sheapthres**) を上げる。
- SQL 照会で使用するソート数を少なくするか、小さくなるようにアプリケーションを調整する。

prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント

アプリケーションが入出力サーバー (プリフェッチャー) によるバッファー・プールへのページのロードの終了を待機していた時間。値はミリ秒単位で示されます。

表 1407. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1407. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1408. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
アプリケーション	appl	バッファー・プール

表 1409. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	Bufferpool
接続	event_db	Bufferpool

使用法 このエレメントを使用すると、入出力サーバーの数と入出力サーバーのサイズの変更を試すことができます。

prefetch_waits - プリフェッチャーの待機カウントのモニター・エレメント

入出力サーバー (プリフェッチャー) がバッファー・プールにページをロードし終えるのを待機した回数。

表 1410. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1410. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1411. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます) event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

prep_time 準備時間 : モニター・エレメント

SQL ステートメントを準備するために要した時間 (ミリ秒単位) (アクティビティーが SQL ステートメントである場合。それ以外の場合の値は 0)。

表 1412. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1413. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

prep_time モニター・エレメントは、このアクティビティーが SQL ステートメントだった場合に、SQL ステートメントが DB2 パッケージ・キャッシュに最初に取り込まれたときのステートメントの準備に費やされた時間を示します。この準備時間

は、アクティビティー存続時間の一部ではありません。また、ステートメントが呼び出しの前に既にパッケージ・キャッシュに入れられていた場合のステートメントの特定の呼び出しに費やされた時間を表しているわけでもありません。

prep_time_best ステートメント最短準備時間 : モニター・エレメント

特定の SQL ステートメントの準備に要した最短時間 (ミリ秒単位)。

表 1414. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

使用法

この値を **prep_time_worst** とともに使用して、コンパイルに長い時間がかかる SQL ステートメントを識別します。

prep_time_worst ステートメント最長準備時間 : モニター・エレメント

特定の SQL ステートメントの準備に要した最長時間 (ミリ秒単位)。

表 1415. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

使用法

この値を **prep_time_best** とともに使用して、コンパイルに長い時間がかかる SQL ステートメントを識別します。

prev_uow_stop_time 直前の作業単位完了タイム・スタンプ

作業単位が完了した時刻です。

エレメント ID

prev_uow_stop_time

エレメント・タイプ

timestamp

表 1416. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

表 1417. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	常に収集される

使用法 このエレメントと *uow_stop_time* を組み合わせて使用すると、 COMMIT と ROLLBACK のポイント間の合計経過時間を計算できます。 *uow_start_time* と組み合わせて使用すると、作業単位間のアプリケーションの合計時間を計算できます。次のアクションのいずれかの時刻を示します。

- アプリケーションが作業単位中の場合は、最後に作業単位が完了した時刻を示す。
- アプリケーションが作業単位中ではない場合は (アプリケーションが 1 つの作業単位を完了し、次の作業単位をまだ開始していない場合)、最後に完了した作業単位の直前の作業単位の停止時刻を示す。最後に完了した作業単位の停止時刻は、uow_stop_time で示す。
- アプリケーションが最初の作業単位中の場合は、データベース接続要求完了時刻となる。

priority - 優先順位の値のモニター・エレメント

メンバーの処理能力を相対的に示します。値が大きいほど、そのメンバーにクライアントから割り振られる処理量が多くなります。

表 1418. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVERLIST 表関数 - メンバーの優先順位の詳細を取得	常に収集される

使用上の注意

- このモニター・エレメントはメンバーの相対負荷 (重みとも呼ばれる) を表しています。例えば、メンバー A の優先順位の値が 80 で、メンバー B の優先順位の値が 40 の場合、メンバー A はメンバー B に与えられる作業量の 2 倍の作業量を受け取ることを意味します。
- この値はパーセンテージを表すものではありません。
- このモニター・エレメントの最大値は 100 です。

priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数

割り振られたメモリーの境界から専用ワークスペースがオーバーフローした回数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1419. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1420. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントと priv_workspace_size_top を組み合わせて使用すると、オーバーフローを防止するのに専用ワークスペースのサイズを大きくする必要があるかどうかを判別できます。専用ワークスペースがオーバーフローすると、パフォーマンスが低下するだけではなく、エージェントの専用メモリーから割り振られたほかのヒープでメモリー不足エラーが発生することがあります。

データベース・レベルでは、「専用ワークスペースの最大サイズ」のある専用ワークスペースとして報告された専用ワークスペースがこのエレメントの報告の対象となります。アプリケーション・レベルでは、現行アプリケーションにサービスを提供した各エージェントのワークスペースがオーバーフローした回数となります。

priv_workspace_section_inserts 専用ワークスペース・セクション挿入

専用ワークスペースへの、アプリケーションによる SQL セクション挿入数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1421. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1422. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 実行可能セクションの作業用コピーは、専用ワークスペース内に保管されます。

このカウンターは、コピーが使用できなかったために挿入が必要だった場合を示します。データベース・レベルでは、データベース内のすべての専用ワークスペースを対象に、すべてアプリケーションでの累計挿入数を示します。アプリケーション・レベルでは、このアプリケーションの専用ワークスペース内にあるすべてのセクションを対象とした累計挿入数を示します。

エージェントが異なるアプリケーションに関連付けられているようなコンセントレーター環境では、新しいエージェントに必要な使用可能なセクションが専用ワークスペース内にない場合に、専用ワークスペースの追加挿入が必要になります。

priv_workspace_section_lookups 専用ワークスペース・セクション検索

エージェントの専用ワークスペースでの、アプリケーションによる SQL セクション検索数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1423. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1424. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 各アプリケーションは、自分に代わって作業するエージェントの専用ワークスペースにアクセスできます。

このカウンターは、アプリケーション用の特定セクションを見つけるために専用ワークスペースがアクセスされた回数を示します。データベース・レベルでは、データベース内のすべての専用ワークスペースを対象に、すべてのアプリケーションでの累計検索数を示します。アプリケーション・レベルでは、このアプリケーションの専用ワークスペース内にあるすべてのセクションを対象とした累計検索数を示します。

このエレメントと「専用ワークスペース・セクション挿入」を組み合わせて使用すると、専用ワークスペースのサイズを調整できます。専用ワークスペースのサイズをコントロールしているのは、applheapsz 構成パラメーターです。

priv_workspace_size_top 専用ワークスペースの最大サイズ

専用ワークスペースが到達した最大サイズ。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1425. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

表 1426. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 各エージェントには 1 つの専用ワークスペースがあり、エージェントがサービスを提供するアプリケーションはこれにアクセスをします。このエレメントは、アプリケーションにサービスを提供するエージェントが必要とする専用ワークスペースの最大バイト数を示します。データベース・レベルでは、現行データベースにアタッチされているすべてのエージェントが必要とする、すべての専用ワークスペースの最大バイト数を示します。アプリケーション・レベルでは、現行アプリケーションにサービスを提供したエージェントのすべての専用ワークスペースの中での最大サイズを示します。

専用ワークスペースがオーバーフローすると、エージェント専用メモリーにあるほかのエンティティーからメモリーを一時的に借用します。この結果、これらのエンティティーでメモリー不足エラーが発生したり、パフォーマンスが低下することがあります。 APPLHEAPSZ を大きくすると、オーバーフローの確率を低くすることができます。

product_name - 製品名

実行中の DB2 インスタンスのバージョンの詳細情報。

表 1427. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

progress_completed_units 完了した進行作業単位

現行フェーズの、完了した作業単位の数。

表 1428. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

通常このエレメントの値は、ユーティリティーが作動するにつれて大きくなります。このエレメントは、常に *progress_total_units* 以下になります (両方のエレメントとも定義されている場合)。

注:

1. このエレメントが組み込まれていないユーティリティーが存在する可能性があります。
2. このエレメントは、 *progress_work_metric* モニター・エレメントで表示される単位で表されます。

使用法 このエレメントを使用して、フェーズ中の完了した作業の量を判別できます。このエレメントを単独で使用すると、実行中のユーティリティーのアクティビティーをモニターできます。このエレメントは、ユーティリティーの実行につれて連続的に大きくなるはずです。 *progress_completed_units* が長期間大きくならない場合は、ユーティリティーが停止している可能性があります。

progress_total_units を定義している場合は、このエレメントを使用して、完了した作業のパーセンテージを計算できます。

```
完了した作業のパーセンテージ = progress_completed_units /
progress_total_units* 100
```

progress_description 進行の記述

作業のフェーズの説明。

表 1429. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

ロード・ユーティリティーの値の例には、以下が含まれます。

- DELETE
- LOAD
- REDO

使用法 このエレメントを使用して、フェーズの一般説明を取得します。

progress_list_attr 現在の進行リストの属性

このエレメントは、進行エレメントのリストを解釈する方法を記述します。

表 1430. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress_list	基本

使用法

このエレメントの値は、以下のいずれかの定数です。

- SQLM_ELM_PROGRESS_LIST_ATTR_SERIAL - リスト内のエレメントは順次フェーズのセットとして解釈されます。つまり、完了した処理数は、エレメント *n*+1 の完了した処理が最初に更新される前にエレメント *n* の合計処理数と等しくならなければなりません。この属性は、次のフェーズが開始する前に 1 つのフェーズが完全に完了する必要がある順次フェーズのセットで構成されるタスクの進行を記述するために使用されます。

- SQLM_ELM_PROGRESS_LIST_ATTR_CONCURRENT - 進行リスト内の任意のエレメントは、いつでも更新できます。

このエレメントを使用すると、進行リストのエレメントを更新する方法を判別できます。

progress_list_cur_seq_num 現行の進行リストのシーケンス番号

ユーティリティーに複数の連続したフェーズが含まれている場合、このエレメントは現行フェーズの番号を表示します。

表 1431. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress_list	基本

使用法 このエレメントを使用して、ユーティリティーの連続したフェーズのうちの現在のフェーズを判別できます。 『progress_seq_num 進行シーケンス番号』を参照してください。

progress_seq_num 進行シーケンス番号

フェーズ番号。

注: 複数の実行フェーズから成るユーティリティーの場合のみ、フェーズ番号が表示されます。

表 1432. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

使用法 このエレメントを使用して、複数フェーズ・ユーティリティー中のフェーズの順序を判別できます。このユーティリティーは、進行シーケンス番号の昇順でフェーズを実行します。複数フェーズ・ユーティリティーの現行フェーズを見つけるには、 *progress_seq_num* と、 *progress_list_current_seq_num* の値を突き合わせます。

progress_start_time 進行開始時刻

フェーズの開始を示すタイム・スタンプ。

表 1433. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

使用法 このエレメントを使用すると、フェーズが開始した時点を判別できます。フェーズがまだ開始されていない場合は、このエレメントは省略されます。

progress_total_units 合計進行作業単位

フェーズを完了するために実行する作業の合計量。

表 1434. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

合計作業量を定量化できないので、このエレメントが継続的に更新されるユーティリティーもあれば、合計作業を見積もれないので、このエレメントが完全に省略されるユーティリティーもあります。

このエレメントは、 *progress_work_metric* モニター・エレメントで表示される単位で表されます。

使用法 このエレメントを使用して、フェーズ中の作業の合計量を判別します。フェーズ中の完了した作業のパーセンテージを計算するには、このエレメントと *progress_completed_units* を併用してください。

```
完了した作業のパーセンテージ = progress_completed_units /
progress_total_units* 100
```

progress_work_metric 進行作業メトリック

progress_total_units エレメントと *progress_completed_units* エレメントを解釈するメトリック。

表 1435. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

値の例には、以下のものがあります。

- SQLM_WORK_METRIC_BYTES
- SQLM_WORK_METRIC_EXTENTS

注:

1. このエレメントが組み込まれていないユーティリティーが存在する可能性があります。
2. このエレメントの値は sqlmon.h 中にあります。

使用法 このエレメントを使用して、 *progress_total_units* と *progress_completed_units* が報告メトリックとして使用しているものを判別します。

pseudo_deletes - 疑似削除 : モニター・エレメント

疑似削除としてマークされたキーの数。

表 1436. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

pseudo_empty_pages - 疑似空ページ : モニター・エレメント

疑似空として識別されたページの数。疑似空ページとは、すべてのキーが疑似削除されたページのことです。

表 1437. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

使用法

注: このモニター・エレメントは、疑似空ページの現在の数をレポートしません。

query_actual_degree - パーティション内並列処理の実際の実行時の多重度のモニター・エレメント

ステートメント、アクティビティー、トランザクション、またはワークロードの各レベルでレポートされるパーティション内並列処理の実際の実行時の多重度。

表 1438. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1439. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	-	常に収集される
ロック		

query_card_estimate 照会行数の見積もり

照会によって戻される行数の見積もり。

表 1440. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 1440. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS ステートメント	dcs_stmt	ステートメント
アクティビティー	event_activity	-

使用法 SQL コンパイラーによるこの見積もりは、ランタイムの実際のものと比較できます。

このエレメントは、DB2 Connect をモニター中は次の SQL ステートメントに関する情報も戻します。

- INSERT、UPDATE、および DELETE

  影響を受ける行数を示します。

- PREPARE

  戻される行数の見積もり。 DRDA サーバーが DB2 for Linux, UNIX, and Windows、DB2 for VM/VSE、または DB2 for OS/400® の場合にのみ収集します。

- FETCH

  フェッチされた行数に設定されます。 DRDA サーバーが DB2 for OS/400 の場合にのみ収集します。

DRDA サーバーで情報が収集されないと、エレメントはゼロに設定されます。

query_cost_estimate - 照会コストの見積もり : モニター・エレメント

SQL コンパイラーによって判別された照会の見積もりコスト。この値は timerons 単位で報告されます。

表 1441. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 1442. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント
アクティビティー	event_activity	-
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このモニター・エレメントにより、実際の実行時とコンパイル時の見積もりの相関をとることができます。

このエレメントは、DB2 Connect をモニター中は次の SQL ステートメントに関する情報も戻します。

- PREPARE

  準備済み SQL ステートメントの相対コストを示します。

- FETCH

  取り出した行の長さが含まれています。 DRDA サーバーが DB2 for OS/400 の場合にのみ収集します。

DRDA サーバーで情報が収集されないと、エレメントはゼロに設定されます。

注: DRDA サーバーが DB2 for OS/390® and z/OS の場合は、この見積もりが 2**32 - 1 (符号なしロング変数を使用して表現できる最大整数) よりも大きい値になることがあります。この場合は、モニターがこのエレメントに戻す値は 2**32 - 1 になります。

query_data_tag_list - 見積もりの照会データ・タグ・リストのモニター・エレメント

query_data_tag_list モニター・エレメントは、ステートメントで参照されるコンパイラーによって見積もられたデータ・タグ値のコンマ区切りリストです。

表スペースがゼロ以外のデータ・タグ属性を定義したデータを含む表に、ステートメントがアクセスするとコンパイラーが予測した場合に、そのデータ・タグ値がリストに入れられます。リストに重複する値は入っていません。

表 1443. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1443. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1444. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	ACTIVITY METRICS BASE
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用上の注意

照会がアクセスしたデータ表スペースに定義されているデータ・タグがない場合には、リストは空になります。

queue_assignments_total キュー割り当ての合計 : モニター・エレメント

最後にリセットされてからこのしきい値キューに接続またはアクティビティーが割り当てられた回数。

表 1445. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 1446. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、統計収集間隔により決定される特定の期間にアクティビティーまたは接続がこの特定のキューに入れられた回数を判別できます。 これは、キューのしきい値の効果性を判別する助けになります。

queue_start_time - キュー開始タイム・スタンプのモニター・エレメント

しきい値チケットを取得するためにキューでの待機をアプリケーションが開始した日時。

表 1447. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

queue_size_top キュー・サイズの最上位 : モニター・エレメント

最後にリセットしてから到達したキュー・サイズの最大値。

表 1448. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 1449. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、キューのしきい値の効果性を測定したり、キューイングが大きすぎる時を検出したりすることができます。

queue_time_total キュー時間の合計 : モニター・エレメント

最後にリセットされてからキューに置かれたすべての接続またはアクティビティーについて、このキューで費やされた合計時間。単位はミリ秒です。

表 1450. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 1451. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	常に収集される

このエレメントは、キューイングのしきい値の有効性を評価する場合や、過度のキューイングを検出する場合に使用します。

使用上の注意

queue_time_total は、統計収集間隔の終わりにリセットされません。**queue_time_total** を複数の間隔で使用した場合は、**wlm_collect_int** と **queue_size_top** の積よりも大きくなることがあります。

queued_agents - キューに入れられているしきい値エージェントのモニター・エレメント

しきい値で現在キューに入れられているエージェント数の合計。

表 1452. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participants	

quiescer_agent_id 静止プログラム・エージェント ID

静止状態を保持するエージェントのエージェント ID。

エレメント ID
quiescer_agent_id

エレメント・タイプ
情報

表 1453. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントと quiescer_auth_id を組み合わせて使用すると、表スペースを静止させたユーザーを判別できます。

quiescer_auth_id 静止プログラム・ユーザー許可 ID

静止状態を保持するユーザーの許可 ID。

エレメント ID
quiescer_auth_id

エレメント・タイプ
情報

表 1454. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントを使用すると、表スペースを静止させたユーザーを判別できます。

quiescer_obj_id 静止プログラム・オブジェクト ID

表スペースを静止させたオブジェクトのオブジェクト ID。

エレメント ID
quiescer_obj_id

エレメント・タイプ
情報

表 1455. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントと quiescer_ts_id および quiescer_auth_id を組み合わせて使用すると、表スペースを静止させたオブジェクトを判別できます。このエレメントの値は、SYSCAT.TABLES ビューの TABLEID 列の値と一致します。

quiescer_state 静止プログラムの状態

行われている静止タイプ (例えば、「共有」、「更新する」、または「排他」)。

エレメント ID

quiescer_state

エレメント・タイプ

情報

表 1456. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントの値は、sqlutil.h の SQLB_QUIESCED_SHARE、SQLB_QUIESCED_UPDATE、または SQLB_QUIESCED_EXCLUSIVE の定数の値と一致します。

quiescer_ts_id 静止プログラム表スペース ID

表スペースを静止させたオブジェクトの表スペース ID。

エレメント ID

quiescer_ts_id

エレメント・タイプ

情報

表 1457. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントと quiescer_obj_id および quiescer_auth_id を組み合わせて使用すると、表スペースを静止させたユーザーを判別できます。このエレメントの値は、SYSCAT.TABLES ビューの TBSPACEID 列の値と一致します。

range_adjustment 範囲調整

この値は、1 つの範囲が実際に開始するコンテナー配列のオフセットを示します。

表 1458. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_container_id 範囲コンテナー

範囲内でコンテナーを一意的に定義する整数。

表 1459. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_end_stripe 終了ストライプ

この値は、1 つの範囲内の最後のストライプの番号を示します。

表 1460. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_max_extent 範囲内の最大エクステント

この値は、1 つの範囲がマップする最大エクステント番号を示します。

表 1461. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_max_page_number 範囲内の最大ページ

この値は、1 つの範囲がマップする最大ページ番号を示します。

表 1462. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_num_containers 範囲内コンテナーの数

この値は、現行範囲のコンテナー数を示します。

表 1463. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_number 範囲番号

この値は、表スペース・マップ内にある 1 つの範囲の番号を示します。

表 1464. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_offset 範囲オフセット

1 つの範囲が属するストライプ・セット開始点のストライプ 0 からのオフセット。

表 1465. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_start_stripe 開始ストライプ

この値は、1 つの範囲内の最初のストライプの番号を示します。

表 1466. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_stripe_set_number ストライプ・セット番号

この値は、中に 1 つ範囲が含まれているストライプ・セットを示します。

表 1467. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

reclaim_wait_time - 再利用待機時間 : モニター・エレメント

DB2 pureScale 環境では、このエレメントは、ページ・ロックの待機にかかった時間を示します。ロック要求によって、ページが再利用されます。この時間の測定単位はミリ秒です。

表 1468. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 1468. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1469. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される

使用法

スペース・マップ・ページ上で再利用を待機するための時間は別個にカウントされ、**spacemappage_reclaim_wait_time** モニター・エレメントでレポートされます。

reclaimable_space_enabled - 再利用可能なスペースが有効な標識 : モニター・エレメント

再利用可能ストレージで表スペースが使用可能な場合、このモニター・エレメントは値 1 を戻します。それ以外の場合は、値 0 を戻します。

表 1470. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

regvar_collection_type - レジストリー変数収集タイプ

レジストリー変数の値がいつ収集されたかを示します。

表 1471. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	REGVAR	常に収集される

使用法

変更履歴イベント・モニターで収集されるこの値は、以下のとおりです。

I イベント・モニターが活動化されたときにキャプチャーされた初期値。

U 更新された値

regvar_level - レジストリー変数のレベル

レジストリー変数のレベルを示します。

表 1472. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	REGVAR	常に収集される

使用法

変更履歴イベント・モニターの場合、レジストリー変数のレベルは、以下のいずれかです。

E 環境

G グローバル

I インスタンス・レベル

P データベース・パーティション

regvar_name - レジストリー変数名

レジストリー変数の名前。

表 1473. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	REGVAR	常に収集される

使用法

変更履歴イベント・モニターでは、このエレメントは、REGVAR イベントの一環として更新されたレジストリー変数、または REGVARVALUES イベントの一環としてイベント・モニターの開始時にキャプチャーされたレジストリー変数を示します。これらのイベントは、以下を示します。

REGVAR
レジストリー変数値の変更

REGVARVALUES
イベント・モニター開始時のレジストリー変数値のキャプチャー

regvar_old_value - レジストリー変数の古い値

レジストリー変数の古い値。

表 1474. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	REGVAR	常に収集される

使用法

変更履歴イベント・モニターでは、レジストリー変数値が設定されていなかった場合、この値は空ストリングです。

regvar_value - レジストリー変数の値

これはレジストリー変数の値です。

表 1475. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	REGVAR	常に収集される

使用法

変更履歴イベント・モニターでは、値が設定されていなかった場合、この値は空ストリングです。

REGVAR イベントは、レジストリー変数の即時更新の場合にのみ生成されます。

rej_curs_blk リジェクトされたブロック・カーソル要求

サーバーでの入出力ブロック要求がリジェクトされて、要求が非ブロック化入出力に変換された回数。

エレメント ID
rej_curs_blk

エレメント・タイプ
カウンター

表 1476. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 1477. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される

使用法 多数のカーソル・ブロッキング・データがあると、通信ヒープが満杯になることがあります。このヒープが満杯になると、エラーは戻されません。その代わりに、カーソルをブロックするための入出力ブロックが割り振られなくなります。カーソルがデータをブロックできない場合は、パフォーマンスに影響が現れます。

多数のカーソルがデータ・ブロックを実行できない場合は、次のようにしてパフォーマンスを改善できます。

- *query_heap* データベース・マネージャー構成パラメーターのサイズを大きくする。

rem_cons_in - データベース・マネージャーへのリモート接続

モニター中のデータベース・マネージャーのインスタンスに対してリモート・クライアントから開始された接続の現在の数。

表 1478. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

リモート・クライアントからこのインスタンス内のデータベースへの接続数を示します。この値は頻繁に変化するため、システム使用量の現実的な値を得るためには、長期にわたり特定のインターバルを設けてサンプルを採取する必要があります。この数値には、データベース・マネージャーと同じインスタンスで開始したアプリケーションは含まれません。

これらのエレメントと local_cons モニター・エレメントを組み合わせて使用すると、**max_coordagents** および **max_connections** 構成パラメーターの設定値を調整するときに利用できます。

rem_cons_in_exec - データベース・マネージャーで実行中のリモート接続

モニター中のデータベース・マネージャー・インスタンス内のデータベースに現在接続していて、作業単位を現在処理しているリモート・アプリケーションの数。

表 1479. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

この数値は、データベース・マネージャーで実行中の並行処理のレベルを判別するときに利用できます。この値は頻繁に変化するため、システム使用量の現実的な値を得るためには、長期にわたり特定のインターバルを設けてサンプルを採取する必要があります。この数値には、データベース・マネージャーと同じインスタンスで開始したアプリケーションは含まれません。

このエレメントと local_cons_in_exec モニター・エレメントを組み合わせて使用すると、**max_coordagents** 構成パラメーターの設定値を調整するときに利用できます。

max_coordagents を AUTOMATIC に設定する場合は、調整を加える必要はありません。 **max_coordagents** を AUTOMATIC に設定せず、rem_cons_in_exec および local_cons_in_exec の合計が **max_coordagents** に近い場合は、**max_coordagents** の値を増やしてください。

remote_lock_time リモート・ロック時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからのリモート・ロックに、このデータ・ソースが要した合計時間が含まれています (ミリ秒単位)。

このモニターは最新の値を格納します。応答時間は、フェデレーテッド・サーバーがデータ・ソースにリモート・ロックをサブミットしてからフェデレーテッド・サーバーがデータ・ソースでリモート・ロックを解放するまでの時間です。

表 1480. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このデータ・ソースでリモート・ロックに要した実際の時間を判別できます。

remote_locks リモート・ロック数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースで呼び出したリモート・ロックの合計数が含まれています。

表 1481. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データ・ソースでリモート側で行われたリモート・ロックの数を判別できます。

remote_member - リモート・メンバー : モニター・エレメント

高速コミュニケーション・マネージャー (FCM) を使用してデータの送信または受信が行われたデータベース・メンバーの数値 ID。

表 1482. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE

使用法

MON_GET_FCM_CONNECTION_LIST 表関数が返すメトリックはすべて、**member** モニター・エレメントと **remote_member** モニター・エレメントで示されたメンバー間の FCM 接続に適用されます。

reopt - REOPT バインド・オプションのモニター・エレメント

このパッケージをプリコンパイルするために使用される REOPT バインド・オプション。可能な値は NONE、ONCE、および ALWAYS です。

表 1483. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participant_activities	

reorg_completion 再編成完了フラグ

表再編成の成功インディケーターです。マルチディメンション・クラスタリング (MDC) 表または挿入時クラスタリング (ITC) 表のエクステントを再利用するための再編成も含まれます。パーティション表の場合、この値は、データ・パーティションでの完了状況を示します。

表 1484. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 表再編成処理またはデータ・パーティション再編成処理が正常に終了すると、このエレメントの値は 0 になります。表再編成処理またはデータ・パ

ーティション再編成処理が失敗すると、このエレメントの値は -1 になります。成功および失敗の値は、sqlmon.h で次のように定義されています。

- 成功: SQLM_REORG_SUCCESS
- 失敗: SQLM_REORG_FAIL

表再編成が異常終了した場合は、履歴ファイルを使用して、警告やエラーなどの診断情報を参照してください。このデータにアクセスするには、LIST HISTORY コマンドを使用します。パーティション表では、完了状況はデータ・パーティションごとに示されます。パーティション表での索引の再作成が失敗した場合、失敗した状況がすべてのデータ・パーティションで更新されます。詳細な診断情報については、管理通知ログを参照してください。

reorg_current_counter 再編成の進行状況

完了した再編成の量を示す進行状況の単位。この値が示す進行の量は、行われる表再編成の合計量を示す reorg_max_counter の値に関連しています。

表 1485. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法

次の公式を使用して、完了した表再編成のパーセンテージを判別することができます。

```
表再編成の進行状況 = reorg_current_counter / reorg_max_counter * 100
```

reorg_end 表再編成終了時刻

表の再編成の終了時刻です。マルチディメンション・クラスタリング (MDC) 表または挿入時クラスタリング (ITC) 表のエクステントを再利用するための再編成も含まれます。パーティション表の場合は、この時刻は、各データ・パーティションの再編成の終了時刻を示します。

表 1486. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_index_id 表の再編成に使用される索引

表の再編成に使用されている索引。

表 1487. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_long_tbspc_id - 長いオブジェクトが再編成される表スペース : モニター・エレメント

任意の長いオブジェクト (LONG VARCHAR または LOB データ) が再編成される表スペース。パーティション化されている表の場合、それぞれのパーティションの LONG VARCHAR と LOB が再編成される表スペースです。

表 1488. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_max_counter 再編成の合計量

再編成において行われる作業の合計量を示す値です。この値には、マルチディメンション・クラスタリング (MDC) 表または挿入時クラスタリング (ITC) 表のエクステントを再利用するための再編成も含まれます。

この値を、完了した作業の量を示す reorg_current_counter とともに使用して、再編成の進行状況を判別することができます。

表 1489. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_max_phase 再編成の最大フェーズ数

再編成処理のときに発生する再編成フェーズの最大数。この数は、従来の (オフライン) 再編成、および `RECLAIM EXTENTS` オプションを指定した再編成の場合に適用されます。

値の範囲は 2 から 4 です ([SORT], BUILD, REPLACE,[INDEX_RECREATE])。この値は、マルチディメンション・クラスタリング (MDC) 表または挿入時クラスタリング (ITC) 表のエクステントを再利用するための再編成で行われる作業の合計量を示す場合もあります。そのような再編成が行われる場合、この値は 3 (SCAN、DRAIN、RELEASE) となります。

表 1490. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_phase - 表の再編成のフェーズ : モニター・エレメント

表の再編成フェーズを示します。パーティション表の場合、それぞれのデータ・パーティションの再編成フェーズも示します。これはオフライン表再編成にのみ適用されます。

表 1491. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法

パーティション表の場合、再編成はデータ・パーティション単位でデータ・パーティション上で行われます。クラシック表再編成の場合は、次のフェーズがあります (sqlmon.h ファイルでの対応する定義と共にフェーズをリストしています)。

- ソート: SQLM_REORG_SORT
- ビルド: SQLM_REORG_BUILD
- 置換: SQLM_REORG_REPLACE
- 索引の再作成: SQLM_REORG_INDEX_RECREATE
- ディクショナリーのビルド: SQLM_REORG_DICT_SAMPLE

パーティション表の場合、そのデータ・パーティションの置換フェーズ後に直接、パーティション索引の索引の再作成フェーズに入ることがあります (パーティション索引がある場合)。すべてのデータ・パーティションで前のフェーズすべてが正常に完了してからでなければ、**reorg_phase** エレメントは索引の再作成フェーズを示しません。

XDA オブジェクト圧縮時の XML データ再編成フェーズで、表の XML ストレージ・オブジェクトの再編成が行われます。 XML ディクショナリーのビルド・フェーズで、XML ストレージ・オブジェクト用コンプレッション・ディクショナリーの作成が試みられます。 XDA オブジェクト圧縮の場合は、次の 2 つのフェーズがあります。

- XML 再編成: SQLM_REORG_XML_DATA
- XML ディクショナリーのビルド: SQLM_REORG_XML_DICT_SAMPLE

エクステントの再利用が行われているパーティション表の場合は、次のフェーズがあります。

- スキャン: SQLM_REORG_SCAN
- ドレーン: SQLM_REORG_DRAIN
- リリース: SQLM_REORG_RELEASE

reorg_phase_start 表再編成フェーズ開始時刻

表再編成または再利用のための再編成のフェーズの開始時刻。 パーティション表の場合、それぞれのデータ・パーティションの再編成フェーズの開始時刻も示します。索引再作成フェーズの時、すべてのデータ・パーティションでは非パーティション索引のデータ・グループは同時に更新されます。

表 1492. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_rows_compressed - 圧縮行数

再編成中に表で圧縮される行の数。

表 1493. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 再編成中に表で圧縮される行の数の連続したカウント。圧縮されないレコードもあります (レコード・サイズが最小レコード長より小さい場合)。

この行数はデータ圧縮の有効性を測るものではないということに注意してください。これは圧縮基準を満たすレコードの数を示しているにすぎません。

reorg_rows_rejected_for_compression - 圧縮がリジェクトされる行

レコード長が最小レコード長以下であったために再編成中に圧縮されなかった行数。

表 1494. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 レコードが最小レコード長以下の場合は圧縮されません。リジェクトされる行数は、この圧縮要件を満たせないこうしたレコードの連続したカウントを反映しています。

reorg_start 表再編成開始時刻

表の再編成の開始時刻です。マルチディメンション・クラスタリング (MDC) 表または挿入時クラスタリング (ITC) 表のエクステントを再利用するための再編成も含まれます。パーティション表の場合は、各データ・パーティションの再編成の開始時刻を示します。

表 1495. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_status 表再編成の状況

インプレース (オンライン) 表またはデータ・パーティション・レベル再編成の状況。これはクラシック (オフライン) 表再編成には適用できません。

表 1496. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 インプレース表またはデータ・パーティション再編成は、次の状態のいずれかになります (各状態は sqlmon.h での対応する定義とともに示しています)。

- 開始/再開: SQLM_REORG_STARTED
- 休止: SQLM_REORG_PAUSED
- 停止: SQLM_REORG_STOPPED
- 完了: SQLM_REORG_COMPLETED
- 切り捨て: SQLM_REORG_TRUNCATE

エクステントを再利用するインプレース表またはデータ・パーティション再編成は、次の状態のいずれかになります。

- 開始: SQLM_REORG_STARTED
- 停止: SQLM_REORG_STOPPED
- 完了: SQLM_REORG_COMPLETED

reorg_tbspc_id - 表またはデータ・パーティションが再編成される表スペース

表が再編成される表スペース。パーティション化されている表の場合、それぞれのデータ・パーティションが再編成される表スペースも示します。

表 1497. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_type 表再編成の属性

表再編成の属性設定値。

表 1498. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 属性設定値として次のものがあります。各属性設定は、 db2ApiDf.h で定義されるビット・フラグ値に基づいています。

- 書き込みアクセスの許可: DB2REORG_ALLOW_WRITE
- 読み取りアクセスの許可: DB2REORG_ALLOW_READ

- アクセスを許可しない: DB2REORG_ALLOW_NONE
- 索引スキャンを介した再クラスタリング: DB2REORG_INDEXSCAN
- LONG フィールド LOB データの再編成: DB2REORG_LONGLOB
- 表の切り捨てなし: DB2REORG_NOTRUNCATE_ONLINE
- コンプレッション・ディクショナリーの置換: DB2REORG_RESET_DICTIONARY
- コンプレッション・ディクショナリーの維持: DB2REORG_KEEP_DICTIONARY
- エクステントの再利用: DB2REORG_RECLAIM_EXTS

前記の属性設定値に加えて、 GET SNAPSHOT FOR TABLES コマンドの CLP 出力に以下の属性が示されます。これらの属性設定値は、他の属性設定値や表再編成に関するモニター・エレメントの値に基づいています。

- 再クラスタリング: reorg_index_id モニター・エレメントの値がゼロ以外の場合は、表再編成処理はこの属性を持ちます。
- 再利用: reorg_index_id モニター・エレメントの値がゼロの場合は、表再編成処理はこの属性を持ちます。
- インプレース表再編成: reorg_status モニター・エレメントの値が非 NULL の場合は、インプレース (オンライン) 再編成方式が使用されています。
- 表の再編成: reorg_phase モニター・エレメントの値が非 NULL の場合は、クラシック (オフライン) 再編成方式が使用されています。
- 表スキャンを介した再クラスタリング: DB2REORG_INDEXSCAN フラグが設定されていない場合は、表再編成処理はこの属性を持ちます。
- データのみの再編成: DB2REORG_LONGLOB フラグが設定されていない場合は、表再編成処理はこの属性を持ちます。

reorg_xml_regions_compressed - 圧縮された XML 領域 : モニター・エレメント

表の再編成プロセス時に圧縮された XML 領域の数。

表 1499. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_xml_regions_rejected_for_compression - 圧縮を拒否された XML 領域 : モニター・エレメント

表の再編成プロセス時に圧縮されなかった XML 領域の数。

表 1500. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

req_agent_tid - ロック取得を待機しているエージェントのスレッド ID : モニター・エレメント

ロックの取得を待機しているエージェントまたはシステム・エンティティーのスレッド ID。

表 1501. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

req_application_handle - ロック取得を待機しているアプリケーションの ID : モニター・エレメント

システム全体での、ロックの取得を待機しているアプリケーションのユニーク ID。

表 1502. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

req_executable_id - ロック取得を待機しているステートメント・セクションの ID : モニター・エレメント

ロックの取得を待機している SQL ステートメント・セクションを一意的に識別する、データ・サーバーで生成されたバイナリー・トークン。非 SQL アクティビティーの場合、長さが 0 のストリング値が戻されます。

表 1503. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

req_member - ロック取得を待機しているアプリケーションのメンバー : モニター・エレメント

このロックの取得を待機しているアプリケーションがあるデータベース・メンバー。

表 1504. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

request_exec_time_avg 要求の平均実行時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスに関連付けられた要求の実行時間の算術平均。内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスの COLLECT AGGREGATE REQUEST DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。単位はミリ秒です。

REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、request_exec_time_avg 平均は再マップに関係した部分要求をカウントします。

表 1505. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE REQUEST DATA

表 1506. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

この統計を使用すると、このサービス・サブクラス中のメンバー上での要求ごとの平均処理時間を即時に知ることができます。

この平均を使用して、要求の実行時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。要求の実行時間のヒストグラムから要求の平均実行時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、要求の実行時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

rf_log_num - ロールフォワードされているログ : モニター・エレメント

ロールフォワード操作で処理中のログ。

表 1507. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	基本

使用法

ロールフォワードが進行中の場合、このエレメントは関係するログを示します。DB2 pureScale 環境において、**rf_log_num** モニター・エレメントは、ロールフォワード操作に現在関係している、各ログ・ストリームのログ・ファイルを示します。

rf_status ログ・フェーズ

リカバリーの状況。

エレメント ID
rf_status

エレメント・タイプ
情報

表 1508. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	基本

使用法 このエレメントはリカバリーの進行状況を示します。リカバリーが取り消し (ロールバック) フェーズにあるのか、あるいは再実行 (ロールフォワード) フェーズにあるのかを示します。

rf_timestamp ロールフォワード・タイム・スタンプ

最後にコミットしたトランザクションのタイム・スタンプ。

エレメント ID
rf_timestamp

エレメント・タイプ
timestamp

表 1509. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	タイム・スタンプ

使用法 ロールフォワードが進行中の場合、これはロールフォワード・リカバリーが処理した、最後にコミットされたトランザクションのタイム・スタンプです。これは、どの程度ロールフォワード操作が進行したかを示す指標になります。

rf_type ロールフォワード・タイプ

進行中のロールフォワードのタイプ。

エレメント ID
rf_type

エレメント・タイプ
情報

表 1510. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	基本

使用法 データベース・レベルと表スペース・レベルのどちらでリカバリーが起きているかを示す標識です。

rollback_sql_stmts - 試行されたロールバック・ステートメント

試行された SQL ROLLBACK ステートメントの合計数。

エレメント ID
rollback_sql_stmts

エレメント・タイプ
カウンター

表 1511. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1512. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 ロールバックは、アプリケーション要求、デッドロック、またはエラー状態の結果として起こります。このエレメントでは、アプリケーションが発行したロールバック・ステートメント**のみ**カウントされます。

アプリケーション・レベルでは、このエレメントはアプリケーションのデータベース・アクティビティー・レベルとその他のアプリケーションとの競合

の量を判別するのに役立ちます。データベース・レベルでは、データベース内のアクティビティーの量とデータベース上でのアプリケーション間の競合の量を判別できます。

注: ロールバック・アクティビティーが多くなるとデータベースのスループットが低下するので、ロールバックの回数を最小限にとどめてください。

これを使用して、作業単位の総数を計算することもできます。これには、以下の式の和を計算します。

```
  commit_sql_stmts
+ int_commits
+ rollback_sql_stmts
+ int_rollbacks
```

rolled_back_agent_id ロールバックされたエージェント

デッドロックが発生したときにロールバックされたエージェント。

エレメント ID

rolled_back_agent_id

エレメント・タイプ

情報

表 1513. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	常に収集される

使用法 システム管理者はこの情報を基にして、更新を完了できなかったアプリケーションや再始動する必要があるアプリケーションを判別できます。

rolled_back_appl_id ロールバック・アプリケーション

デッドロックが発生したときにロールバックされたアプリケーション ID。

エレメント ID

rolled_back_appl_id

エレメント・タイプ

情報

表 1514. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	常に収集される

使用法 システム管理者はこの情報を基にして、更新を完了できなかったアプリケーションや再始動する必要があるアプリケーションを判別できます。

rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント

ロールバックされたアプリケーションを識別する参加者の番号。

表 1515. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
デッドロック[1]	event_deadlock	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

システム管理者はこの情報を基にして、更新を完了できなかったアプリケーションや始動する必要があるアプリケーションを判別できます。

rolled_back_sequence_no ロールバックされたシーケンス番号

デッドロックが発生したときにロールバックされたアプリケーションのシーケンス番号。

エレメント ID
rolled_back_sequence_no

エレメント・タイプ
情報

表 1516. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	常に収集される

使用法 システム管理者はこの情報を基にして、更新を完了できなかったアプリケーションや再始動する必要があるアプリケーションを判別できます。

root_node_splits - ルート・ノードの分割 : モニター・エレメント

挿入操作時に索引のルート・ノードが分割された回数。

表 1517. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE

routine_id - ルーチン ID : モニター・エレメント

固有なルーチン ID。このモニター・エレメントでは、アクティビティーがルーチンの一部ではない場合、0 が戻されます。

表 1518. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 1519. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	常に収集される
作業単位	uow_package_list	常に収集される
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE

使用法

このエレメントの値は、ビュー SYSCAT.ROUTINES の列 ROUTINEID の値と一致します。アクティビティーが別の SQL PL ルーチン内で宣言された SQL PL ルーチンの一部である場合、このエレメントの値は外部ルーチンの ROUTINEID です。

routine_module_name - ルーチン・モジュール名のモニター・エレメント

ルーチンが属するモジュールの非修飾名。

このエレメントは、ルーチンがモジュールに属していない場合には NULL になります。

表 1520. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される

routine_name - ルーチン名 : モニター・エレメント

ルーチンの非修飾名。

このエレメントは、サブルーチンと動的に準備されたコンパウンド SQL ステートメント、または PL/SQL の無名ブロック用にシステム生成される場合があります。

表 1521. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される

routine_schema - ルーチン・スキーマのモニター・エレメント

ルーチンがモジュールに属していない場合はルーチンのスキーマ名。属している場合はそのモジュールのスキーマ名。

このエレメントは、サブルーチンと動的に準備されたコンパウンド SQL ステートメント、または PL/SQL の無名ブロック用にシステム生成される場合があります。

表 1522. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1522. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される

routine_type - ルーチン・タイプのモニター・エレメント

ルーチンのタイプを示します。

ルーチン・タイプは以下のいずれかになります。

C 動的に準備されたコンパウンド SQL ステートメント、または PL/SQL 無名ブロック

F 関数

P プロシージャー

T トリガー

表 1523. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される

rows_deleted - 削除行数 : モニター・エレメント

これは、試行された行の削除の数です。

表 1524. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

表 1525. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1526. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、データベース内の現在のアクティビティーのレベルがわかります。

このカウントには、**int_rows_deleted** モニター・エレメントでカウントされる試行回数は含まれません。

rows_fetched フェッチ行数 : モニター・エレメント

表から読み取られた行の数。

このモニター・エレメントは、**rows_read** モニター・エレメントの別名です。

注: このモニター・エレメントは、この情報を記録する対象としたメンバーにおける値のみを報告します。複数メンバー・データベース環境の場合、これらの値は、アクティビティー全体での総量を正確に示していない場合があります。

表 1527. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	ステートメント

使用法

詳しくは、**rows_read** モニター・エレメントを参照してください。

rows_inserted - 挿入行数 : モニター・エレメント

試行された行の挿入の数。

表 1528. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

表 1529. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1530. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、データベース内の現在のアクティビティーのレベルがわかります。

フェデレーテッド・システムの場合は、状況によってはフェデレーテッド・サーバーが INSERT FROM SUBSELECT をデータ・ソースにプッシュできるので、 INSERT ステートメントごとに複数の行を挿入できます。

このカウントには、**int_rows_inserted** モニター・エレメントでカウントされる試行回数は含まれません。

rows_modified 変更行数 : モニター・エレメント

挿入、更新、または削除された行数。

このモニター・エレメントは、**rows_written** モニター・エレメントの別名です。

表 1531. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1531. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1532. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
アクティビティー	event_activity	ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

詳しくは、**rows_written** モニター・エレメントを参照してください。

rows_read - 読み取り行数 : モニター・エレメント

表から読み取られた行の数。

表 1533. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 1533. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	REQUEST METRICS BASE

表 1534. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
表	table	表
アプリケーション	appl	基本
アプリケーション	stmt	基本
アプリケーション	subsection	ステートメント
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1535. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
接続	event_conn	常に収集される
表	event_table	常に収集される
ステートメント	event_stmt	常に収集される
トランザクション	event_xact	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用すると、使用率が高く、新たな索引を追加する必要があるような表を識別できます。不要な索引の保守を回避するには、SQL EXPLAIN ステートメントを使用して、パッケージが索引を使用するかどうかを判別します。

このカウントは、呼び出し側のアプリケーションに戻された行数ではありません。結果セットを戻すために、読み取りが必要になった行数です。例えば、次のステートメントを使用すると、アプリケーションに戻されるのは 1 行ですが、平均給与を判別するために多数の行が読み取られます。

```
SELECT AVG(SALARY) FROM USERID.EMPLOYEE
```

このカウントには、**overflow_accesses** モニター・エレメントの値が含まれます。ただし、索引アクセスは含まれません。つまり、アクセス・プランが索引アクセスだけを使用して、実際の行を見るために表の読み取りを行わなければ、**rows_read** モニター・エレメントの値は増えません。

rows_returned 戻り行数 : モニター・エレメント

rows_returned モニター・エレメントは、選択されてアプリケーションに戻された行の数です。

このエレメントは、アクティビティー・レコードが部分的な場合 (例えば、アクティビティーがまだ実行中に収集された場合、またはメモリーの制約のために完全なアクティビティー・レコードをイベント・モニターに書き込むことができなかったとき) に、値が 0 になります。

このモニター・エレメントは、**fetch_count** モニター・エレメントの別名です。

表 1536. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1536. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	REQUEST METRICS BASE

表 1537. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
アクティビティー	event_activity	常に収集される

表 1537. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用すると、アプリケーションに戻される行数のしきい値を判別する助けになります。または、そのようなしきい値が正しく構成され、作動しているかを検証するために使用できます。

rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント

rows_returned_top モニター・エレメントは、サービス・クラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの実際の戻り行数の最高水準点です。

サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了したサービス・サブクラスの rows_returned_top 最高水準点のみが更新されます。アクティビティーがマップされたサービス・サブクラスでも、アクティビティーがそこで完了しなかった場合、そのサービス・サブクラスの最高水準点は何も影響を受けません。

表 1538. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーで到達した DML アクティビティーの実際の戻り行数の最大数を調べることができます。

rows_selected 選択行数

これは、選択されてアプリケーションに戻された行の数です。

表 1539. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1540. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、データベース内の現在のアクティビティーのレベルがわかります。

このエレメントには、 COUNT(*) や結合などのアクションで読み込まれた行のカウントは含まれません。

フェデレーテッド・システムの場合は、データ・ソースからフェデレーテッド・サーバーに行を戻すのに要する平均時間を計算できます。

```
平均時間 = 戻された行数 / 照会応答合計時間
```

この結果を使用すると、SYSCAT.SERVERS 内の CPU 速度または通信速度などのパラメーターを変更できます。これらのパラメーターを変更すると、オプティマイザーが要求をデータ・ソースに送信するかしないかに影響を与えます。

注: モニター対象のゲートウェイが DB2 データベース・バージョン 7.2 以前のものである場合は、このエレメントはスナップショット・モニターの dcs_dbase および dcs_appl 論理データ・グループで収集されます。

rows_updated - 更新行数 : モニター・エレメント

これは、試行された行の更新の数です。

表 1541. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED

表 1542. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1543. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、データベース内の現在のアクティビティーのレベルがわかります。

この値には、**int_rows_updated** モニター・エレメントでカウントされる更新は含まれません。ただし、複数の UPDATE ステートメントにより更新された行は、更新ごとにカウントされます。

rows_written 書き込み行数

これは、表内で変更 (挿入、削除、または更新) された行の数です。

表 1544. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
アプリケーション	subsection	ステートメント
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1545. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される
表	event_table	常に収集される
ステートメント	event_stmt	常に収集される
トランザクション	event_xact	常に収集される

使用法 表レベルの情報で数値が高い場合は表の使用率が高いことを示しているため、 Run Statistics (RUNSTATS) ユーティリティーを使用して、この表に使用されるパッケージの効率を維持する必要があります。

アプリケーション接続およびステートメントでは、一時表で挿入、更新および削除があった行数がこのエレメントに含まれます。

アプリケーション、トランザクション、およびステートメントのレベルでこのエレメントを使用すると、相対的アクティビティー・レベルを解析したり、調整できる項目を見つけるのに便利です。

rqsts_completed_total - 完了した要求の合計 : モニター・エレメント

実行された要求の合計数。アプリケーション要求と内部要求の両方を含みます。サービス・サブクラスでは、このモニター・エレメントは要求が完了した場合にのみ更新されます。異なるサービス・サブクラス間を要求が移動した場合、2 回カウントされることはありません。

表 1546. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1547. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

savepoint_id - セーブポイント ID

作業単位内で設定されたセーブポイントの ID。

表 1548. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	DDLSTMTEXEC TXNCOMPLETION	常に収集される

sc_work_action_set_id サービス・クラス作業アクション・セット ID : モニター・エレメント

このアクティビティーがサービス・クラス有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスが所属する作業クラス・セットに関連した作業アクション・セットの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 1549. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1550. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントと **sc_work_class_id** エレメントを組み合わせて使用すると、アクティビティーのサービス・クラス作業クラスが存在する場合にはそれを一意的に識別できます。

sc_work_class_id サービス・クラス作業クラス ID : モニター・エレメント

このアクティビティーがサービス・クラス有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、このアクティビティーに割り当てられた作業クラスの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 1551. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1552. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントと **sc_work_action_set_id** エレメントを組み合わせて使用すると、アクティビティーのサービス・クラス作業クラスが存在する場合にはそれを一意的に識別できます。

sec_log_used_top 使用された最大 2 次ログ・スペース

使用された 2 次ログ・スペースの最大量 (バイト単位)。

表 1553. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1554. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 1555. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと *sec_logs_allocated* および *tot_log_used_top* を組み合わせて使用すると、 2 次ログへの現在の依存度が示されます。この値が高い場合は、より大きなログ・ファイル、またはより多くの 1 次ログ・ファイル、あるいはアプリケーション内でより頻度の高い COMMIT ステートメントが必要になります。

結果として、次の構成パラメーターの調整が必要になります。

- logfilsiz
- logprimary
- logsecond
- logarchmeth1

データベースに 2 次ログ・ファイルがまったくない場合は、この値はゼロになります。定義されていない場合もゼロになります。

注: データベース・システム・モニター情報はバイト単位で示されますが、構成パラメーターは各 4K バイトのページ単位で設定されます。

sec_logs_allocated 現在割り振られている 2 次ログ

データベースで現在使用されている 2 次ログ・ファイルの合計数。

表 1556. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1557. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントと *sec_log_used_top* および *tot_log_used_top* を組み合わせて使用すると、 2 次ログへの現在の依存度が分かります。この値が常に高い場合は、より大きなログ・ファイル、またはより多くの 1 次ログ・ファイル、あるいはアプリケーション内でより頻度の高い COMMIT ステートメントが必要になります。

結果として、次の構成パラメーターの調整が必要になります。

- logfilsiz
- logprimary
- logsecond
- logarchmeth1

section_actuals - セクション実行時統計 : モニター・エレメント

実行されたセクションに使用された実行時統計を含む、データ・サーバーで生成されるバイナリー・ストリング。セクションのキャプチャーまたは実行時統計の収集が有効ではない場合、この値は長さが 0 のストリングです。非 SQL アクティビティー (LOAD など) の場合、この値は長さが 0 のストリングです。

表 1558. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

section_actuals モニター・エレメントで収集されたデータ、または WLM_SET_CONN_ENV を使用して接続ごとに収集されたデータは、EXPLAIN_FROM_ACTIVITY ストアード・プロシージャーを使用してセクション EXPLAIN が実行されるときに使用されます。EXPLAIN 処理では、このデータを使用して、 EXPLAIN_ACTUALS Explain 表にデータを追加し、アクセス・プランのオペレーターに関する実行時統計を表します。

注:

- セクション実行時統計を使用できるのは、**section_actuals** データベース構成パラメーターを使用して有効にされている (BASE に設定されている) 場合、または WLM_SET_CONN_ENV ストアード・プロシージャーを使用して特定のアプリケーションに対して有効にされている場合のみです。このストアード・プロシージャーについて詳しくは、WLM_SET_CONN_ENV を参照してください。
- WLM_SET_CONN_ENV プロシージャーによってアプリケーションに対して指定された **section_actuals** 設定は、直ちに有効になります。

section_env セクション環境 : モニター・エレメント

SQL ステートメントのセクションを含む BLOB。これは、実際のセクション内容で、照会プランの実行可能形式です。

表 1559. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	常に収集される
パッケージ・キャッシュ	pkgcache	COLLECT DETAILED DATA

使用法

セクション Explain のプロシージャーと一緒にこのエレメントを使用すると、ステートメントを Explain して、ステートメントのアクセス・プランを表示させることができます。

section_number - セクション番号 : モニター・エレメント

静的 SQL ステートメントの、パッケージ内での内部セクション番号。

表 1560. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1560. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1561. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 1562. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-
パッケージ・キャッシュ	-	COLLECT BASE DATA

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

静的 SQL ステートメントの場合は、このエレメントと **creator**、**package_version_id**、および **package_name** モニター・エレメントを組み合わせて使用すると、次のサンプル照会を使用して、SYSCAT.STATEMENTS システム・カタログ表を照会し、静的 SQL ステートメント・テキストを取得できます。

```
SELECT SEQNO, SUBSTR(TEXT,1,120)
       FROM SYSCAT.STATEMENTS
       WHERE PKGNAME   = 'package_name' AND
             PKGSCHEMA = 'creator'      AND
             VERSION = 'package_version_id' AND
             SECTNO    = section_number
       ORDER BY SEQNO
```

注: 静的ステートメント・テキストを取得するときに、システム・カタログ表に対するこの照会が原因でロックの競合が生じることがあるので注意してください。この照会を使用するのは、できるだけデータベースに対するその他のアクティビティーが少ないときだけにしてください。

section_type - セクション・タイプ標識 : モニター・エレメント

SQL ステートメント・セクションが動的であるかまたは静的であるかを示します。

表 1563. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1564. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

このモニター・エレメントに可能な値は以下のとおりです。

- D: 動的
- S: 静的

select_sql_stmts 実行された選択 SQL ステートメント

実行された SQL SELECT ステートメントの数。

表 1565. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
表スペース	tablespace	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1566. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、アプリケーション・レベルまたはデータベース・レベルのデータベース・アクティビティーのレベルを判別できます。

次の公式を使用すると、すべてのステートメントに対する SELECT ステートメントの比率を計算できます。

```
  select_sql_stmts
/ ( static_sql_stmts
  + dynamic_sql_stmts )
```

この情報は、アプリケーションのアクティビティーおよびスループットの分析に役立ちます。

select_time 照会応答時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの照会に対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

このモニターは最新の値を格納します。

注: 照会ブロッキングが行われるため、フェデレーテッド・サーバーが行の読み取りを試行しても、その通信がすべて処理されるとは限りません。取得を要求した次の行は、戻される行のブロックに入っている可能性があります。そのため、照会応答合計時間は、データ・ソースでの処理を示すとは限らず、実際にはデータ・ソースまたはクライアントでの処理を示します。

表 1567. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このデータ・ソースのデータを待機した実際の時間を判別できます。この情報は、キャパシティー・プランニング、CPU のチューニング、および SYSCAT.SERVERS の通信速度を調整するときに役に立ちます。これらのパラメーターを変更すると、オプティマイザーが要求をデータ・ソースに送信するかしないかに影響を与えます。

応答時間は、フェデレーテッド・サーバーがデータ・ソースから行を要求してからフェデレーテッド・サーバーがその行を利用できるようになるまでの時間です。

sequence_no シーケンス番号 : モニター・エレメント

作業単位が終了するごとに (つまり、 COMMIT または ROLLBACK が作業単位を終了するごとに) この ID が増加します。 **appl_id** と **sequence_no** を使用してトランザクションを一意的に識別します。

表 1568. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本

表 1568. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

表 1569. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-
接続	event_connheader	-
ステートメント	event_stmt	-
トランザクション	event_xact	-
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-
詳細付きデッドロック履歴	event_detailed_dlconn	-
詳細付きデッドロック履歴	event_stmt_history	-
詳細付きデッドロック履歴値	event_detailed_dlconn	-
詳細付きデッドロック履歴値	event_stmt_history	-

sequence_no_holding_lk ロックを保持しているシーケンス番号

このアプリケーションが取得のために待機しているオブジェクトのロックを保留しているアプリケーションのシーケンス番号。

エレメント ID

sequence_no_holding_lk

エレメント・タイプ

情報

表 1570. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本
ロック	appl_lock_list	基本

表 1571. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	常に収集される
詳細付きデッドロック	event_detailed_dlconn	常に収集される

使用法 この ID と appl_id を組み合わせて使用すると、このアプリケーションが取得しようと待機しているオブジェクトのロックを保留しているトランザクションを一意的に識別できます。

server_db2_type モニター対象 (サーバー) ノードのデータベース・マネージャーのタイプ

モニター中のデータベース・マネージャーのタイプを識別します。

表 1572. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

使用法 データベース・マネージャーについて、次の構成タイプの 1 つが含まれます。

API シンボリック定数
コマンド行プロセッサー出力

sqlf_nt_server
ローカルとリモート・クライアントを持つデータベース・サーバー

sqlf_nt_stand_req
ローカル・クライアントを持つデータベース・サーバー

API シンボリック定数は、組み込みファイルの *sqlutil.h* に定義されています。

server_instance_name サーバー・インスタンス名

スナップショットが取られるデータベース・マネージャー・インスタンスの名前。

エレメント ID
server_instance_name

エレメント・タイプ
情報

表 1573. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

表 1574. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	常に収集される

使用法 複数のデータベース・マネージャーのインスタンスが同一のシステム上にある場合、このデータ項目は、スナップショット呼び出しが行われたインスタンスを一意的に識別するために使用されます。この情報は、モニター出力を後で解析するためにファイルやデータベースに保管しており、データベース・マネージャー の各インスタンスごとにデータを区別する必要がある場合に役立ちます。

server_platform サーバーのオペレーティング・システム

データベース・サーバーが稼働中のオペレーティング・システム。

エレメント ID

server_platform

エレメント・タイプ

情報

表 1575. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 1576. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントを使用して、リモート・アプリケーションの問題判別を行えます。このフィールドの値は、ヘッダー・ファイルの *sqlmon.h* にあります。

server_prdid - サーバー製品/バージョン ID

サーバー上で実行中の製品とバージョン。

表 1577. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

表 1578. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-

使用法 PPPVVRRM の形式になっています。各部分の定義は次のとおりです。

PPP SQL です。

VV 2 桁でバージョン番号を示します (バージョン番号が 1 桁の場合には、高位の桁は 0 になります)。

RR 2 桁でリリース番号を示します (リリース番号が 1 桁の場合には、高位の桁は 0 になります)。

M 1 文字で修正レベルを示します (0-9 または A-Z)。

server_version サーバー・バージョン

情報を戻しているサーバーのバージョン。

表 1579. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

使用法

このフィールドは、データベース・システム・モニター情報を収集しているデータベース・サーバーのレベルを識別します。これにより、アプリケーションは、データを戻しているサーバーのレベルに基づいてデータを解釈することができます。有効な値は以下のとおりです。

SQLM_DBMON_VERSION1
データは、DB2 バージョン 1 によって戻されました。

SQLM_DBMON_VERSION2
データは、DB2 バージョン 2 によって戻されました。

SQLM_DBMON_VERSION5
データは、DB2 Universal Database™ バージョン 5 によって戻されました。

SQLM_DBMON_VERSION5_2
データは、DB2 Universal Database バージョン 5.2 によって戻されました。

SQLM_DBMON_VERSION6
データは、DB2 Universal Database バージョン 6 によって戻されました。

SQLM_DBMON_VERSION7
データは、DB2 Universal Database バージョン 7 によって戻されました。

SQLM_DBMON_VERSION8
データは、DB2 Universal Database バージョン 8 によって戻されました。

SQLM_DBMON_VERSION9
データは、DB2 for Linux, UNIX, and Windows バージョン 9 によって戻されました。

SQLM_DBMON_VERSION9_5
データは、DB2 for Linux, UNIX, and Windows バージョン 9.5 によって戻されました。

service_class_id サービス・クラス ID : モニター・エレメント

サービス・サブクラスのユニーク ID。作業単位の場合、この ID は、その作業単位を発行している接続が関連付けられているワークロードのサービス・サブクラス ID を表します。

表 1580. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 1581. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
ロッキング	-	常に収集される
作業単位	-	常に収集される
統計	event_histogrambin	常に収集される
統計	event_scstats	常に収集される

使用法

このエレメントの値は、ビュー SYSCAT.SERVICECLASSES の列 SERVICECLASSID の値と一致します。このエレメントを使用して、サービス・サブクラス名、または別のソースのサービス・サブクラスに関するリンク情報を検索します。例えば、サービス・クラス統計をヒストグラム・ビン・レコードと結合させます。

以下の条件が満たされている場合、このエレメントの値は 0 になります。

- このエレメントが、event_histogrambin 論理データ・グループでレポートされる。
- ヒストグラム・データが、サービス・クラスではないオブジェクトに関して収集される。

service_level - サービス・レベル・

DB2 インスタンスの現在の修正サービス・レベルを示します。

表 1582. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

service_subclass_name サービス・サブクラス名 : モニター・エレメント

サービス・サブクラスの名前。

表 1583. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE

表 1584. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE

表 1584. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
統計	event_scstats	常に収集される
統計	event_qstats	常に収集される

使用法

このエレメントを他のアクティビティー・エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと一緒に使用すると、サービス・クラスまたはしきい値キューの動作の分析をすることができます。

service_superclass_name サービス・スーパークラス名 : モニター・エレメント

サービス・スーパークラスの名前。

表 1585. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE

表 1585. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE

表 1586. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
統計	event_scstats	常に収集される
統計	event_qstats	常に収集される

使用法

このエレメントを他のアクティビティー・エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと一緒に使用すると、サービス・クラスまたはしきい値キューの動作の分析をすることができます。

session_auth_id セッション許可 ID : モニター・エレメント

このアプリケーションによって使用されている現在のセッション許可 ID。ワークロード管理アクティビティーのモニターでは、このモニター・エレメントは、アクティビティーがシステムに挿入された時のセッション許可 ID を記述します。

表 1587. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 1588. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
ロック	appl_lock_list	基本

表 1589. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
しきい値違反	event_activity	常に収集される
変更履歴	changesummary	常に収集される

使用法

このエレメントは、SQL ステートメントの準備、SQL ステートメントの実行、またはその両方を行うのにどの許可 ID が使用されているかを判別するのに役立ちます。このモニター・エレメントは、ストアード・プロシージャーの実行中に設定されたセッション許可 ID 値は報告しません。

shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数

割り振られたメモリーの境界から共有ワークスペースがオーバーフローした回数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1590. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1591. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントと shr_workspace_size_top を組み合わせて使用すると、オーバーフローを防止するのに共有ワークスペースのサイズを大きくする必要があるかどうかを判別できます。共有ワークスペースがオーバーフローする

と、パフォーマンスが低下するだけではなく、アプリケーションの共有メモリーから割り振られたほかのヒープでメモリー不足エラーが発生することがあります。

データベース・レベルでは、「共有ワークスペースの最大サイズ」のある共有ワークスペースとして報告された共有ワークスペースがこのエレメントの報告の対象となります。アプリケーション・レベルでは、現行アプリケーションが使用するワークスペースのオーバーフロー回数を示します。

shr_workspace_section_inserts 共有ワークスペース・セクション挿入数

共有ワークスペースへの、アプリケーションによる SQL セクション挿入数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1592. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1593. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 実行可能セクションの作業用コピーは、共有ワークスペース内に保管されます。このカウンターは、コピーが使用できなかったために挿入が必要だった場合を示します。

データベース・レベルでは、データベース内のすべての共有ワークスペースを対象に、すべてのアプリケーションでの累計挿入数を示します。アプリケーション・レベルでは、このアプリケーションの共有ワークスペース内にあるすべてのセクションを対象とした累計挿入数を示します。

shr_workspace_section_lookups 共有ワークスペース・セクション検索

共有ワークスペースでの、アプリケーションによる SQL セクション検索数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1594. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1595. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 各アプリケーションは、実行可能セクションの作業用コピーがある共有ワークスペースにアクセスできます。

このカウンターは、アプリケーションの特定のセクションを見つけるために共有ワークスペースがアクセスされた回数を示します。データベース・レベルでは、データベース内のすべての共有ワークスペースを対象に、すべてのアプリケーションでの累計検索数を示します。アプリケーション・レベルでは、このアプリケーションの共有ワークスペース内にあるすべてのセクションを対象とした累計検索数を示します。

このエレメントと「共有ワークスペース・セクション挿入数」を組み合わせて使用すると、共有ワークスペースのサイズを調整できます。共有ワークスペースのサイズをコントロールしているのは、 app_ctl_heap_sz 構成パラメーターです。

shr_workspace_size_top 最大共有ワークスペース・サイズ

共有ワークスペースが到達した最大サイズ。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1596. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

表 1597. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントは、データベースが活動化されて以降、データベースでワークロードを実行したときに必要となった共有ワークスペースの最大バイト数

を示します。データベース・レベルでは、すべての共有ワークスペースが到達した最大サイズを示します。アプリケーション・レベルでは、現行アプリケーションが使用する共有ワークスペースの最大サイズです。

共有ワークスペースがオーバーフローした場合、このエレメントは、オーバーフロー時に共有ワークスペースが到達した最大サイズになります。このような状態が発生したかどうかを確認するには、「共有ワークスペースのオーバーフロー回数」をチェックしてください。

共有ワークスペースがオーバーフローすると、アプリケーションの共有メモリーにあるほかのエンティティーからメモリーを一時的に借用します。この結果、これらのエンティティーでメモリー不足エラーが発生したり、パフォーマンスが低下することがあります。 APP_CTL_HEAP_SZ を大きくすると、オーバーフローの確率を低くすることができます。

skipped_prefetch_data_p_reads - スキップされたプリフェッチ (データ物理読み取り) のモニター・エレメント

既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップしたデータ・ページの数。

表 1598. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_*_p_reads** エレメントと一緒に、プリフェッチャーで取り出す予定であったページが、既にバッファー・プール内に存在したためにプリフェッチされなかった回数を示します。ページが既にバッファー・プール内に存在した理由としては、以下のようないくつかの理由が考えられます。

- 新しいページであったため、まだディスク上には作成されていなかった。
- 別のエージェントが同じページを必要としたため、別のプリフェッチ要求によってバッファー・プールにロードされた。この場合は、上記のケースと同様、スキップされたプリフェッチ要求の増加は問題ではありません。生成された追加のプリフェッチ要求が冗長だっただけだからです。
- プリフェッチャーがプリフェッチ操作を完了できるようになる前に、エージェントが直接ディスクからページを取り出した。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する

操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_*_p_reads エレメントは、読み取りがスキップされた理由にかかわらず、すべてのスキップされた読み取り要求を示します。プリフェッチャーがページを取り出せるようになる前に、同じ作業単位のエージェントが読み取りを実行したためにスキップされた要求数を調べるには、 **skipped_prefetch_uow_*_p_reads** モニター・エレメントを確認してください。

skipped_prefetch_index_p_reads - スキップされたプリフェッチ (索引物理読み取り) のモニター・エレメント

既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした索引ページの数。

表 1599. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_*_p_reads** エレメントと一緒に、プリフェッチャーで取り出す予定であったページが、既にバッファー・プール内に存在したためにプリフェッチされなかった回数を示します。ページが既にバッファー・プール内に存在した理由としては、以下のようないくつかの理由が考えられます。

- 新しいページであったため、まだディスク上には作成されていなかった。
- 別のエージェントが同じページを必要としたため、別のプリフェッチ要求によってバッファー・プールにロードされた。この場合は、上記のケースと同様、スキップされたプリフェッチ要求の増加は問題ではありません。生成された追加のプリフェッチ要求が冗長だっただけだからです。
- プリフェッチャーがプリフェッチ操作を完了できるようになる前に、エージェントが直接ディスクからページを取り出した。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構

成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_*_p_reads エレメントは、読み取りがスキップされた理由にかかわらず、すべてのスキップされた読み取り要求を示します。プリフェッチャーがページを取り出せるようになる前に、同じ作業単位のエージェントが読み取りを実行したためにスキップされた要求数を調べるには、 **skipped_prefetch_uow_*_p_reads** モニター・エレメントを確認してください。

skipped_prefetch_temp_data_p_reads - スキップされたプリフェッチ (一時データ物理読み取り) のモニター・エレメント

既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした TEMPORARY 表スペースのデータ・ページの数。

表 1600. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_*_p_reads** エレメントと一緒に、プリフェッチャーで取り出す予定であったページが、既にバッファー・プール内に存在したためにプリフェッチされなかった回数を示します。ページが既にバッファー・プール内に存在した理由としては、以下のようないくつかの理由が考えられます。

- 新しいページであったため、まだディスク上には作成されていなかった。
- 別のエージェントが同じページを必要としたため、別のプリフェッチ要求によってバッファー・プールにロードされた。この場合は、上記のケースと同様、スキップされたプリフェッチ要求の増加は問題ではありません。生成された追加のプリフェッチ要求が冗長だっただけだからです。
- プリフェッチャーがプリフェッチ操作を完了できるようになる前に、エージェントが直接ディスクからページを取り出した。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余

儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_*_p_reads エレメントは、読み取りがスキップされた理由にかかわらず、すべてのスキップされた読み取り要求を示します。プリフェッチャーがページを取り出せるようになる前に、同じ作業単位のエージェントが読み取りを実行したためにスキップされた要求数を調べるには、 **skipped_prefetch_uow_*_p_reads** モニター・エレメントを確認してください。

skipped_prefetch_temp_index_p_reads - スキップされたプリフェッチ (一時索引物理読み取り) のモニター・エレメント

既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした TEMPORARY 表スペースの索引ページの数。

表 1601. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_*_p_reads** エレメントと一緒に、プリフェッチャーで取り出す予定であったページが、既にバッファー・プール内に存在したためにプリフェッチされなかった回数を示します。ページが既にバッファー・プール内に存在した理由としては、以下のようないくつかの理由が考えられます。

- 新しいページであったため、まだディスク上には作成されていなかった。
- 別のエージェントが同じページを必要としたため、別のプリフェッチ要求によってバッファー・プールにロードされた。この場合は、上記のケースと同様、スキップされたプリフェッチ要求の増加は問題ではありません。生成された追加のプリフェッチ要求が冗長だっただけだからです。
- プリフェッチャーがプリフェッチ操作を完了できるようになる前に、エージェントが直接ディスクからページを取り出した。システムに十分な数のプリフェッチ

ャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_*_p_reads エレメントは、読み取りがスキップされた理由にかかわらず、すべてのスキップされた読み取り要求を示します。プリフェッチャーがページを取り出せるようになる前に、同じ作業単位のエージェントが読み取りを実行したためにスキップされた要求数を調べるには、 **skipped_prefetch_uow_*_p_reads** モニター・エレメントを確認してください。

skipped_prefetch_temp_xda_p_reads - スキップされたプリフェッチ (一時 XDA データ物理読み取り) のモニター・エレメント

既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした TEMPORARY 表スペースの XML ストレージ・オブジェクト (XDA) のデータ・ページの数。

表 1602. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_*_p_reads** エレメントと一緒に、プリフェッチャーで取り出す予定であったページが、既にバッファー・プール内に存在したためにプリフェッチされなかった回数を示します。ページが既にバッファー・プール内に存在した理由としては、以下のようないくつかの理由が考えられます。

- 新しいページであったため、まだディスク上には作成されていなかった。
- 別のエージェントが同じページを必要としたため、別のプリフェッチ要求によってバッファー・プールにロードされた。この場合は、上記のケースと同様、スキップされたプリフェッチ要求の増加は問題ではありません。生成された追加のプリフェッチ要求が冗長だっただけだからです。

- プリフェッチャーがプリフェッチ操作を完了できるようになる前に、エージェントが直接ディスクからページを取り出した。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_*_p_reads エレメントは、読み取りがスキップされた理由にかかわらず、すべてのスキップされた読み取り要求を示します。プリフェッチャーがページを取り出せるようになる前に、同じ作業単位のエージェントが読み取りを実行したためにスキップされた要求数を調べるには、 **skipped_prefetch_uow_*_p_reads** モニター・エレメントを確認してください。

skipped_prefetch_uow_data_p_reads - スキップされたプリフェッチ (作業単位のデータ物理読み取り) のモニター・エレメント

同じ作業単位のエージェントによって既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップしたデータ・ページの数。

表 1603. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_uow_*_p_reads** エレメントと一緒に、プリフェッチ要求対象だったページのうち、プリフェッチ要求を作成した作業単位と同じ作業単位のエージェントによって直接読み取られたページ数を示します。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンな

どの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_uow_index_p_reads - スキップされたプリフェッチ (作業単位の索引物理読み取り) のモニター・エレメント

同じ作業単位のエージェントによって既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした索引ページの数。

表 1604. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_uow_*_p_reads** エレメントと一緒に、プリフェッチ要求対象だったページのうち、プリフェッチ要求を作成した作業単位と同じ作業単位のエージェントによって直接読み取られたページ数を示します。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_uow_temp_data_p_reads - スキップされたプリフェッチ (作業単位の一時データ物理読み取り) のモニター・エレメント

同じ作業単位のエージェントによって既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした TEMPORARY 表スペースのデータ・ページの数。

表 1605. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_uow_*_p_reads** エレメントと一緒に、プリフェッチ要求対象だったページのうち、プリフェッチ要求を作成した作業単位と同じ作業単位のエージェントによって直接読み取られたページ数を示します。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_uow_temp_index_p_reads - スキップされたプリフェッチ (作業単位の一時索引物理読み取り) のモニター・エレメント

同期トランザクションによって既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした TEMPORARY 表スペースの索引ページの数。

表 1606. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 1606. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

skipped_prefetch_uow_temp_xda_p_reads - スキップされたプリフェッチ (作業単位の一時 XDA データ物理読み取り) のモニター・エレメント

同期トランザクションによって既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした TEMPORARY 表スペースの XML ストレージ・オブジェクト (XDA) のデータ・ページの数。

表 1607. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

skipped_prefetch_uow_xda_p_reads - スキップされたプリフェッチ (作業単位の XDA データ物理読み取り) のモニター・エレメント

同じ作業単位のエージェントによって既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした XML ストレージ・オブジェクト (XDA) のデータ・ページの数。

表 1608. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_uow_*_p_reads** エレメントと一緒に、プリフェッチ要求対象だったページのうち、プリフェッチ要求を作成した作業単位と同じ作業単位のエージェントによって直接読み取られたページ数を示します。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求

することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_xda_p_reads - スキップされたプリフェッチ (XDA 物理読み取り) のモニター・エレメント

既にバッファー・プールにロードされていたために、入出力サーバー (プリフェッチャー) がスキップした XML ストレージ・オブジェクト (XDA) のデータ・ページの数。

表 1609. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

このモニター・エレメントは、他の **skipped_prefetch_*_p_reads** エレメントと一緒に、プリフェッチャーで取り出す予定であったページが、既にバッファー・プール内に存在したためにプリフェッチされなかった回数を示します。ページが既にバッファー・プール内に存在した理由としては、以下のようないくつかの理由が考えられます。

- 新しいページであったため、まだディスク上には作成されていなかった。
- 別のエージェントが同じページを必要としたため、別のプリフェッチ要求によってバッファー・プールにロードされた。この場合は、上記のケースと同様、スキップされたプリフェッチ要求の増加は問題ではありません。生成された追加のプリフェッチ要求が冗長だっただけだからです。
- プリフェッチャーがプリフェッチ操作を完了できるようになる前に、エージェントが直接ディスクからページを取り出した。システムに十分な数のプリフェッチャーが構成されていない場合、あるいは別のタイプのプリフェッチのボトルネックが存在する場合、エージェントが直接ディスクからページを読み取ることを余儀なくされることがあります。 . 例えば OLTP システムの場合、本質的に、ほとんどのワークロードが通常はトランザクションのワークロードであるため、構成パラメーター **num_ioservers** を 1 に設定して最小数のプリフェッチャーを構成することがあります。ところが、表スキャンなどの、プリフェッチを使用する操作が実行されると、 1 つのプリフェッチャーでは処理が追いつかない場合があります。その結果、エージェントが直接ページを要求することになります。この動作は、本来プリフェッチャーが実行したはずの入出力にアプリケーションが対応することになるため、パフォーマンスの低下を招く可能性があります。この場

合は、構成パラメーター **num_ioservers** を調整してプリフェッチャー数を増やすことを検討してください。他に考えられる原因としては、プリフェッチ・サイズが大きすぎるためにプリフェッチ時間が通常より長くなっている、あるいは、**db2_parallel_io** レジストリー変数が設定されていないために、表スペース・コンテナー内の並列プリフェッチが制限されているなどの原因があります。

skipped_prefetch_*_p_reads エレメントは、読み取りがスキップされた理由にかかわらず、すべてのスキップされた読み取り要求を示します。プリフェッチャーがページを取り出せるようになる前に、同じ作業単位のエージェントが読み取りを実行したためにスキップされた要求数を調べるには、 **skipped_prefetch_uow_*_p_reads** モニター・エレメントを確認してください。

smallest_log_avail_node 使用可能なログ・スペースが最小のノード

このエレメントはグローバル・スナップショットの場合にだけ戻され、使用可能なログ・スペースが最も少ない (バイト数) ノードを示します。

エレメント ID

smallest_log_avail_node

エレメント・タイプ

情報

表 1610. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントと appl_id_oldest_xact を組み合わせて使用すると、データベースに十分なログ・スペースがあることを確認できます。グローバル・スナップショットでは、appl_id_oldest_xact、total_log_used、および total_log_available がこのノードの値に対応します。

snapshot_timestamp - スナップショットのタイム・スタンプ : モニター・エレメント

スナップショットが取得された日時。

sort_heap_allocated 割り振られたソート・ヒープの合計

スナップショットが取られたときに、選択したレベルのすべてのソートに割り振られたソート・ヒープ・スペース用のページ数の合計。

表 1611. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
データベース	dbase	基本

使用法 各ソートに割り振られたメモリー量は、利用可能なソート・ヒープ・サイズ

の一部だけの場合とすべての場合があります。ソート・ヒープ・サイズは各ソートで利用できるメモリー量を示し、 *sortheap* データベース構成パラメーターに定義されている値です。

1 つのアプリケーションが同時に複数のソートをアクティブにすることができます。例えば、副照会付きの SELECT ステートメントを使用すると、同時に複数のソートが行われる場合があります。

情報は 2 つのレベルで収集できます。

- データベース・マネージャーのレベルでは、データベース・マネージャー内のアクティブなすべてのデータベースのすべてのソートを対象に、割り振られたソート・ヒープ・スペースの合計を示す。
- データベース・レベルでは、1 つのデータベース内のすべてのソートを対象に、割り振られたソート・ヒープ・スペースの合計を示す。

通常のメモリーの見積もりにはソート・ヒープ・スペースは含まれません。過剰なソートが発生している場合は、ソート・ヒープに使用される追加のメモリー量をデータベース・マネージャーを実行するのに必要な基本メモリー量に加える必要があります。一般に、ソート・ヒープが大きくなるほど、ソート効率は高くなります。索引を正しく使用すると、ソートに必要な量を少なくできます。

データベース・マネージャー・レベルに戻された情報は、 *sheapthres* 構成パラメーターの調整に利用できます。エレメントの値が *sheapthres* 以上になっている場合は、 *sortheap* パラメーターに定義されているソート・ヒープをソートで完全に得られていないことを示します。

sort_heap_top ソート専用ヒープの最高水準点

データベース・マネージャーでの専用ソート・メモリーの最高水準点 (4 KB ページ単位)。

表 1612. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 このエレメントを使用して、 SHEAPTHRES 構成パラメーターが最適な値に設定されているかどうかを判別できます。例えば、この水準点が SHEAPTHRES に近づいたり超えている場合は、おそらく SHEAPTHRES を大きくする必要があります。これは、SHEAPTHRES を超えると専用ソートに与えられるメモリーが常に少なくなり、その結果として逆にシステム・パフォーマンスに影響を与える場合があるためです。

sort_overflows - ソート・オーバーフロー : モニター・エレメント

ソート・ヒープを使い果たし、一時記憶用のディスク・スペースが必要になった可能性のあるソートの合計数。

表 1613. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1614. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
動的 SQL	dynsql	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1615. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	ステートメント、ソート
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

データベース・レベルまたはアプリケーション・レベルでは、この値と **total_sorts** を組み合わせて使用すると、ディスクにオーバーフローしたソートのパーセンテージを計算できます。このパーセンテージが高い場合は、**sortheap** の値を大きくして、データベース構成を調整する必要があります。

ステートメント・レベルでこのエレメントを使用すると、大量のソートを必要とするステートメントを識別できます。このようなステートメントは、さらに調整を行って必要となるソート量を少なくすると効率が上がります。

ソートがオーバーフローすると、ソートにマージ・フェーズが必要となり、データをディスクに書き込む必要がある場合は入出力がさらに必要となるので、必要な処理時間が増えます。

このエレメントは、1 ステートメント、1 アプリケーション、または 1 つのデータベースにアクセスするすべてのアプリケーションについて情報を提供します。

sort_shrheap_allocated 現在割り振られているソート共有ヒープ

データベースに割り振られている共有ソート・メモリーの合計量。

表 1616. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを使用して、共有ソート・メモリーのしきい値を評価できます。この値が共有ソート・メモリーの現行しきい値より大幅に高いことや低いことが頻繁にある場合は、おそらく、しきい値を調整する必要があります。

注: 「共有ソート・メモリーしきい値」は、 SHEAPTHRES_SHR データベース構成パラメーターが 0 の場合は SHEAPTHRES データベース・マネージャー構成パラメーターの値で決まります。 0 でない場合は SHEAPTHRES_SHR の値で決まります。

sort_shrheap_top ソート共有ヒープの最高水準点

データベース全体の共有ソート・メモリーの最高水準点 (4 KB ページ単位)。

表 1617. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを使用して、 SHEAPTHRES (または SHEAPTHRES_SHR) が最適な値に設定されているかどうかを評価できます。例えば、この最高水準点が常に共有ソート・メモリーしきい値よりも大幅に低い場合は、おそらくこのしきい値を小さくしてデータベースの他の機能にメモリーを解放する必要があります。逆にこの最高水準点が共有ソート・メモリーしきい値に近づき始めたら、そのしきい値を大きくする必要がある場合があります。共有ソート・メモリーしきい値はハード・リミットなので余裕を持たせておくことは重要です。ソート・メモリーの合計量がそのしきい値に達したら、共有ソートは開始できなくなります。

このエレメントは、専用ソート・メモリーの最高水準点と組み合わせて使用すると、共有および専用ソートのしきい値をそれぞれ単独に設定する必要があるかどうかを判別することにも利用できます。SHEAPTHRES_SHR データベース構成オプションの値が 0 の場合は通常、共有ソート・メモリーしきい値は SHEAPTHRES データベース・マネージャー構成オプションの値で決まります。ただし専用ソート・メモリーと共有ソート・メモリーの最高水準点に大きな違いがある場合は、SHEAPTHRES をオーバーライドして、SHEAPTHRES_SHR を共有ソート・メモリーの最高水準点を基にした、より適切な値に設定する必要がある場合があります。

注: このエレメントは、ソート・メモリー・コントローラーによって付与されたソート予約要求の最高水準点をレポートします。付与される要求によって、常にメモリー割り振りが同じレベルになるわけではありません。ソート・ヒープのコンシューマーのみが、SQL 要求の処理中に、必要に応じて

メモリーを割り振る (付与された量まで) ことができるためです。このエレメントの値と共有ソート・メモリー・プールの最高水準点 (pool_watermark) との間に矛盾が生じるのは正常です。

source_service_class_id ソース・サービス・クラス ID : モニター・エレメント

このエレメントのしきい値違反レコードが生成された時に、アクティビティーから再マップしたサービス・サブクラスの ID。 しきい値アクションが REMAP ACTIVITY アクション以外の場合、このエレメントの値はゼロです。

表 1618. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントは、アクティビティーが再マップされたサービス・クラスをたどるのに使用できます。 これを使用して、特定のサービス・サブクラスからマップされたアクティビティー数の総計を計算することもできます。

sp_rows_selected ストアード・プロシージャーによって戻された行数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このアプリケーションのストアード・プロシージャーの処理の結果として、データ・ソースからフェデレーテッド・サーバーに送信された行の数が含まれています。

表 1619. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントは複数の目的に使用できます。次の公式を使用すると、ストアード・プロシージャー単位でデータ・ソースからフェデレーテッド・サーバーに送信された平均行数を計算できます。

```
   ストアード・プロシージャー単位の行数
 = 戻された行数
 / 呼び出されたストアード・プロシージャーの数
```

このアプリケーションについて、データ・ソースからフェデレーテッド・サーバーに行を戻すときの平均時間も計算できます。

```
平均時間 = ストアード・プロシージャー応答合計時間 / 戻された行数
```

specific_name - 特定名のモニター・エレメント

ルーチン・インスタンスの名前。

このエレメントは、システム生成されることがあります。

表 1620. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される

sql_chains 試行された SQL チェーンの数

ステートメント処理中に DB2 Connect ゲートウェイとホストの間で *n* 回のデータ伝送が行われる際の、 SQL ステートメントの数を表します。範囲 *n* は、*num_transmissions_group* エレメントで指定されます。

表 1621. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

例えば、チェーニングが ON の場合に、 PREP ステートメントと OPEN ステートメントがチェーニングされ、チェーンが合計 2 つの伝送を要する場合は、*sql_chains* は「1」と報告され、 *sql_stmts* は「2」と報告されます。

チェーニングが OFF の場合は、 *sql_chains* のカウントは、 *sql_stmts* のカウントと等しくなります。

使用法 このエレメントを使用すると、処理中に 2、3、 4 などのデータ伝送回数をいくつのステートメントが使用したかについて統計を得ることができます。(1 つのステートメントを処理するには、少なくとも送信と受信の 2 回以上のデータ伝送が必要です。) この統計により、データベース・レベルおよびアプリケーション・レベルでのデータベースやアプリケーションのアクティビティーやネットワーク・トラフィックの状態がより明確になります。

注: *sql_stmts* モニター・エレメントは、 SQL ステートメントのサーバーへの送信が試行される回数を表します。伝送レベルでは、同一カーソル中のすべてのステートメントは単一の SQL ステートメントとしてカウントされます。

sql_req_id SQL ステートメントの要求 ID

SQL ステートメントでの操作の要求 ID。

表 1622. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	-

使用法 最初のアプリケーションがデータベースに接続した後、データベース・マネージャーが SQL 操作を処理するごとに、この ID が増分します。この値はデータベース全体でユニークであり、ステートメント操作を一意的に識別します。

sql_reqs_since_commit 最終コミット後の SQL 要求

最後のコミット以降にサブミットされた SQL 要求の数。

表 1623. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントを使用すると、トランザクションの進行状況をモニターできます。

sql_stmts 試行された SQL ステートメントの数

データ伝送スナップショットの場合、このエレメントは、ステートメント処理中に DB2 Connect ゲートウェイとホストの間で *n* 回のデータ伝送が行われる際の、SQL ステートメントの数を表します。範囲 *n* は、 *num_transmissions_group* エレメントで指定されます。

表 1624. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本
データ伝送	stmt_transmissions	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

DCS DATABASE スナップショットの場合、このステートメントのカウントは、データベースが活動化された後のステートメントの数になります。

DCS APPLICATION スナップショットの場合、このステートメントのカウントは、データベースへの接続がこのアプリケーションによって確立された後のステートメントの数になります。

使用法 データベース・レベルまたはアプリケーション・レベルでは、このエレメントを使用してデータベース・アクティビティーを測定します。ある一定の期

間について SQL ステートメントのスループットを計算するには、 2 つのスナップショットの間の経過時間の値でこのエレメントの値を割ります。

データ伝送レベルの場合: このエレメントを使用すると、処理中に 2、3、4 などのデータ伝送回数をいくつのステートメントが使用したかについて統計を得ることができます。 (1 つのステートメントを処理するには、少なくとも送信と受信の 2 回以上のデータ伝送が必要です。) この統計により、データベース・レベルおよびアプリケーション・レベルでのデータベースやアプリケーションのアクティビティーやネットワーク・トラフィックの状態がより明確になります。

注:

1. *sql_stmts* モニター・エレメントは、 SQL ステートメントのサーバーへの送信が試行される回数を表します。
   - アプリケーション・レベルおよびデータベース・レベルでは、カーソル中の個々の SQL ステートメントは個別にカウントされます。
   - 伝送レベルでは、同一カーソル中のすべてのステートメントは単一の SQL ステートメントとしてカウントされます。

sqlca SQL 連絡域 (SQLCA)

ステートメントの完了時にアプリケーションに戻された SQLCA データ構造体。

表 1625. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	-
アクティビティー	event_activity	-

使用法

SQLCA データ構造体を使用すると、ステートメントが正常に終了したかどうかを判別できます。 SQLCA の内容についての詳細は、「*SQL リファレンス 第 1 巻* 」の『SQLCA (SQL 連絡域)』 または「*管理 API リファレンス* 」の『SQLCA データ構造』を参照してください。

sqlrowsread_threshold_id - SQL 読み取り行数しきい値 ID : モニター・エレメント

アクティビティーに適用されていた SQLROWSREAD しきい値の ID。

表 1626. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 SQLROWSREAD しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

sqlrowsread_threshold_value - SQL 読み取り行数しきい値 : モニター・エレメント

アクティビティーに適用されていた SQLROWSREAD しきい値の上限。

表 1627. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、SQLROWSREAD しきい値がアクティビティーに適用されている場合、その値を判別します。

sqlrowsread_threshold_violated - SQL 読み取り行数しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが SQLROWSREAD しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 1628. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた SQLROWSREAD しきい値にアクティビティーが違反したかどうかを判別します。

sqlrowsreadinsc_threshold_id - サービス・クラス内の SQL 読み取り行数しきい値 ID : モニター・エレメント

アクティビティーに適用されていた SQLROWSREADINSC しきい値の ID。

表 1629. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 SQLROWSREADINSC しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

sqlrowsreadinsc_threshold_value - サービス・クラス内の SQL 読み取り行数しきい値 : モニター・エレメント

アクティビティーに適用されていた SQLROWSREADINSC しきい値の上限。

表 1630. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、SQLROWSREADINSC しきい値がアクティビティーに適用されている場合、その値を判別します。

sqlrowsreadinsc_threshold_violated - サービス・クラス内の SQL 読み取り行数しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが SQLROWSREADINSC しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 1631. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた SQLROWSREADINSC しきい値にアクティビティーが違反したかどうかを判別します。

sqlrowsreturned_threshold_id - 戻される SQL 読み取り行数しきい値 ID : モニター・エレメント

アクティビティーに適用されていた SQLROWSRETURNED しきい値の ID。

表 1632. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 SQLROWSRETURNED しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

sqlrowsreturned_threshold_value - 戻される SQL 読み取り行数しきい値 : モニター・エレメント

アクティビティーに適用されていた SQLROWSRETURNED しきい値の上限。

表 1633. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、SQLROWSRETURNED しきい値がアクティビティーに適用されている場合、その値を判別します。

sqlrowsreturned_threshold_violated - 戻される SQL 読み取り行数しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが SQLROWSRETURNED しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 1634. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた SQLROWSRETURNED しきい値にアクティビティーが違反したかどうかを判別します。

sqltempspace_threshold_id - SQL 一時スペースしきい値 ID : モニター・エレメント

アクティビティーに適用されていた SQLTEMPSPACE しきい値の ID。

表 1635. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、 SQLTEMPSPACE しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

sqltempspace_threshold_value - SQL 一時スペースしきい値 : モニター・エレメント

アクティビティーに適用されていた SQLTEMPSPACE しきい値の上限。

表 1636. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、SQLTEMPSPACE しきい値がアクティビティーに適用されている場合、その値を判別します。

sqltempspace_threshold_violated - SQL 一時スペースしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが SQLTEMPSPACE しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 1637. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた SQLTEMPSPACE しきい値にアクティビティーが違反したかどうかを判別します。

spacemappage_page_reclaims_x - スペース・マップ・ページ再利用の排他的アクセス : モニター・エレメント

スペース・マップ・ページに関連したページが、DB2 pureScale内の別のメンバーによって計画的な解放より前に再利用された回数。そのページを再利用したメンバーは、スペース・マップ・ページに対する排他的アクセスを必要としました。

表 1638. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

この値は、オブジェクト関連の表スペースに関してのみレポートされます。オブジェクト関連の表スペースとは、再利用可能なストレージで使用可能な表スペースです。**reclaimable_space_enabled** モニター・エレメントを使用して、再利用可能なストレージに対して表スペースが使用可能かどうかを判別してください。

エクステント・マップ・ページ (EMP) はメタデータであるため、このモニター・エレメントの値には EMP が含まれています。

データ・スペース・マップ・ページにはユーザー・データが含まれているので、**page_reclaims_x** モニター・エレメントの値に入っていると同時に、**spacemappage_page_reclaims_x** モニター・エレメントの値にも含まれます。索引ス

ペース・マップ・ページにはユーザー・データが含まれていないので、**spacemappage_page_reclaims_x** モニター・エレメントの値にしか含まれません。

spacemappage_page_reclaims_s - スペース・マップ・ページ再利用の共有アクセス : モニター・エレメント

スペース・マップ・ページに関連したページが、DB2 pureScale内の別のメンバーによって計画的な解放より前に再利用された回数。そのページを再利用したメンバーは、スペース・マップ・ページに対する共有アクセスを必要としました。

表 1639. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

この値は、オブジェクト関連の表スペースに関してのみレポートされます。オブジェクト関連の表スペースとは、再利用可能なストレージで使用可能な表スペースです。**reclaimable_space_enabled** モニター・エレメントを使用して、再利用可能なストレージに対して表スペースが使用可能かどうかを判別してください。

エクステント・マップ・ページ (EMP) はメタデータであるため、このモニター・エレメントの値には EMP が含まれています。

データ・スペース・マップ・ページにはユーザー・データが含まれているので、**page_reclaims_s** モニター・エレメントの値に入っていると同時に、**spacemappage_page_reclaims_s** モニター・エレメントの値にも含まれます。索引スペース・マップ・ページにはユーザー・データが含まれていないので、**spacemappage_page_reclaims_s** モニター・エレメントの値にしか含まれません。

spacemappage_page_reclaims_initiated_x - 排他的アクセスで開始されたスペース・マップ・ページ再利用 : モニター・エレメント

ページが別のメンバーから再利用される原因となった、スペース・マップ・ページのための排他モードでのページ・アクセス回数。

表 1640. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

この値は、オブジェクト関連の表スペースに関してのみレポートされます。オブジェクト関連の表スペースとは、再利用可能なストレージで使用可能な表スペースで

す。**reclaimable_space_enabled** モニター・エレメントを使用して、再利用可能なストレージに対して表スペースが使用可能かどうかを判別してください。

エクステント・マップ・ページ (EMP) はメタデータであるため、このモニター・エレメントの値には EMP が含まれています。

データ・スペース・マップ・ページにはユーザー・データが含まれているので、**page_reclaims_initiated_x** モニター・エレメントの値に入っていると同時に、**spacemappage_page_reclaims_initiated_x** モニター・エレメントの値にも含まれます。索引スペース・マップ・ページにはユーザー・データが含まれていないので、**spacemappage_page_reclaims_initiated_x** モニター・エレメントの値にしか含まれません。

spacemappage_page_reclaims_initiated_s - 共有アクセスで開始されたスペース・マップ・ページ再利用 : モニター・エレメント

ページが別のメンバーから再利用される原因となった、スペース・マップ・ページのための共有モードでのページ・アクセス回数。

表 1641. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

この値は、オブジェクト関連の表スペースに関してのみレポートされます。オブジェクト関連の表スペースとは、再利用可能なストレージで使用可能な表スペースです。**reclaimable_space_enabled** モニター・エレメントを使用して、再利用可能なストレージに対して表スペースが使用可能かどうかを判別してください。

エクステント・マップ・ページ (EMP) はメタデータであるため、このモニター・エレメントの値には EMP が含まれています。

データ・スペース・マップ・ページにはユーザー・データが含まれているので、**page_reclaims_initiated_s** モニター・エレメントの値に入っていると同時に、**spacemappage_page_reclaims_initiated_s** モニター・エレメントの値にも含まれます。索引スペース・マップ・ページにはユーザー・データが含まれていないので、**spacemappage_page_reclaims_initiated_s** モニター・エレメントの値にしか含まれません。

spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント

DB2 pureScale 環境では、このエレメントは、内部的に保守されているオブジェクト・スペース管理に関連したページのページ・ロックの待機時間を示します。その際、ロック要求によって、ページは他のメンバーが再利用します。この時間の測定単位はミリ秒です。

表 1642. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1642. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1643. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される

ss_exec_time サブセクション実行経過時間

サブセクションの実行に要した時間 (秒数)。

表 1644. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

表 1645. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	-

使用法 サブセクションの進行状況を追跡することができます。

ss_node_number サブセクション・ノード番号

サブセクションが実行されたノード。

表 1646. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

表 1647. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	-

使用法 各サブセクションとそれが実行されたデータベース・パーティションを関連付けるために使用します。

ss_number サブセクション番号 : モニター・エレメント

戻された情報に関連したサブセクションを示します。

表 1648. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE

表 1649. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

表 1650. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	常に収集される

使用法

この数値は、**db2expln** コマンドを使用して取得可能なアクセス・プラン内のサブセクション番号に関連しています。

ss_status サブセクションの状況 : モニター・エレメント

実行中のサブセクションの現在の状況。

表 1651. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

使用法

現在の状況の値として、次のものがあります。

- 実行中 (sqlmon.h の SQLM_SSEXEC)
- ロック待ち
- 表キュー (table queue) でデータの受信待ち
- 表キュー (table queue) でデータの送信待ち

ss_sys_cpu_time サブセクションに使用されたシステム CPU 時間

現在実行中のステートメント・サブセクションによって使用されたシステム CPU 時間の合計 (秒およびマイクロ秒単位)。表に書き込むイベント・モニターの場合、このエレメントの値は、BIGINT データ・タイプを使用して、マイクロ秒単位で示されます。

エレメント ID

ss_sys_cpu_time

エレメント・タイプ

time

表 1652. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	タイム・スタンプ

表 1653. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	タイム・スタンプ

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

ss_usr_cpu_time サブセクションに使用されたユーザー CPU 時間

現在実行中のステートメント・サブセクションによって使用されたユーザー CPU 時間の合計 (秒およびマイクロ秒単位)。表に書き込むイベント・モニターでは、このエレメントの値は、BIGINT データ・タイプを使用してマイクロ秒単位で指定されます。

エレメント ID

ss_usr_cpu_time

エレメント・タイプ

time

表 1654. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	タイム・スタンプ

表 1655. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	タイム・スタンプ

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

ssl_port_number - SSL ポート番号のモニター・エレメント

メンバーがクライアント接続を listen している SSL TCP/IP ポート。

表 1656. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVERLIST 表関数 - メンバーの優先順位の詳細を取得	常に収集される

start_event_id - 開始イベント ID

対応する UTILSTART イベントまたは UTILSTARTPROC イベントの固有 ID。

表 1657. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILSTOP	常に収集される

使用法

変更履歴イベント・モニターでは、対応するユーティリティー・イベント開始 (UTILSTART または UTILSTARTPROC) の固有 ID。このエレメントを START_EVENT_TIMESTAMP およびメンバー・エレメントと併用して、停止レコードを、対応する開始レコードと関連付けます。

start_event_timestamp - 開始イベント・タイム・スタンプ

対応する UTILSTART イベントまたは UTILSTARTPROC イベントの時刻。

表 1658. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILSTOP	常に収集される

使用法

変更履歴イベント・モニターでは、START_EVENT_ID およびメンバー・エレメントを併用して、停止レコードを対応する開始レコードと関連付けます。

start_time イベント開始時刻

作業単位開始、ステートメント開始、またはデッドロック検出の日時。このエレメントは、event_start API 構造内ではイベント・モニターの開始を示します。

表 1659. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_start	タイム・スタンプ
ステートメント	event_stmt	タイム・スタンプ
デッドロック	event_deadlock	タイム・スタンプ
デッドロック	event_dlconn	タイム・スタンプ
詳細付きデッドロック	event_detailed_dlconn	タイム・スタンプ

使用法 このエレメントを使用すると、デッドロック接続レコードとデッドロック・イベント・レコードを関連付けることができます。 *stop_time* と組み合わせて使用すると、ステートメントの経過時間またはトランザクション実行時間を計算できます。

注: 「タイム・スタンプ」スイッチが OFF のときは、このエレメントは「0」を報告します。

static_sql_stmts 試行された静的 SQL ステートメント

試行された静的 SQL ステートメントの数。

表 1660. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1661. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、データベース・レベルまたはアプリケーション・レベルで成功した SQL ステートメントの合計数を計算できます。

```
  dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= モニター期間中のスループット
```

statistics_timestamp 統計タイム・スタンプ : モニター・エレメント

この統計レコードが生成された時刻。

表 1662. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-
統計	event_wcstats	-
統計	event_qstats	-
統計	event_histogrambin	-

使用法

このエレメントを使用すると、この統計レコードが生成された時点を判別できます。

このエレメントと **last_wlm_reset** エレメントを組み合わせて使用すると、この統計レコードの統計が生成された時間間隔を識別できます。

このモニター・エレメントを使用すると、同じ収集間隔において生成されたすべての統計レコードをグループ化することもできます。

stats_cache_size - 統計キャッシュのサイズ : モニター・エレメント

統計キャッシュの現在のサイズ (バイト単位)。統計キャッシュは、リアルタイム統計収集により生成された統計情報をキャッシュに入れるためにカタログ・パーティションで使用されます。

注: 統計キャッシュはカタログ・パーティションにあるため、カタログ・パーティションで取られたスナップショットのみ統計キャッシュ・サイズを報告します。その他のパーティションで取られたスナップショットは、代わりにゼロの値を報告します。グローバル・スナップショットを取る際には、すべてのデータベース・パーティションで報告された値が集約されます。

表 1663. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
SNAP_GET_DB_V97 表関数 - dbase 論理グループからのスナップショット情報の検索	
SNAPDB 管理ビューおよび SNAP_GET_DB 表関数 - dbase 論理グループからのスナップショット情報の検索	

表 1664. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-

表 1665. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このエレメントを使用すると、現在の統計キャッシュのサイズを判別できます。この値は頻繁に変わります。システム使用量を評価するには、長期にわたり特定のインターバルを設けてスナップショットを取ってください。このエレメントを使用すると、**catalogcache_sz** 構成パラメーターの値を調整できます。

stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間 : モニター・エレメント

stats_fabricate_time モニター・エレメントは、リアルタイム統計収集により統計作成で費やされた合計時間 (ミリ秒単位) を格納します。統計作成とは、照会をコンパイルする際に、統計を生成するのに必要な統計収集アクティビティーのことです。

このモニター・エレメントがデータベース・レベルで収集される場合、データベース上で実行中のすべてのアプリケーションに対するリアルタイム統計収集アクティビティーで費やされた合計時間を表します。これがステートメント・レベルで収集される場合、そのステートメントの最新のリアルタイム統計収集アクティビティーで費やされた時間を表します。すべてのデータベース・パーティションで報告された時間は集約されます。

表 1666. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
SNAP_GET_DB_V97 表関数 - dbase 論理グループからのスナップショット情報の検索	
SNAPDB 管理ビューおよび SNAP_GET_DB 表関数 - dbase 論理グループからのスナップショット情報の検索	

表 1667. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このエレメントはリセットできます。

表 1668. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
ステートメント	event_stmt	常に収集される

使用法

このエレメントと **stats_fabrications** を組み合わせて使用すると、データベース・レベルのリアルタイム統計収集のパフォーマンスへの影響を評価できます。動的 SQL のスナップショット・モニターの場合、このエレメントと **total_exec_time** および **num_executions** を組み合わせて使用すると、統計作成の影響を評価できます。ステートメント・イベント・モニターの場合、このエレメントを **stmt_start** および **stmt_stop** と結合させて使用すると、リアルタイム統計収集の影響をさらに評価することができます。

stats_fabrications - 統計作成の合計数 : モニター・エレメント

stats_fabrications モニター・エレメントは、すべてのデータベース・アプリケーションに関する照会のコンパイル中にリアルタイム統計により処理される統計作成の合計数です。

表または索引に保管されているデータをスキャンして統計を取得するのではなく、統計は索引およびデータ・マネージャーによって保守されているメタデータに基づいて作成されます。すべてのデータベース・パーティションで報告された値が集約されます。

表 1669. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
SNAP_GET_DB_V97 表関数 - dbase 論理グループからのスナップショット情報の検索	
SNAPDB 管理ビューおよび SNAP_GET_DB 表関数 - dbase 論理グループからのスナップショット情報の検索	

表 1670. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1671. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このエレメントを使用すると、データベースの統計作成の頻度を判別できます。この値は頻繁に変わります。システム使用量の全体像をより正確に知るには、長期にわたり特定のインターバルを設けてスナップショットを取ってください。このエレメントと **stats_fabricate_time** を組み合わせて使用すると、統計作成の影響を評価する助けになります。

status_change_time アプリケーション状況変更時刻

アプリケーションが現在の状況になった日時。

エレメント ID

status_change_time

エレメント・タイプ

timestamp

表 1672. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	作業単位、タイム・スタンプ
ロック	appl_lock_list	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl_info	作業単位、タイム・スタンプ

使用法 このエレメントを使用して、アプリケーションが現在の状況になっている時間を判別できます。同じ状況が長時間にわたり継続している場合は、問題が起きている可能性があります。

stmt_elapsed_time 最新のステートメント経過時間

最後に完了したステートメントの実行経過時間。

表 1673. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ

使用法

ステートメントの完了にかかる時間の標識として、このエレメントを使用します。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

stmt_exec_time - ステートメント実行時間 : モニター・エレメント

このメンバーのすべてのエージェントがステートメントを実行するのにかかった時間の合計。値はミリ秒単位で示されます。

表 1674. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1675. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

stmt_first_use_time - ステートメントの最初の使用のタイム・スタンプ : モニター・エレメント

このエレメントは、ステートメント項目が最初に処理されたときを示します。カーソル操作の場合、**stmt_first_use_time** はカーソルがオープンされたときを示します。アプリケーション調整ノードでは、この値はアプリケーション要求を反映します。非コーディネーター・ノードでは、この値は要求が起点ノードから受信されたときを反映します。

表 1676. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	timestamp

表 1676. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴[1]	event_stmt_history	timestamp
アクティビティー	event_activitystmt	timestamp

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_history_id ステートメント履歴 ID

この数値エレメントは、sequence_no エレメントで示された作業単位内でステートメントが実行された位置を、他のステートメント履歴エレメントとの相対位置で示します。作業単位内で最も早く実行されるエレメントは、最も低い値を持ちます。

同じ作業単位内で同じステートメントが 2 回実行される場合、2 つの異なる stmt_history_id 値を持つステートメントが 2 箇所示されます。

表 1677. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	-
詳細付きデッドロック履歴値	event_data_value	-
詳細付きデッドロック履歴	event_stmt_history	-

使用法 このステートメントを使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_history_list_size - ステートメント履歴リストのサイズ

履歴付きのデッドロック詳細イベント・モニターが実行している場合、このエレメントは、ステートメント履歴リスト項目を追跡するために、データベース・モニター・ヒープ (MON_HEAP_SZ) から使用されているバイト数を報告します。

表 1678. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	-
データベース	db	-

使用法 このエレメントは、データベース・モニター・ヒープをチューニングする際に使用できます。

stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント

ルーチンの 1 つの呼び出しを、作業単位内の同じネスト・レベルの他の呼び出しと区別する ID。その ID は特定のネスト・レベルに関して作業単位内で固有です。

表 1679. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1680. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
作業単位	パッケージ・リストに報告されます。	-

1 このオプションは推奨されなくなりました。このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを使用して、特定の SQL ステートメントが実行された呼び出しを一意に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_isolation ステートメント分離

このエレメントは、ステートメントが実行されていた間にそのステートメントに対して有効だった、分離値を示します。

表 1681. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	-
詳細付きデッドロック履歴	event_stmt_history	-
アクティビティー	event_activitystmt	-

考えられる分離レベル値は以下のとおりです。

- SQLM_ISOLATION_LEVEL_NONE 0 (分離レベルが指定されていない)
- SQLM_ISOLATION_LEVEL_UR 1 (非コミット読み取り)

- SQLM_ISOLATION_LEVEL_CS 2 (カーソル固定)
- SQLM_ISOLATION_LEVEL_RS 3 (読み取り固定)
- SQLM_ISOLATION_LEVEL_RR 4 (反復可能読み取り)

使用法 このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因と、特定の SQL ステートメントの実行の動作を理解することができます。

stmt_last_use_time - ステートメント最終使用時タイム・スタンプ : モニター・エレメント

このエレメントは、ステートメント項目が最後に処理されたときを示します。

カーソル操作の場合、**stmt_last_use_time** は、カーソルに対する最後のアクションの時刻を示します。そのときのアクションとして、オープン、フェッチ、またはクローズが考えられます。アプリケーション調整ノードでは、この値はアプリケーション要求を反映します。非コーディネーター・ノードでは、この値は要求が起点ノードから受信されたときを反映します。

表 1682. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	timestamp
詳細付きデッドロック履歴[1]	event_stmt_history	timestamp
アクティビティー	event_activitystmt	timestamp

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント

このエレメントは、ステートメントが実行されていた間にそのステートメントに対して有効だった、ロック・タイムアウト値を示します。

表 1683. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-

表 1683. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因と、特定の SQL ステートメントの実行の動作を理解することができます。

stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント

このエレメントは、ステートメントが実行されていた間にそのステートメントに対して有効だった、ネストまたは再帰のレベルを示します。ネストの各レベルは、ストアード・プロシージャーまたはユーザー定義関数 (UDF) のネストされた呼び出しまたは再帰的呼び出しに対応します。

表 1684. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1685. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
アクティビティー	event_activitystmt	-
作業単位	パッケージ・リストに報告されます。	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを **stmt_invocation_id** モニター・エレメントと一緒に使用して、特定の SQL ステートメントが実行された呼び出しを一意に識別できます。また、

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_node_number ステートメント・ノード

ステートメントが実行されたノード。

表 1686. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

使用法 各ステートメントをそのステートメントが実行されたノードと関連付けるときに使用します。

stmt_operation/operation ステートメント操作 : モニター・エレメント

現在処理中または (現在実行中のものがない場合は) 最後に処理されたステートメント操作。

表 1687. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
SNAPSHOT_STATEMENT 表関数	ACTIVITY METRICS BASE
SNAPSTMT 管理ビューおよび SNAP_GET_STMT 表関数 - ステートメント・スナップショット情報の取得	ACTIVITY METRICS BASE

表 1688. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 1689. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される
ステートメント	event_stmt	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを使用すると、実行中の操作または最後に終了した操作を判別できます。

以下の値のどれかになります。

SQL 操作の場合:

- SELECT
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- OPEN
- FETCH
- CLOSE
- DESCRIBE
- STATIC COMMIT
- STATIC ROLLBACK
- FREE LOCATOR
- PREP_COMMIT
- CALL
- PREP_OPEN
- PREP_EXEC
- COMPILE
- DROP PACKAGE

非 SQL 操作の場合:

- RUN STATISTICS
- REORG
- REBIND
- REDISTRIBUTE
- GET TABLE AUTHORIZATION
- GET ADMINISTRATIVE AUTHORIZATION

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル `sqlmon.h` を参照してください。

stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID : モニター・エレメント

このエレメントは、動的 SQL ステートメントの内部パッケージ・キャッシュ ID を示します。

表 1690. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1690. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1691. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

表 1692. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
詳細付きデッドロック履歴値[1]	event_stmt_history	常に収集される
詳細付きデッドロック履歴[1]	event_stmt_history	常に収集される
アクティビティー	event_activitystmt	常に収集される
パッケージ・キャッシュ	-	COLLECT BASE DATA

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

複数パーティション環境では、各パーティションに、キャッシュされたステートメントに対する固有のステートメント ID があります。特定のステートメントが、複数のパーティションにわたって同一の ID を持つことはできません。

グローバルな動的 SQL スナップショットでは、最初のステートメント ID のみが戻されます。

stmt_query_id ステートメント照会 ID : モニター・エレメント

このエレメントは、カーソルとして使用された SQL ステートメントに付けられた内部照会 ID を示します。

表 1693. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-

表 1693. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-

使用法

このエレメントを **stmt_nest_level** モニター・エレメントと一緒に使用して、特定の SQL ステートメントの呼び出しを固有に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することもできます。

stmt_sorts ステートメント・ソート回数

stmt_operation を処理するためにデータ集合がソートされた合計回数。

表 1694. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント
アプリケーション	stmt	ステートメント
動的 SQL	dynsql	ステートメント

使用法 このエレメントを使用すると、索引が必要かどうかを識別できます。索引があればデータをソートする必要性を少なくできるからです。上記の表の関連エレメントを使用すると、このエレメントがソート情報を提供している SQL ステートメントを識別できます。次にこのステートメントを分析し、ソート対象の列を見ると索引候補を判別できます (例えば、 ORDER BY および GROUP BY 節に使用されている列、および結合列)。ソート効率を最適化するために索引が使用されているかどうかを確認する方法については、「*管理ガイド*」の『**EXPLAIN**』を参照してください。

このカウントには、ステートメントを実行するためにデータベース・マネージャーが内部的に生成する一時表のソートが含まれます。ソート数は、SQL ステートメントの最初の FETCH 操作と関連しています。この情報は、ステートメントの操作が最初の FETCH の場合にユーザーに戻されます。ブロック・カーソルの場合は、カーソルが開いたときに複数のフェッチが行われるので注意してください。このような場合、 DB2 が最初の FETCH を内部で発行している間にスナップショットをとる必要があるため、スナップショット・モニターを使用してソート回数を取得するのは困難になります。

ブロック・カーソルを使用して実行されたソートの数を確認するより確実な方法としては、ステートメントに宣言されたイベント・モニターを使用する方法があります。 CLOSE カーソルのステートメント・イベントにある total_sorts カウンターには、カーソルが定義されたステートメントを実行したときに実行されるソートの合計回数が含まれています。

stmt_source_id ステートメント・ソース ID

このエレメントは、実行された SQL ステートメントのソースに付けられた内部 ID を示します。

表 1695. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
アクティビティー	event_activitystmt	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを **appl_id** モニター・エレメントと一緒に使用して、特定の SQL ステートメントの実行要求の発信元を固有に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することもできます。

stmt_start ステートメント操作開始タイム・スタンプ

stmt_operation の実行開始日時。

エレメント ID
stmt_start

エレメント・タイプ
timestamp

表 1696. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ

使用法 このエレメントと stmt_stop を組み合わせて使用すると、ステートメント操作の実行経過時間を計算できます。

stmt_stop ステートメント操作停止タイム・スタンプ

stmt_operation の実行停止日時。

エレメント ID
stmt_stop

エレメント・タイプ
タイム・スタンプ

表 1697. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ

使用法 このエレメントと stmt_start を組み合わせて使用すると、ステートメント操作の実行経過時間を計算できます。

stmt_sys_cpu_time ステートメントが使用したシステム CPU 時間

現在実行中のステートメントによって使用されたシステム CPU 時間の合計 (秒およびマイクロ秒単位)。

エレメント ID
stmt_sys_cpu_time

エレメント・タイプ
time

表 1698. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント、タイム・スタンプ
アプリケーション	stmt	ステートメント、タイム・スタンプ

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

このカウンターには、SQL および非 SQL のステートメントに要した時間のほか、アプリケーションが実行した unfenced ユーザー定義関数 (UDF) およびストアード・プロシージャーも含まれます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

stmt_text - SQL ステートメント・テキスト : モニター・エレメント

SQL ステートメントのテキスト。

表 1699. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1700. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
動的 SQL	dynsql	基本
DCS ステートメント	dcs_stmt	ステートメント

表 1701. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される
詳細付きデッドロック履歴[1]	event_stmt_history	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activitystmt	常に収集される
パッケージ・キャッシュ	-	COLLECT BASE DATA
変更履歴	ddlstmtexec	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

アプリケーション・スナップショットの場合は、このステートメント・テキストに基づいて、スナップショットを取った時点でアプリケーションが何を実行していたかを識別できます。またスナップショットを取った時点でステートメントが処理されていなかった場合は、最後に処理されたものを識別できます。

このエレメントが戻す情報は、SQL ステートメント・キャッシュから取り出されるので、キャッシュがオーバーフローした場合は情報は得られません。ステートメントの SQL テキストを必ずキャプチャーするには、ステートメントのイベント・モニターを使用してください。

動的 SQL ステートメントの場合は、このエレメントを使用してパッケージに関連付けられた SQL テキストを識別します。

ステートメント・イベント・モニターの場合、このエレメントは動的ステートメントについてのみ戻されます。 ステートメント・イベント・モニター・レコードがステートメント・イベント・モニターの BUFFERSIZE オプションで指定されたバッファーのサイズに収まらない場合には、レコードが収まるように **stmt_text** モニターの値が切り捨てられることがあります。

EVENT_STMT_HISTORY イベント・モニターの場合、このエレメントは動的ステートメントについてのみ戻されます。 残されたイベント・モニターの場合、動的ステートメントと静的ステートメントの **stmt_text** は、SQL ステートメント・キャッシュ内で使用可能な場合にのみ戻されます。

パフォーマンスを考慮したために静的 SQL ステートメント・テキストが提供されない場合、これを取得するためにシステム・カタログ表を照会する方法については、**section_number** モニター・エレメントを参照してください。

stmt_type ステートメント・タイプ : モニター・エレメント

処理されるステートメントのタイプ。

表 1702. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 1703. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activitystmt	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを使用すると、実行中のステートメントのタイプを判別できます。次のいずれかのステートメントになります。

- 静的 SQL ステートメント。

- 動的 SQL ステートメント。
- SQL ステートメント以外の操作。例えば、バインドやプリコンパイルなどの操作。

スナップショット・モニターの場合は、このエレメントにより、現在処理中または最後に処理されたステートメントがわかります。

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル `sqlmon.h` を参照してください。

stmt_type_id - ステートメント・タイプ ID : モニター・エレメント

ステートメント・タイプの ID。

表 1704. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1705. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

stmt_type_id モニター・エレメントが取る可能性のある値は以下のとおりです。

- `Statement not prepared`
- `DDL, (not Set Constraints)`
- `DDL, Set Constraints`
- `DML, Select`
- `DML, Insert/Update/Delete`
- `Authorization`
- `DML, Select (blockable)`
- `DML, Lock Table`
- `DML, Commit/Rollback`
- `Set environment`
- `DDL, Savepoint`
- `DDL, (declared user temp)`
- `Passthru support`
- `CALL`
- `Free locator`

- DML, Select with IUD
- DML, Select with IUD (blockable)
- Top-level SET, no SQL
- Top-level SET, reads SQL
- DDL, (issues internal commit)
- Top-level SET, modifies SQL
- Unknown

stmt_unicode - ステートメントのユニコード・フラグのモニター・エレメント

SQL ステートメントのユニコード・フラグ。値: Yes または No

表 1706. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	lock_participant_activities	

stmt_usr_cpu_time ステートメントに使用されたユーザー CPU 時間

現在実行中のステートメントによって使用されたユーザー CPU 時間の合計 (秒およびマイクロ秒単位)。

エレメント ID

stmt_usr_cpu_time

エレメント・タイプ

time

表 1707. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント、タイム・スタンプ
アプリケーション	stmt	ステートメント、タイム・スタンプ

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

このカウンターには、SQL および非 SQL のステートメントに要した時間のほか、アプリケーションが実行した unfenced ユーザー定義関数 (UDF) およびストアード・プロシージャーも含まれます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

stmt_value_data 値データ

このエレメントは、SQL ステートメントに対するデータ値のストリング表記です。LOB、LONG および構造化タイプ・パラメーターは空ストリングとして示されます。日付、時刻、およびタイム・スタンプ・フィールドは ISO フォーマットで記録されます。

表 1708. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1709. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	stmt_value_data	-
アクティビティー	event_activityvals	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することができます。

stmt_value_index 値索引

このエレメントは、SQL ステートメントで使用される入力パラメーター・マーカーまたはホスト変数の位置を表します。

表 1710. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1711. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	stmt_value_data	-

表 1711. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activityvals	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することができます。

stmt_value_isnull NULL 値の値 : モニター・エレメント

このエレメントは、SQL ステートメントに関連したデータ値が NULL 値かどうか、デフォルト値を指定するための拡張標識が使用されたかどうか、またはこのステートメント値が未割り当てであることを示します。

可能な値は以下のとおりです。

- 値が NULL でない場合は 0 (「いいえ」)
- 値が NULL である場合は 1 (「はい」)
- このステートメント値に対してデフォルトのための拡張標識値 (-5) が指定された場合は 2 (「デフォルト」)
- このステートメント値に対して未割り当てのための拡張標識値 (-7) が指定された場合は 3 (「未割り当て」)

表 1712. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1713. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	stmt_value_isnull	-
アクティビティー	event_activityvals	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することができます。

stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント

このエレメントは、提供された値がステートメント再最適化中に使用された値かどうかを示します。

ステートメントが再最適化され (例えば、REOPT BIND オプションの設定のため)、かつこの再最適化中に SQL コンパイラーへの入力として値が使用された場合、値「True」が戻されます。

表 1714. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1715. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_data_value	-
アクティビティー	event_activityvals	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを提供されたコンパイル環境と一緒に使用して、SQL コンパイラーによる SQL ステートメントの処理を完全に分析できます。

stmt_value_type 値タイプ : モニター・エレメント

このエレメントは、SQL ステートメントに関連したデータ値のタイプのストリング表記です。

表 1716. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1717. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	stmt_value_type	-
アクティビティー	event_activityvals	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することができます。

stmtno - ステートメント番号のモニター・エレメント

静的 SQL ステートメントの、パッケージ内でのステートメント番号。

このエレメントは、動的 SQL ステートメントの場合には「1」に設定されます。DDL ステートメントのステートメント番号が使用できない場合など、ステートメント番号が使用不可なときにはこのエレメントは「-1」に設定されます。

表 1718. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	常に収集される
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	常に収集される

表 1719. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
アクティビティー	event_activitystmt	常に収集される
パッケージ・キャッシュ・アクティビティー	event_pkgcache	常に収集される

使用法

静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS カタログ・ビューで使用されているものと同じです。

sto_path_free_size 自動ストレージ・パスのフリー・スペース : モニター・エレメント

このエレメントは、ストレージ・パスが指し示すファイル・システム上で使用可能なフリー・スペースの量を (バイト単位で) 示します。複数のストレージ・パスが同じファイル・システムを指す場合、空きサイズはそれらのストレージ・パス間で分割されません。

表 1720. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE

表 1721. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法

このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するノードごとのデータを収集することができます。

- **db_storage_path**
- **fs_used_size**
- **fs_total_size**
- **fs_id**

stop_time イベント停止時刻

ステートメントが実行を停止した日時。

表 1722. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	タイム・スタンプ

使用法 このエレメントと *start_time* を組み合わせて使用すると、ステートメントの実行経過時間を計算できます。

FETCH ステートメント・イベントの場合は、最後に正常なフェッチが行われた時刻です。

注: 「タイム・スタンプ」スイッチが OFF のときは、このエレメントは「0」を報告します。

storage_group_id - ストレージ・グループ ID

現行データベースで使用されているストレージ・グループを一意的に表す整数。

表 1723. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

使用上の注意

- ADMIN_GET_STORAGE_PATHS 表関数を使用している場合、ストレージ・グループ ID は、ストレージ・パスが定義されているストレージ・グループを示します。
- MON_GET_TABLESPACES 表関数を使用している場合、ストレージ・グループ ID は、表スペースが定義されているストレージ・グループを示します。

storage_group_name - ストレージ・グループ名

ストレージ・グループの名前。

表 1724. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

使用上の注意

- ADMIN_GET_STORAGE_PATHS 表関数を使用している場合、このモニター・エレメントは、ストレージ・パスが定義されているストレージ・グループを示します。
- MON_GET_TABLESPACES 表関数を使用している場合、このモニター・エレメントは、表スペースが定義されているストレージ・グループを示します。

stored_proc_time ストアード・プロシージャー時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからのストアード・プロシージャー・ステートメントに対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

表 1725. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

応答時間は、フェデレーテッド・サーバーがストアード・プロシージャーをデータ・ソースにサブミットしてからデータ・ソースがストアード・プロシージャーを処理したことを応答するまでの時間です。

使用法 このエレメントを使用すると、このデータ・ソースでストアード・プロシージャーの処理に要した実際の時間を判別できます。

stored_procs ストアード・プロシージャー数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースで呼び出したストアード・プロシージャーの合計数が含まれています。

表 1726. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、フェデレーテッド・データベースでローカルに行われたストアード・プロシージャーの呼び出し数、またはアプリケーションがフェデレーテッド・データベースに対して呼び出したストアード・プロシージャーの呼び出し数を判別できます。

subroutine_id - サブルーチン ID のモニター・エレメント

固有なサブルーチン ID。

このエレメントは、オブジェクトがサブルーチン以外の場合には NULL を戻します。

表 1727. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される

使用法

宣言されるプロシージャーには親と同じ外部 ROUTINE_ID 値があるので、このエレメントを使用してそれらを区別します。

swap_pages_in - ディスクからスワップインされたページ数のモニター・エレメント

システムの始動以降に、ディスクからスワップインされたページ数。 AIX および Linux システムの場合のみ報告されます。

表 1728. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

swap_pages_out - ディスクにスワップアウトされたページ数のモニター・エレメント

システムの始動以降に、ディスクにスワップアウトされたページ数。 AIX および Linux システムの場合のみ報告されます。

表 1729. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

swap_page_size - スワップ・ページ・サイズのモニター・エレメント

スワップ・スペースに使用されているページ・サイズ (バイト単位)。 AIX および Linux システムの場合のみ報告されます。

表 1730. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

sync_runstats - 同期 RUNSTATS アクティビティーの合計数 : モニター・エレメント

データベース内のすべてのアプリケーションのリアルタイム統計収集により起動される同期 RUNSTATS アクティビティーの合計数。この値には、同期 RUNSTATS コマンドにおいて、成功したものと成功しなかったものの両方が含まれます。すべてのデータベース・パーティションで報告された値が集約されます。

表 1731. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
SNAP_GET_DB_V97 表関数 - dbase 論理グループからのスナップショット情報の検索	
SNAPDB 管理ビューおよび SNAP_GET_DB 表関数 - dbase 論理グループからのスナップショット情報の検索	

表 1732. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1733. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このモニター・エレメントを使用すると、データベースのリアルタイム統計収集により起動された同期 RUNSTATS アクティビティーの数を判別できます。この値は頻繁に変わります。システム使用量をより正確に知るには、長期にわたり特定のインターバルを設けてスナップショットを取ってください。このエレメントと **sync_runstats_time** を組み合わせて使用すると、リアルタイム統計収集により起動された同期 RUNSTATS アクティビティーのパフォーマンスへの影響を評価する助けになります。

sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間 : モニター・エレメント

sync_runstats_time モニター・エレメントは、リアルタイム統計収集により起動される同期 RUNSTATS アクティビティーに費やされた合計時間 (ミリ秒単位) を格納します。

同期 RUNSTATS アクティビティーは、照会のコンパイル中に発生します。データベース・レベルでは、このモニター・エレメントは、リアルタイム統計収集により起動された、データベース上で実行中のすべてのアプリケーションに対する同期 RUNSTATS アクティビティーで費やされた合計時間を表します。ステートメント・レベルでは、リアルタイム統計収集により起動された、特定のステートメントに対する最新の同期 RUNSTATS アクティビティーで費やされた時間を表します。すべてのデータベース・パーティションで報告された値が集約されます。

表 1734. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
SNAP_GET_DB_V97 表関数 - dbase 論理グループからのスナップショット情報の検索	
SNAPDB 管理ビューおよび SNAP_GET_DB 表関数 - dbase 論理グループからのスナップショット情報の検索	

表 1735. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このエレメントはリセットできます。

表 1736. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
ステートメント	event_stmt	常に収集される

使用法

このエレメントと **sync_runstats** を組み合わせて使用すると、データベース・レベルでリアルタイム統計収集により起動された、同期 RUNSTATS アクティビティーのパフォーマンスへの影響を評価できます。

動的 SQL のスナップショット・モニターの場合、このエレメントを **total_exec_time** および **num_executions** と組み合わせて使用すると、同期 RUNSTATS の照会パフォーマンスへの影響を評価できます。

ステートメント・イベント・モニターの場合、このエレメントを **stmt_start** および **stmt_stop** と組み合わせて使用すると、リアルタイム統計収集の影響をさらに評価することができます。

system_auth_id - システム許可 ID : モニター・エレメント

接続のシステム許可 ID。

表 1737. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 1738. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	changesummary	常に収集される

system_cpu_time システム CPU 時間 : モニター・エレメント

データベース・マネージャーのエージェント・プロセス、作業単位、またはステートメントで使用されたシステム CPU 時間の合計 (秒およびマイクロ秒単位)。表に書き込むイベント・モニターの場合、このエレメントの値は、BIGINT データ・タイプを使用して、マイクロ秒単位で示されます。

ステートメント・モニター・スイッチまたはタイム・スタンプ・スイッチがオンになっていない場合は、このエレメントは収集されません。この場合には、このモニター・エレメントは代わりに -1 を表示します。

表 1739. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される
トランザクション	event_xact	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	常に収集される

使用法

このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

注: DB2 システムが統計を収集する際の細分度の違いのために、**total_exec_time** モニター・エレメントの値が、**system_cpu_time** モニター・エレメントの値と **user_cpu_time** モニター・エレメントの値の合計と等しくならないことがあります。この場合、**system_cpu_time** モニター・エレメントと **user_cpu_time** モニター・エレメントの合計の方が実際の実行時間の合計を正確に反映しています。

tab_file_id - 表ファイル ID : モニター・エレメント

表のファイル ID (FID)。

表 1740. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE_METRICS 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

tab_type - 表タイプ : モニター・エレメント

このインターフェースは、sqlmon.h 内の定義に基づいたテキスト ID を戻します。USER_TABLE、TEMP_TABLE、または CATALOG_TABLE のいずれかになります。

表 1741. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE_METRICS 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

table_file_id - 表ファイル ID : モニター・エレメント

表のファイル ID (FID)。

表 1742. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMINTEMPTABLES 管理ビューおよび ADMIN_GET_TEMP_TABLES 表関数 - 一時表の情報を取得する	ACTIVITY METRICS BASE
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

表 1743. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
表	table	基本
ロック	appl_lock_list	ロック
ロック	lock	ロック

表 1744. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	lock	常に収集される

使用法

スナップショット・モニターの場合、このエレメントは情報提供のみを目的としています。これは、データベース・システム・モニターの以前のバージョンとの互換性のために返されるものであり、表を一意的に識別しない可能性があります。**table_name** および **table_schema** モニター・エレメントを使用して、表を識別します。

MON_GET_LOCKS 表関数および MON_GET_APPL_LOCKWAIT 表関数では、このエレメントは、ロックが参照する表のファイル ID (FID) を表します。

table_name - 表名 : モニター・エレメント

表の名前。

表 1745. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_TAB_COMPRESS_INFO 表関数 - 圧縮節約量の見積もり	ACTIVITY METRICS BASE
ADMIN_GET_TAB_DICTIONARY_INFO 表関数 - 既存の表ディクショナリーのプロパティーの報告	ACTIVITY METRICS BASE
ADMINTABINFO 管理ビューおよび ADMIN_GET_TAB_INFO 表関数 - 表のサイズおよび状態に関する情報の検索	ACTIVITY METRICS BASE
ADMINTEMPCOLUMNS 管理ビューおよび ADMIN_GET_TEMP_COLUMNS 表関数 - 一時表の列情報を取得する	ACTIVITY METRICS BASE

表 1745. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
ADMINTEMPTABLES 管理ビューおよび ADMIN_GET_TEMP_TABLES 表関数 - 一時表の情報を取得する	ACTIVITY METRICS BASE
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

表 1746. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	ロック
ロック	lock_wait	ロック

表 1747. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
表	event_table	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントと **table_schema** を組み合わせて使用すると、リソースの競合の原因を判別できます。

アプリケーション・レベル、アプリケーション・ロック・レベル、およびデッドロック・モニター・レベルでは、現在別のアプリケーションにロックされているため

にロック待ちになっているアプリケーションの表を示します。スナップショット・モニターの場合、この項目が有効なのは、「lock」モニター・グループ情報が ON に設定され、さらにアプリケーションが表ロックの取得待ちであることを **lock_object_type** が示している場合に限ります。

オブジェクト・レベルのスナップショット・モニターの場合は、表レベルおよび行レベルのロックについてこの項目が戻されます。このレベルで報告される表は、このアプリケーションがこれらのロックを保留している表です。

表レベルのスナップショット・モニターおよびイベント・モニターの場合は、情報が収集された表を示します。一時表の場合、 **table_name** の形式は 『TEMP (*n*, *m*)』 です。

- *n* は表スペース ID
- *m* は **table_file_id** エレメント

table_scans - 表スキャン : モニター・エレメント

この表でのスキャンの数。

表 1748. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

table_schema - 表スキーマ名 : モニター・エレメント

表のスキーマ。

表 1749. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_TAB_COMPRESS_INFO 表関数 - 圧縮節約量の見積もり	ACTIVITY METRICS BASE
ADMIN_GET_TAB_DICTIONARY_INFO 表関数 - 既存の表ディクショナリーのプロパティーの報告	ACTIVITY METRICS BASE
ADMINTABINFO 管理ビューおよび ADMIN_GET_TAB_INFO 表関数 - 表のサイズおよび状態に関する情報の検索	ACTIVITY METRICS BASE
ADMINTEMPCOLUMNS 管理ビューおよび ADMIN_GET_TEMP_COLUMNS 表関数 - 一時表の列情報を取得する	ACTIVITY METRICS BASE

表 1749. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
ADMINTEMPTABLES 管理ビューおよび ADMIN_GET_TEMP_TABLES 表関数 - 一時表の情報を取得する	ACTIVITY METRICS BASE
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

表 1750. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	ロック
ロック	lock_wait	ロック

表 1751. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
表	event_table	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントと **table_name** を組み合わせて使用すると、リソースの競合の原因を判別できます。

アプリケーション・レベル、アプリケーション・ロック・レベル、およびデッドロック・モニター・レベルでは、現在別のアプリケーションにロックされているため

にロック待ちになっているアプリケーションの表のスキーマを示します。このエレメントは、アプリケーションが表ロックの取得待ちであることを **lock_object_type** が示している場合にのみ設定できます。アプリケーション・レベルおよびアプリケーション・ロック・レベルでのスナップショット・モニターの場合、この項目は「lock」モニター・グループ情報が ON に設定されている場合にのみ有効です。

オブジェクト・レベルのスナップショット・モニターの場合は、表レベルおよび行レベルのロックについてこの項目が戻されます。このレベルで報告される表は、このアプリケーションがこれらのロックを保留している表です。

表レベルのスナップショット・モニターおよびイベント・モニターの場合は、このエレメントは情報が収集された表スキーマを示します。一時表の場合、**table_schema** の形式は 『*<agent_id><auth_id>*』 です。

- *agent_id* は、一時表を作成しているアプリケーションのアプリケーション・ハンドルです。
- *auth_id* は、アプリケーションがデータベースに接続するときに使用する許可 ID です。

table_type - 表タイプ : モニター・エレメント

情報が戻される表のタイプ。

表 1752. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

表 1753. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 1754. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法

このエレメントを使用すると、情報が戻される表の識別に役立ちます。表がユーザー表またはシステム・カタログ表の場合は、 **table_name** および **table_schema** を使用して表を識別できます。

表のタイプは、次の値のいずれかになります。可能な値は、sqlmon.h ファイル内の定義に基づくテキスト・ストリングです。

USER_TABLE
ユーザー表。

TEMP_TABLE
一時表。表が使用された後に表がデータベース内に保持されない場合も、一時表に関する情報が戻されます。その場合でも、このタイプの表に関する情報は役に立ちます。

CATALOG_TABLE
システム・カタログ表。

tablespace_auto_resize_enabled - 表スペースの自動サイズ変更可能 : モニター・エレメント

このエレメントは、表スペースの自動サイズ変更が使用可能かどうかを記述します。値 1 は「はい」を意味し、値 0 は「いいえ」を意味します。

表 1755. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1756. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法

このエレメントは、DMS 表スペースおよび非一時自動ストレージ表スペースにのみ適用されます。このエレメントが 1 に設定されている場合、自動サイズ変更は使用可能です。表スペースの増加率および最大サイズについては、次のモニター・エレメントを参照してください。

- **tablespace_max_size**
- **tablespace_increase_size**
- **tablespace_increase_size_percent**

tablespace_content_type - 表スペースのコンテンツ・タイプ : モニター・エレメント

表スペース内のコンテンツのタイプ。

表 1757. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1758. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法

表スペース内のコンテンツのタイプ (sqlmon.h 内に定義) は、次の値のいずれかになります。

- すべてのタイプの永続データ。
  - REGULAR 表スペース: SQLM_TABLESPACE_CONTENT_ANY
  - LARGE 表スペース: SQLM_TABLESPACE_CONTENT_LARGE
- システム一時データ: SQLM_TABLESPACE_CONTENT_SYSTEMP
- ユーザー一時データ: SQLM_TABLESPACE_CONTENT_USRTEMP

tablespace_cur_pool_id - 現在使用中のバッファー・プール : モニター・エレメント

表スペースが現在使用中のバッファー・プールのバッファー・プール ID。

表 1759. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1760. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法 バッファー・プールは、それぞれ固有の整数で識別できます。このエレメントの値は、SYSCAT.BUFFERPOOLS ビューの BUFFERPOOLID 列の値と一致します。

tablespace_current_size 表スペースの現行サイズ

このエレメントは、表スペースの現行サイズをバイト数で示します。

表 1761. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 DMS および自動ストレージ表スペースの場合、このエレメントはすべての表スペース・コンテナーの合計サイズをバイト数で表します。この値は、表スペースの合計ページ (tablespace_total_pages) に表スペースのページ・サイズ (tablespace_page_size) を掛けた値に等しくなります。このエレメントは、SMS 表スペース、または一時自動ストレージ表スペースには適用されません。

自動ストレージ表スペースの表スペース作成時に、現行サイズが初期サイズと一致しないことがあります。現行サイズの値は、作成時の初期サイズのページ・サイズ、エクステント・サイズ、およびストレージ・パスの数をすべて掛け合わせた値の範囲内になります (通常は大きくなりますが、場合によ

ってては小さくなることもあります)。常に tablespace_max_size 以下になります (設定されている場合)。これは、コンテナーがフル・エクステント単位でしか大きくなれず、かつ複数のコンテナーがセットとして大きくならなければならないためです。

tablespace_extent_size - 表スペースのエクステント・サイズ : モニター・エレメント

表スペースが使用するエクステント・サイズ。

表 1762. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1763. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

tablespace_free_pages 表スペース内のフリー・ページ数 : モニター・エレメント

1 つの表スペース内で現在フリーの合計ページ数。

表 1764. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1765. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

これは、DMS 表スペースにのみ適用できます。

tablespace_id - 表スペース ID : モニター・エレメント

現行データベースが使用する表スペースを一意的に示す整数。

表 1766. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMINTEMPTABLES 管理ビューおよび ADMIN_GET_TEMP_TABLES 表関数 - 一時表の情報を取得する	ACTIVITY METRICS BASE
MON_GET_APPL_LOCKWAIT 表関数 - アプリケーションが待機しているロックについての情報の取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1767. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本
表	table	基本

表 1768. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法

このエレメントの値は、SYSCAT.TABLESPACES のビューの TBSPACEID 列の値と一致します。

tablespace_increase_size バイト単位のサイズの増加

このエレメントは、自動サイズ変更表スペースが満杯になって、さらにスペースが必要になったときに、表スペースをどれだけ大きくするかをバイト数で示します。

表 1769. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 これは、自動サイズ変更可能な表スペースが満杯になって、さらにスペースが必要になり、かつ表スペースの最大サイズに達していない場合に、表スペースに追加するスペースの量を示します。このエレメントの値が -1 (またはスナップショット出力で 『AUTOMATIC』) の場合、スペースの追加が必要になったときに DB2 が自動的に値を決定します。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_increase_size_percent パーセント単位のサイズの増加 : モニター・エレメント

このエレメントは、自動サイズ変更表スペースが満杯になって、さらにスペースが必要になったときに、表スペースをどれだけ大きくするかを示します。実際のバイト数は、表スペースがサイズ変更されるときに、そのときの表スペースのサイズに基づいて決まります。

表 1770. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 これは、自動サイズ変更可能な表スペースが満杯になって、さらにスペースが必要になり、かつ表スペースの最大サイズに達していない場合に、表スペースに追加するスペースの量を示します。増加率は、表スペースがサイズ変更されるときの、表スペースの現行サイズ (tablespace_current_size) のパーセンテージに基づいています。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_initial_size 表スペースの初期サイズ

自動ストレージ表スペースの初期サイズ (バイト数)。

表 1771. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 非一時自動ストレージ表スペースの場合、このモニター・エレメントは、表スペースが作成されたときのその表スペースの初期サイズをバイト数で表します。

tablespace_last_resize_failed 失敗した最後のサイズ変更

このエレメントは、表スペースのサイズを自動的に大きくする最後の試みが失敗したかどうかを記述します。値 1 は「はい」を意味し、0 は「いいえ」を意味します。

表 1772. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 自動ストレージ表スペースの場合、このエレメントは、データベースのどのストレージ・パスにもスペースが残っていないことを示している可能性があります。非自動ストレージ表スペースの場合、失敗とは、コンテナーのファイル・システムが満杯だったため、コンテナーのいずれかが拡張できなかったことを意味します。失敗の別の理由は、表スペースの最大サイズに達していることです。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_last_resize_time 最後にサイズ変更が正常に行われた時刻

このエレメントは、表スペースのサイズが正常に大きくなった最後の時刻を表すタイム・スタンプを示します。

表 1773. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 自動的にサイズ変更可能な表スペースの場合、このエレメントは、その表スペースが満杯になって、さらにスペースが必要になり、かつ表スペースの最大サイズに達しておらず、表スペースに自動的にスペースが追加された最後の時刻を表します。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_max_size 表スペースの最大サイズ

このエレメントは、表スペースが自動的にサイズ変更したり、大きくなったりできる最大サイズをバイト数で示します。

表 1774. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1775. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 これは、自動的にサイズ変更可能な表スペースが、どの最大サイズまで自動

的に大きくなれるかをバイト数で示します。この値が tablespace_current_size エレメントと等しい場合、表スペースが大きくなる余地はありません。このエレメントの値が -1 の場合、最大サイズは「無制限」と見なされ、ファイル・システムが満杯になるか、表スペースの体系的なサイズ制限に達するまで、表スペースは自動的にサイズ変更できます (この制限については、「*SQL リファレンス*」の『SQL Limits』の付録で説明されています)。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_min_recovery_time - ロールフォワードの最小リカバリー時間 : モニター・エレメント

表スペースをロールフォワードできる最も早い時点を示すタイム・スタンプ。このタイム・スタンプは、ローカル時間を表します。

表 1776. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1777. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

ゼロ以外のときだけ表示されます。

tablespace_name - 表スペース名 : モニター・エレメント

表スペースの名前。

表 1778. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE
MON_GET_REBALANCE_STATUS - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1779. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本
ロック	appl_lock_list	基本
ロック	lock	ロック
ロック	lock_wait	ロック

表 1780. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
表スペース	tablespace_list	-

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントは、リソースの競合の原因を判別するときに役立ちます。

これはデータベース・カタログ表の SYSCAT.TABLESPACES にある TBSPACE 列と同じです。アプリケーション・レベル、アプリケーション・ロック・レベル、およびデッドロック・モニター・レベルでは、アプリケーションがロックを待機している表スペースの名前です。ほかのアプリケーションがこの表スペースのロックを保留しています。

ロック・レベルでは、アプリケーションがロックを保留している表スペースの名前です。

表スペース・レベルでは (バッファー・プール・モニター・グループが ON の場合)、情報が戻される表スペースの名前です。

このエレメントは、パーティション表で保持されている表ロックの場合は戻されません。

tablespace_next_pool_id - 次の始動時に使用されるバッファー・プール : モニター・エレメント

データベースを次に始動したときに表スペースが使用するバッファー・プールのバッファー・プール ID。

表 1781. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1782. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法 バッファー・プールは、それぞれ固有の整数で識別できます。このエレメントの値は、SYSCAT.BUFFERPOOLS ビューの BUFFERPOOLID 列の値と一致します。

tablespace_num_containers 表スペース内のコンテナー数

表スペース内のコンテナーの合計数。

表 1783. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

tablespace_num_quiescers - 静止プログラム数

表スペースを静止させているユーザーの数 (0 から 5 の範囲)。

表 1784. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 この値は表スペースを静止させたエージェントの数を示します (「共有」、「更新」、または「排他」モード)。それぞれの静止プログラムについて、次の情報が tablespace_quiescer 論理データ・グループに戻されます。

- 静止プログラムのユーザー許可 ID
- 静止プログラムのエージェント ID
- この表スペースが静止することになった、静止されたオブジェクトの表スペース ID
- この表スペースが静止することになった、静止されたオブジェクトのオブジェクト ID
- 静止状態

tablespace_num_ranges 表スペース・マップ内の範囲数

表スペース・マップ内の範囲 (項目) の数。この範囲は 1 から 100 です (ただし通常は 12 未満)。表スペース・マップがあるのは、DMS 表スペースの場合だけです。

表 1785. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

tablespace_page_size - 表スペースのページ・サイズ : モニター・エレメント

表スペースが使用するページ・サイズ (バイト単位)。

表 1786. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1787. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

tablespace_page_top 表スペース最高水準点 : モニター・エレメント

最高水準点を保持している表スペース内のページ。

表 1788. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1789. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

DMS の場合、このエレメントは、表スペースで最後に割り振られたエクステントの次にある最初のフリー・エクステントのページ番号を示します。この値は減少するので、実際の「最高水準点」ではなく「現在の水準点」であることに注意してください。この情報は SMS には適用できません。

tablespace_paths_dropped - ドロップされたパスを使用している表スペース : モニター・エレメント

ドロップされたストレージ・パスを表スペースが使用していることを示します。

表 1790. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1791. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

自動ストレージを使用する表スペースの場合に、このモニター・エレメントを使用して、ドロップされたストレージ・パスに表スペースのコンテナーがあるかどうかを判別します。ストレージ・パスがデータベースから物理的にドロップされる前に、すべての表スペースはその使用を停止しなければなりません。ドロップされたストレージ・パスの使用を停止するには、表スペースをドロップするか、または ALTER TABLESPACE ステートメントの REBALANCE 節を使用して表スペースのリバランスを行います。

tablespace_pending_free_pages 表スペース内のペンディング・フリー・ページ数 : モニター・エレメント

すべてのペンディング・トランザクションがコミットまたはロールバックされ、オブジェクトのための新しいスペースが要求されたときにフリーになる表スペースのページ数。

表 1792. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1793. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

これは、DMS 表スペースにのみ適用できます。

tablespace_prefetch_size - 表スペースのプリフェッチ・サイズ : モニター・エレメント

プリフェッチャーがディスクから 1 回に取得できる最大ページ数。

表 1794. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1795. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本
表スペース	tablespace_nodeinfo	基本

使用法

- 表関数モニターの場合、このエレメントは表スペース・プリフェッチ・サイズの実際の値を常に報告します。
- スナップショット・モニターの場合、自動プリフェッチ・サイズが有効であれば、このエレメントは値「-1」を *tablespace* 論理データ・グループ内に報告し、実際の値が *tablespace_nodeinfo* 論理データ・グループ内に報告されます。
- スナップショット・モニターの場合、自動プリフェッチ・サイズが無効であれば、このエレメントは実際の値を *tablespace* 論理データ・グループ内に報告し、*tablespace_nodeinfo* 論理データ・グループ内にはエレメントが表示されません。

tablespace_rebalancer_extents_processed リバランサーで処理されたエクステントの数

リバランサーの開始後または再始動後 (どちらか後で実行された方) に、リバランサーですでに移動されたエクステントの数。

表 1796. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

表 1797. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーの完了レベルを示す標識として使用できます。このエレメントの内容が時間とともに変化する様子を追跡すると、リバランスの進行状況をモニターできます。 tablespace_state と rebalance_mode を使用すると、リバランスが完了したかどうかチェックできます。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_extents_remaining リバランサーで処理されるエクステントの合計数

移動するエクステントの数。この値は、リバランサーの開始時刻または再始動時刻 (どちらか後に実行された方) の時点で計算されます。

表 1798. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

表 1799. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 このエレメントを、リバランサーの完了レベルを示す標識として使用できます。このエレメントの内容が時間とともに変化する様子を追跡すると、リバランスの進行状況をモニターできます。リバランスが終了したかどうかを確認するには、tablespace_state を使用します。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_last_extent_moved リバランサーによって最後に移動されたエクステント

リバランサーによって最後に移動されたエクステント。

表 1800. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

表 1801. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーの完了レベルを示す標識として使用できます。このエレメントの内容が時間とともに変化する様子を追跡すると、リバランスの進行状況をモニターできます。 tablespace_state と rebalance_mode を使用すると、リバランスが完了したかどうかチェックできます。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_mode - リバランサー・モード : モニター・エレメント

現在のリバランス処理が表スペースからスペースを除去しているのか、あるいは表スペースにスペースを追加しているのかを示します。

表 1802. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1803. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

フォワード・リバランス は、新規コンテナーが追加されたとき、または既存のコンテナーのサイズが大きくなったときに行われます。フォワード・リバランス操作では、データ移動は表スペースの最初のエクステントから始まり、最高水準点のエクステントで終わります。

リバース・リバランス は、コンテナーが除去されたりサイズが縮小されたりする場合に、解放されるスペースからデータを移動する必要があるときに行われます。リバース・リバランス操作では、データ移動は最高水準点のエクステントから始まり、表スペースを逆順に移動していき、表スペースの最初のエクステントで終わります。

2 パス・リバランス は、フォワード・リバランスの後にリバース・リバランスが行われる処理です。リバランス操作の一環としてコンテナーの追加とドロップの両方が行われるときに、2 パス・リバランスが行われることがあります。

DMS 非自動ストレージ表スペースの場合、このモニター・エレメントは、表スペースに対して行われているリバランスのタイプを示します。 DMS 非自動表スペースでは、単一のフォワード・リバランスまたは単一のリバース・リバランスのみを行えます。

自動ストレージ表スペースの場合、このモニター・エレメントは、現在のリバランス処理が表スペースに対して何を行っているかを示します。一般に、リバランスが開始されると、必要なのは単一のフォワード・リバランスまたは単一のリバース・リバランスのみです。ただし、自動ストレージ表スペースの場合は、2 パス・リバランスが必要な場合があります。

tablespace_rebalancer_mode に可能な値は、sqlmon.h ファイルで定義されています。次の値が、スナップショット・モニターで戻ります。

SQLM_TABLESPACE_NO_REBAL
リバランスは行われていません。

SQLM_TABLESPACE_FWD_REBAL
フォワード・リバランスが行われていす。

SQLM_TABLESPACE_REV_REBAL
リバース・リバランスが行われています。

SQLM_TABLESPACE_FWD_REBAL_OF_2PASS
2 パス・リバランスのうちのフォワード・リバランス・フェーズが行われています。

SQLM_TABLESPACE_REV_REBAL_OF_2PASS
2 パス・リバランスのうちのリバース・リバランス・フェーズが行われています。

MON_GET_TABLESPACE 表関数または MON_GET_REBALANCE_STATUS 表関数のいずれかを使用している場合、以下の rebalancer_mode 値が戻ります。

- NO_REBAL
- FWD_REBAL
- REV_REBAL
- FWD_REBAL_OF_2PASS
- REV_REBAL_OF_2PASS

tablespace_rebalancer_priority 現行のリバランサー優先順位

データベース内で実行されているリバランサーの優先順位。

表 1804. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

表 1805. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_restart_time リバランサー再始動時刻

リバランサーが一時停止または中断された後に再始動されたときを示すタイム・スタンプ。

表 1806. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

表 1807. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーの完了レベルを示す標識として使用できます。リバランサーが再始動されたときを示し、リバランサーの速度を導出できるので、推定完了時刻を得られます。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_source_storage_group_id - リバランサー・ソース・ストレージ・グループ ID

リバランサーがストレージ・グループ間で表スペースを移動する際のソース・ストレージ・グループ ID。移動していない場合には、-1 です。

表 1808. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

tablespace_rebalancer_source_storage_group_name - リバランサー・ソース・ストレージ・グループ名

リバランサーがストレージ・グループ間で表スペースを移動する際のソース・ストレージ・グループ名。移動していない場合には、NULL です。

表 1809. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

tablespace_rebalancer_start_time リバランサー開始時刻

リバランサーを最初に開始した時刻を示すタイム・スタンプ。

表 1810. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

表 1811. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーを最初に開始した時刻を知るのに使用します。これは、リバランサーの処理速度を測定したり、リバランサーの終了時刻を推定するときに使用できます。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_status - リバランサー状況 : モニター・エレメント

リバランス操作の現行状況を示します。

表 1812. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

使用上の注意

リバランス操作の現行状況は以下のいずれかになります。

- ACTIVE - リバランス操作はアクティブです。
- SUSPENDED - ユーザーが ALTER TABLESPACE ステートメントを使用して、明示的にリバランス操作を中断しました。
- PAUSED - オンライン・バックアップのために、リバランス操作が暗黙のうちに一時停止しました。バックアップが完了するとリバランスが続行されます。

リバランス操作が明示的に中断され、暗黙のうちに一時停止すると、状況は SUSPENDED とレポートされます。

tablespace_rebalancer_target_storage_group_id - リバランサー・ターゲット・ストレージ・グループ ID

リバランサーがストレージ・グループ間で表スペースを移動する際のターゲット・ストレージ・グループ ID。移動していない場合には、-1 です。

表 1813. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

このモニター・エレメントは、target_storage_group_name モニター・エレメントと source_storage_group_id モニター・エレメントおよび source_storage_group_name モニター・エレメントに関連があります。これらのエレメントを一緒に使用すると、リバランス操作によって表スペースがストレージ・グループ間を移動するかどうかを調べることができます。また、表スペースの移動元 (ソース) のストレージ・グループおよび表スペースの移動先 (ターゲット) のストレージ・グループを調べることができます。

tablespace_rebalancer_target_storage_group_name - リバランサー・ターゲット・ストレージ・グループ名

リバランサーがストレージ・グループ間で表スペースを移動する際のターゲット・ストレージ・グループ名。移動していない場合には、NULL です。

表 1814. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE

tablespace_state - 表スペースの状態 : モニター・エレメント

このエレメントは、表スペースの現在の状態を示します。

表 1815. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1816. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

管理ビューと表関数では、このモニター・エレメントは `sqlutil.h` での定義に基づいたテキスト ID を返します。これは以下の値を「+」記号で区切って組み合わせたものです。

- BACKUP_IN_PROGRESS
- BACKUP_PENDING
- DELETE_PENDING
- DISABLE_PENDING
- DROP_PENDING
- LOAD_IN_PROGRESS
- LOAD_PENDING
- MOVE_IN_PROGRESS
- NORMAL
- OFFLINE
- PSTAT_CREATION
- PSTAT_DELETION
- QUIESCED_EXCLUSIVE
- QUIESCED_SHARE
- QUIESCED_UPDATE

- REBAL_IN_PROGRESS
- REDIST_IN_PROGRESS
- REORG_IN_PROGRESS
- RESTORE_IN_PROGRESS
- RESTORE_PENDING
- ROLLFORWARD_IN_PROGRESS
- ROLLFORWARD_PENDING
- STORDEF_ALLOWED
- STORDEF_CHANGED
- STORDEF_FINAL_VERSION
- STORDEF_PENDING
- SUSPEND_WRITE

このエレメントには、表スペースの現在の状態を示す 16 進値が含まれています。外部から見ることができる表スペースの状態は、特定の状態値が 16 進数の合計値で構成されています。例えば、状態が "quiesced: EXCLUSIVE" かつ "Load pending" の場合、その値は 0x0004 + 0x0008、つまり 0x000c となります。 **db2tbst** コマンドを使用して、特定の 16 進値と関連した表スペース状態を取得します。

表 1817. sqlutil.h 中にリストされているビット定義

16 進値	**10 進値**	**状態**
0x0	0	標準 (sqlutil.h 内の SQLB_NORMAL の定義を参照)
0x1	1	静止モードでの共有
0x2	2	静止モードでの更新
0x4	4	静止モードでの排他
0x8	8	ロード・ペンディング
0x10	16	削除ペンディング
0x20	32	バックアップ・ペンディング
0x40	64	ロールフォワード進行中
0x80	128	ロールフォワード・ペンディング
0x100	256	リストア・ペンディング
0x100	256	リカバリー・ペンディング (未使用)
0x200	512	使用不可ペンディング
0x400	1024	REORG 進行中
0x800	2048	バックアップ進行中
0x1000	4096	ストレージを定義する必要がある
0x2000	8192	リストア進行中
0x4000	16384	オフラインのためアクセス不可
0x8000	32768	ドロップ・ペンディング
0x10000	65536	書き込み不許可
0x20000	131072	ロード進行中
0x40000	262144	再配分進行中

表 1817. sqlutil.h 中にリストされているビット定義 (続き)

16 進値	10 進値	状態
0x80000	524288	移動進行中
0x2000000	33554432	ストレージを定義可能
0x4000000	67108864	ストレージ定義は最終状態
0x8000000	134217728	ストレージ定義はロールフォワードの前に変更された
0x10000000	268435456	DMS リバランサー進行中
0x20000000	536870912	表スペース削除進行中
0x40000000	1073741824	表スペース作成進行中

注: DB2 LOAD は、表スペース状態を `Load pending` および `Delete pending` に設定しません。

tablespace_state_change_object_id 状態変更オブジェクト ID

表スペースの状態を「ロード・ペンディング」または「削除ペンディング」に設定したオブジェクト。

エレメント ID

tablespace_state_change_object_id

エレメント・タイプ

情報

表 1818. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 このエレメントに意味があるのは、表スペースの状態が「ロード・ペンディング」または「削除ペンディング」の場合だけです。このエレメントの値がゼロ以外の場合は、SYSCAT.TABLES ビューの TABLEID 列の値と一致します。

注: DB2 LOAD は、表スペース状態を `Load pending` および `Delete pending` に設定しません。

tablespace_state_change_ts_id 状態変更表スペース ID

表スペースの状態が「ロード・ペンディング」または「削除ペンディング」の場合は、表スペースの状態の設定を引き起こしたオブジェクトの表スペース ID が示されます。

エレメント ID

tablespace_state_change_ts_id

エレメント・タイプ

情報

表 1819. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 このエレメントに意味があるのは、表スペースの状態が「ロード・ペンディング」または「削除ペンディング」の場合だけです。このエレメントの値がゼロ以外の場合は、 SYSCAT.TABLES ビューの TABLESPACEID 列の値と一致します。

注: DB2 LOAD は、表スペース状態を `Load pending` および `Delete pending` に設定しません。

tablespace_total_pages 表スペース内の合計ページ数 : モニター・エレメント

1 つの表スペース内の合計ページ数。

表 1820. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1821. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法

1 つの表スペースが使用するオペレーティング・システムの合計スペースです。DMS の場合、これは、コンテナー・サイズの合計です。SMS の場合は、この表スペースに保管される表に使用されるすべてのファイル・スペースの合計です (この情報は、バッファー・プール・スイッチが ON の場合にのみ収集されます)。

tablespace_type - 表スペース・タイプ : モニター・エレメント

表スペースのタイプ。

表 1822. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1823. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法

このエレメントは、この表スペースがデータベース管理の表スペース (DMS) か、システム管理の表スペース (SMS) かを示します。

tablespace_type の値 (sqlmon.h 内に定義) は次のとおりです。

- DMS の場合: SQLM_TABLESPACE_TYP_DMS
- SMS の場合: SQLM_TABLESPACE_TYP_SMS

tablespace_usable_pages 表スペース内の使用可能ページ数 : モニター・エレメント

表スペース内の合計ページ数からオーバーヘッド・ページ数を引いた数値。

表 1824. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1825. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法

このエレメントが適用されるのは DMS 表スペースだけです。 SMS 表スペースの場合は、このエレメントの値は **tablespace_total_pages** モニター・エレメントと同じになります。

表スペースのリバランス中に、使用可能ページ数に新たに追加されたコンテナーが加えられますが、これらの新しいページ数は、リバランス完了までの間フリー・ページ数には反映されません。表スペースのリバランスが実行されていない場合は、使用済みページ数、フリー・ページ数、およびペンディング・フリー・ページ数の合計が使用可能ページ数に等しくなります。

tablespace_used_pages 表スペース内の使用されているページ数 : モニター・エレメント

表スペース内で現在使用中 (フリーではない) の合計ページ数。

表 1826. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1827. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法

DMS 表スペースで使用中の合計ページ数です。 SMS 表スペースの場合は、**tablespace_total_pages** モニター・エレメントと同じ値になります。

tablespace_using_auto_storage - 自動ストレージが使用可能な表スペース : モニター・エレメント

このエレメントは、表スペースが自動ストレージ表スペースとして作成されたかどうかを記述します。値 1 は「はい」を意味し、値 0 は「いいえ」を意味します。

表 1828. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 1829. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法

このエレメントを使用して、指定された表スペースが、明示的に提供されているコンテナーを使用するのではなく、自動ストレージを使用して作成されたかどうか (つまり MANAGED BY AUTOMATIC STORAGE 節を指定して作成されているかどうか) を判別できます。表スペースは、データベースに関連している一部の (または全部の) ストレージ・パスに存在するコンテナーを持つことができます。

target_cf_gbp_size - ターゲット・クラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント

このモニター・エレメントは、動的サイズ変更の際に、グループ・バッファー・プール・メモリーのターゲット値をページ・サイズ 4 KB のページ単位で表示します。ターゲット値が構成値と一致すると、サイズ変更が完了します。

表 1830. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

target_cf_lock_size - ターゲット・クラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント

このモニター・エレメントは、動的サイズ変更の際に、グローバル・ロック・メモリーのターゲット値をページ・サイズ 4 KB のページ単位で表示します。ターゲット値が構成値と一致すると、サイズ変更が完了します。

表 1831. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

target_cf_sca_size - ターゲット・クラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント

このモニター・エレメントは、動的サイズ変更の際に、共用通信域メモリーのターゲット値をページ・サイズ 4 KB のページ単位で表示します。ターゲット値が構成値と一致すると、サイズ変更が完了します。

表 1832. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

tbsp_datatag - 表スペース・データ・タグ

このエレメントは、表スペースの実際のデータ・タグ値を示します。実際のデータ・タグ値とは、明示的に表スペースに指定されたデータ・タグ値、または表スペース・ストレージ・グループから継承されたデータ・タグ値です。

ユーザーが指定できる有効な範囲は、1 から 0 までです。0 は、データ・タグが指定されていないことを表します。

表 1833. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

使用上の注意

データ・タグは、 WLM 構成内の参照可能なデータを識別してグループ化するために使用されます。WLM 構成によって、ユーザーの作業の処理優先度に影響を与える可能性がある、タグ付けの効果が決まります。

tbsp_last_consec_page - オブジェクト表の連続ページの最後のページのモニター・エレメント

表スペースの連続メタデータ・ページの最後のページのオブジェクト相対ページ番号。この値は、DMS 表スペースの場合のみ有効です。そうでない場合は 0 です。

表 1834. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

tbsp_max_page_top - 最大表スペース・ページの最高水準点 : モニター・エレメント

データベースが活動化された以降に DMS 表スペースに割り振られた最高ページ番号。

表 1835. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

使用法

この値は、**tablespace_page_top** モニター・エレメントの値が増えるときにはいつでも変更されます。

tbsp_names - 表スペース名

このエレメントは、ユーティリティーの実行対象である表スペース名をリストします。

表 1836. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILSTART	常に収集される

使用法

変更履歴イベント・モニターでは、object_type エレメントが DATABASE または TABLESPACE である場合、これはユーティリティーの実行対象である表スペース名のコンマ区切りリストです。

tbsp_trackmod_state - 表スペースの trackmod 状態 : モニター・エレメント

前回または次回のバックアップに関連した表スペースの変更状態。

表 1837. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	常に行う

使用法

このモニター・エレメントを使用して、表スペースの変更状況を判別できます。表スペースの状況は、次のいずれかになります。

CLEAN
前回のバックアップ以降、表スペースに変更は加えられていません。この時点で増分バックアップか差分バックアップが実行される場合、この表スペースからバックアップされるデータ・ページはありません。

DIRTY
表スペースには、次回のバックアップ時に選出される必要があるデータが含まれています。

ININCREMENTAL
表スペースには、増分バックアップ内にコピーされた変更が含まれています。この状態は、フルバックアップと比較して、将来の増分バックアップ時にこのプールからページを含める必要があるという点では DIRTY 状態といえます。また、将来の差分バックアップ時にこのプールからページを含める必要がないという点では CLEAN 状態でもあります。

READFULL
表スペースの変更状態が前回変わった原因は、正常に完了しなかった可能性があるフルバックアップか、現在進行中のフルバックアップで読み取られているダーティー表スペースにあります。

READINCREMENTAL
表スペースの変更状態が前回変わった原因は、正常に完了しなかった可能性がある増分バックアップか、現在進行中の増分バックアップで読み取られているダーティー表スペースにあります。

UNAVAILABLE
trackmod 構成パラメーターが No に設定されています。したがって、表スペースの変更状況の情報は利用できません。

tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント

データ・サーバーがクライアントから TCP/IP 経由で受信したデータの量。この値はバイト単位で示されます。

表 1838. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1839. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント

TCP/IP 経由での着信クライアント要求の待機にかかった時間。アイドル時間は除きます。値はミリ秒単位で示されます。

表 1840. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY _ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1841. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント

データベース・サーバーがクライアント・アプリケーションから TCP/IP 経由でデータを受信した回数。

表 1842. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1843. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント

データ・サーバーによってクライアントに送信されたデータの量。この値はバイト単位で示されます。

表 1844. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1845. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント

クライアントへの TCP/IP 送信のブロックにかかった時間。値はミリ秒単位で示されます。

表 1846. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY _ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1847. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント

データベース・サーバーからクライアント・アプリケーションへ TCP/IP 経由でデータが送信された回数。

表 1848. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1849. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	常に収集される

temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント

temp_tablespace_top モニター・エレメントは、サービス・クラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの TEMPORARY 表スペース使用量 (KB 単位) の最高水準点です。

サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了したサービス・サブクラスの temp_tablespace_top 最高水準点のみが変更されます。 アクティビティーがマップされたサービス・サブクラスでも、アクティビティーがそこで完了しなかった場合、そのサービス・サブクラスの最高水準点は何も影響を受けません。

表 1850. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーで到達した DML アクティビティーの SYSTEM TEMPORARY 表スペース使用量の最大値を判別することができます。

このエレメントは、適用される TEMPORARY 表スペースのしきい値があるアクティビティーによってのみ更新されます。アクティビティーに TEMPORARY 表スペースのしきい値が適用されない場合、0 の値が返されます。

territory_code データベース・テリトリー・コード

モニター・データが収集されるデータベースのテリトリー・コード。このモニター・エレメントは以前は country_code と呼んでいました。

表 1851. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本

表 1852. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	常に収集される
接続	event_connheader	常に収集される

使用法 テリトリー・コード情報は、データベース構成ファイルに記録されています。

DRDA AS 接続の場合、このエレメントは 0 に設定されます。

thresh_violations - しきい値違反の回数 : モニター・エレメント

しきい値が違反された回数。

このモニター・エレメントは、スナップショット・モニター・ルーチンおよびデータベース・イベント・モニターによって戻される、 1221 ページの『num_threshold_violations しきい値違反の回数 : モニター・エレメント』 モニター・エレメントの別名です。

表 1853. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 1853. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1854. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このエレメントを使用して、違反された WLM しきい値があるかどうかを迅速に判別できます。しきい値が違反された場合、しきい値の違反イベント・モニター (すでに作成済みでアクティブな場合) を使用してしきい値違反に関する詳細を入手できます。

一例として、どのしきい値が違反されたのか詳細を入手できます。

threshold_action しきい値アクション : モニター・エレメント

このしきい値違反レコードが適用されるしきい値のアクション。可能な値は「停止」、「続行」および「再マップ」です。

表 1855. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値を違反したアクティビティーが、違反が起きた時点で停止したか、実行を継続できたか、それとも他のサービス・サブクラスに再マップされたかを判別できます。 アクティビティーが停止された場合は、そのアクティビティーをサブミットしたアプリケーションは SQL4712N エラーを受け取ることになります。アクティビティーが他のサービス・サブクラスに再マップされた場合、メンバー上でアクティビティー用に作動しているエージェントはしきい値のターゲット・サービス・サブクラスに移動します。

threshold_domain しきい値ドメイン : モニター・エレメント

このキューに関係するしきい値のドメイン。

可能な値は以下のとおりです。

- データベース
- 作業アクションセット
- サービス・スーパークラス
- サービス・サブクラス
- ワークロード

表 1856. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 1857. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、述部は同じでもドメインが異なるしきい値のキュー統計を区別することができます。

threshold_maxvalue しきい値最大値 : モニター・エレメント

このモニター・エレメントは、キューイング非対象しきい値においては、このしきい値を超えてしまった値を表します。キューのしきい値の場合は、このモニター・エレメントは、キューイングの原因となった並行性のレベルを表します。

キューのしきい値の違反の原因となった並行性のレベルは、**threshold_maxvalue** および **threshold_queuesize** モニター・エレメントの合計です。

表 1858. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	常に収集される

使用法

アクティビティーしきい値では、このエレメントは、しきい値の違反が発生した時点でのしきい値の最大値の履歴レコードを提供します。これは、違反の発生以降にしきい値の最大値が変更され、古い値が SYSCAT.THRESHOLDS ビューで表示できなくなった場合に便利です。DATATAGINSC IN および DATATAGINSC NOT IN しきい値の場合、このエレメントには、しきい値に違反したデータ・タグ値が入っています。

threshold_name しきい値名 : モニター・エレメント

このキューに関係するしきい値の固有の名前。

表 1859. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 1860. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、このレコードが示す統計の元となるキューのしきい値を一意的に識別できます。

threshold_predicate しきい値述部 : モニター・エレメント

違反したしきい値または統計の収集の対象となったしきい値のタイプを識別します。

表 1861. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 1862. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	常に収集される
統計	event_qstats	常に収集される

使用法

このモニター・エレメントを他の統計またはしきい値違反モニター・エレメントと一緒に使用すると、しきい値違反の分析をすることができます。

event_thresholdviolations 論理グループでレポートされる場合のこのモニター・エレメントの有効な値は、次のとおりです。

AggSQLTempSpace

SQLTempSpace

SQLRowsReturned

ActivityTotalTime

EstimatedSQLCost

TotalMemberConnections

ConnectionIdleTime

ConcurrentWorkloadOccurrences

ConcurrentWorkloadActivities

ConcurrentDBCoordActivities

TotalSCMemberConnections

SQLRowsRead

SQLRowsReadInSC

CPUTime

CPUTimeInSC

UowTotalTime

DataTagInSC

DataTagNotInSC

event_qstats 論理グループでレポートされる場合のこのモニター・エレメントの有効な値は、次のとおりです。

TotalMemberConnections

ConcurrentDBCoordActivities

TotalSCMemberConnections

threshold_queuesize しきい値キュー・サイズ : モニター・エレメント

キューのしきい値におけるキューのサイズ。このサイズを超えようとすると、しきい値違反が発生します。キューのしきい値以外では、この値は 0 です。

表 1863. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値の違反が発生した時点でのこのしきい値のキューにおけるアクティビティーまたは接続の数を判別できます。

thresholdid しきい値 ID : モニター・エレメント

しきい値違反レコードを適用するしきい値か、キュー統計の収集対象のしきい値を識別します。

表 1864. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 1865. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-
統計	event_qstats	-

使用法

このモニター・エレメントを他のアクティビティー履歴モニター・エレメントと一緒に使用すると、しきい値キューの分析またはしきい値に違反したアクティビティーの分析をすることができます。

time_completed 完了時刻 : モニター・エレメント

このアクティビティー・レコードにより記述されているアクティビティーが実行を完了した時刻。このエレメントは、ローカル・タイム・スタンプです。

表 1866. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

メモリーの制約のためにアクティビティー・レコード全体を表イベント・モニターに書き込むことができなかった場合、このフィールドの値は「0000-00-00-00.00.00.000000」になります。進行中のアクティビティーがキャプチャーされた場合、このフィールドは、アクティビティーが収集された時間を表します。

time_created 作成時刻 : モニター・エレメント

ユーザーが、このアクティビティー・レコードにより記述されているアクティビティーをサブミットした時刻。このエレメントは、ローカル・タイム・スタンプです。

表 1867. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

time_of_violation 違反時刻 : モニター・エレメント

このしきい値違反レコードに記述されているしきい値違反が発生した時刻。このエレメントは、ローカル・タイム・スタンプです。

表 1868. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを他のしきい値違反モニター・エレメントと一緒に使用すると、しきい値違反の分析をすることができます。

time_stamp スナップショット時刻

データベース・システム・モニター情報が収集された日時。

表 1869. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

使用法 このエレメントを使用すると、継続的な分析作業でその結果をファイルやデータベースに保管してある場合は、データを時系列的に関連付けることができます。

time_started 開始時刻 : モニター・エレメント

このアクティビティー・レコードにより記述されているアクティビティーが実行を開始した時刻。このエレメントは、ローカル・タイム・スタンプです。

表 1870. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

アクティビティーが拒否された場合、**act_exec_time** モニター・エレメントの値は 0 です。この場合、**time_started** モニター・エレメントの値は **time_completed** モニター・エレメントの値に等しくなります。

time_zone_disp 時間帯変位

グリニッジ標準時 (GMT) と現地時間帯の差を示す秒数。

表 1871. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

使用法 データベース・システム・モニターから報告される時刻はすべて GMT であり、現地時間はこの差に基づいて計算されます。

top ヒストグラム・ビンの最上位 : モニター・エレメント

ヒストグラム・ビンの範囲の包括的最上端。このモニター・エレメントの値は、次のヒストグラム・ビンの範囲の排他的最下端でもあります。

表 1872. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントと対応する **bottom** エレメントを一緒に使用して、ヒストグラム中のビンの範囲を判別します。

tot_log_used_top 使用された最大合計ログ・スペース

使用された全ログ・スペースの最大量 (バイト単位)。

表 1873. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1874. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 1875. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントは、ユーザーが割り振った 1 次ログ・スペースの量を評価するときに利用できます。このエレメントの値と割り振った 1 次ログ・スペースの量を比較すると、構成パラメーターの設定値を評価するときに利用できます。割り振った 1 次ログ・スペースは、次の公式で計算できます。

```
logprimary x logfilsiz x 4096 (下記の注を参照)
```

このエレメントと *sec_log_used_top* および *sec_logs_allocated* を組み合わせて使用すると、 2 次ログの依存度を示すことができます。

この値には、1 次と 2 次の両方のログ・ファイルに使用されるスペースが含まれます。

次の構成パラメーターの調整が必要になります。

- logfilsiz
- logprimary
- logsecond

注: データベース・システム・モニター情報はバイト単位で示されますが、構成パラメーターは各 4K バイトのページ単位で設定されます。

total_act_time - 合計アクティビティー時間 : モニター・エレメント

アクティビティーの実行にかかった時間の合計。この値はミリ秒単位で示されます。

表 1876. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1877. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このモニター・エレメントを **total_act_wait_time** モニター・エレメントと一緒に使用して、データ・サーバーがアクティビティーの処理にかかった時間のパーセンテージを判別します。

```
(total_act_time - total_act_wait_time)/(total_act_time) =
 % of time data server is actively working on activity
```

total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント

アクティビティーの処理中に、DB2 データベース・サーバー内での待機にかかった合計時間。値はミリ秒単位で示されます。

表 1878. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY _ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1878. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1879. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このモニター・エレメントを **total_act_time** モニター・エレメントと一緒に使用して、データ・サーバーがアクティビティーの処理にかかった時間のパーセンテージを判別します。

```
(total_act_time - total_act_wait_time)/(total_act_time) =
 % of time data server is actively working on activity
```

total_app_commits - アプリケーションのコミットの合計数 : モニター・エレメント

クライアント・アプリケーションが発行した commit ステートメントの合計数。

表 1880. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1881. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 1881. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_app_rollbacks - アプリケーションの合計ロールバック数 : モニター・エレメント

クライアント・アプリケーションが発行した rollback ステートメントの合計数。

表 1882. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1883. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント

アプリケーション要求のためにかかった経過時間の合計。これは、アプリケーション要求を実行するサーバー上でコーディネーター・エージェントが費やした合計時間です。この値はミリ秒単位で報告されます。

表 1884. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1884. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントを使用して、DB2 データ・サーバーでアプリケーション要求が費やした時間を判別します。この値は、観察されたパフォーマンス上の問題の原因がデータ・サーバーであるかどうかを判別するために利用できます。

例えば、アプリケーションで問題が生じ、回復するのに 20 分かかるとユーザーが報告する場合、合計アプリケーション要求時間が 1 分と判別され、その接続について現在進行中のアプリケーション要求がないのであれば、そのパフォーマンス上の問題は DB2 データ・サーバー以外のところにある可能性があります。

total_app_section_executions - アプリケーションのセクション実行数の合計 : モニター・エレメント

アプリケーションによって実行されたセクション実行の数。

表 1885. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1885. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1886. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_buffers_rcvd 受信された FCM バッファーの合計

スナップショット・モニターの場合、このモニター・エレメントは、**node_number** モニター・エレメントで識別されるノードから送信されて GET SNAPSHOT コマンドを発行するノードで受信された FCM バッファーの総数を報告します。表関数モニターの場合、このモニター・エレメントは、リモート・データベース・メンバーから受信した FCM バッファーの総数を報告します。

表 1887. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE

表 1888. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm_node	基本

使用法

このエレメントを使用して、現行メンバーとリモート・メンバーの間のトラフィック・レベルを測定できます。このメンバーから受信した FCM バッファーの総数が多い場合は、データベースを再配分するか表を移動してメンバー間のトラフィックを減らすことを検討してください。

total_buffers_sent 送信された FCM バッファーの合計

スナップショット・モニターの場合、このモニター・エレメントは、GET SNAPSHOT コマンドを発行するノードから **node_number** モニター・エレメントで識別されるノードに送信された FCM バッファーの総数を報告します。表関数モニターの場合、このモニター・エレメントは、現行データベース・メンバーからリモート・データベース・メンバーに送信された FCM バッファーの総数を報告します。

表 1889. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続の詳細の取得	ACTIVITY METRICS BASE

表 1890. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm_node	基本

使用法

このエレメントを使用して、現行メンバーとリモート・メンバーの間のトラフィック・レベルを測定できます。このメンバーに送信された FCM バッファーの総数が多い場合は、データベースを再配分するか表を移動してメンバー間のトラフィックを減らすことを検討してください。

total_bytes_received - 受信バイト数のモニター・エレメント

ネットワーク・アダプターの開始以降に受信した合計バイト数。

表 1891. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE

total_bytes_sent - 送信バイト数のモニター・エレメント

ネットワーク・アダプターの開始以降に送信した合計バイト数。

表 1892. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE

total_commit_proc_time - コミット処理時間の合計 : モニター・エレメント

データベース・サーバー上でコミット処理を実行するのにかかった処理時間 (待機ではない) の合計。値はミリ秒単位で示されます。

表 1893. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 1893. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1894. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_commit_time - 合計コミット時間 : モニター・エレメント

データベース・サーバー上でコミット処理を実行するのにかかった時間の合計。値はミリ秒単位で示されます。

表 1895. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 1895. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1896. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_compilations - 合計コンパイル数 : モニター・エレメント

データベース・サーバーでの明示的なコンパイル数の合計。明示的なコンパイルとは、バインド、再バインド、準備、または即時に実行などのユーザー要求によって直接開始されるコンパイルのことです。

表 1897. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1898. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 1898. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_compile_proc_time - コンパイル処理時間の合計 : モニター・エレメント

データベース・サーバー上で明示的なコンパイルを実行するのにかかった処理時間 (待機ではない) の合計。明示的なコンパイルとは、バインド、再バインド、準備、または即時に実行などのユーザー要求によって直接開始されるコンパイルのことです。値はミリ秒単位で示されます。

表 1899. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1899. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1900. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_compile_time - 合計コンパイル時間 : モニター・エレメント

データベース・サーバー上で明示的なコンパイルを実行するのにかかった時間の合計。明示的なコンパイルとは、バインド、再バインド、準備、または即時に実行などのユーザー要求によって直接開始されるコンパイルのことです。値はミリ秒単位で示されます。

表 1901. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 1901. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1902. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_cons データベース活動化以降の接続

最初の接続、活動化、または最後のリセット (コーディネーター・エージェント) 以降のデータベースへの接続数を示します。

表 1903. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1904. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法 このエレメントと db_conn_time および db2start_time モニター・エレメントを組み合わせて使用すると、アプリケーションがデータベースに接続する頻度を計算できます。

接続の頻度が少ない場合は、他のアプリケーションに接続する前に、ACTIVATE DATABASE コマンドを使用して、データベースを活動化することもできます。これは、初めてデータベースに接続する際に時折生じる処理時間の増加 (初期バッファー・プール割り振りなど) のためです。こうすると、それ以後の接続処理の速度を上げることができます。

注: このエレメントをリセットすると、値はゼロではなく、現在接続中のアプリケーションの数に設定されます。

total_connect_authentication_proc_time - 接続認証の合計処理時間のモニター・エレメント

接続またはユーザー切り替えの認証の実行に費やした (待機以外の) 処理時間 (ミリ秒)

表 1905. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1905. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1906. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_connect_authentications - 実行された接続またはユーザー切り替え認証のモニター・エレメント

実行された接続またはユーザー切り替え認証の数。

表 1907. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1907. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1908. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

total_connect_authentication_time - 接続またはユーザー切り替えの認証要求の合計時間のモニター・エレメント

接続またはユーザー切り替えの認証の実行に費やした時間 (ミリ秒)

表 1909. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1909. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1910. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

total_connect_request_proc_time - 接続要求またはユーザー切り替え要求の合計処理時間のモニター・エレメント

接続要求またはユーザー切り替え要求の処理に費やした (待機以外の) 処理時間 (ミリ秒)

表 1911. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 1911. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1912. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

total_connect_requests - 接続要求またはユーザー切り替え要求のモニター・エレメント

接続要求またはユーザー切り替え要求の合計数。

表 1913. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 1913. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1914. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

total_connect_request_time - 接続要求またはユーザー切り替え要求の合計時間のモニター・エレメント

接続要求またはユーザー切り替え要求の実行に費やした時間 (ミリ秒)

表 1915. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1915. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1916. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

total_cpu_time - 合計 CPU 時間 : モニター・エレメント

DB2 内で使用された CPU 時間の合計。ユーザーおよびシステム両方の CPU 時間の合計を表します。この値はマイクロ秒単位で示されます。

WLM_GET_SERVICE_SUBCLASS_STATS または WLM_GET_WORKLOAD_STATS 表関数によって戻される場合、統計が最後にリセットされて以降の合計 CPU 時間を表します。 MON_SAMPLE_SERVICE_CLASS_METRICS または MON_SAMPLE_WORKLOAD_METRICS 表関数によって戻される場合、関数が実行されて以降の合計 CPU 時間を表します。

MON_GET_ROUTINE 表関数または MON_GET_ROUTINE_DETAILS 表関数で戻される場合、このエレメントは、このルーチンの現行メンバーのエージェントとサブエージェントで費やされた合計 CPU 時間を表します。 fenced プロセスで費やされた CPU 時間は含まれません。

表 1917. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1917. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	REQUEST METRICS BASE

表 1917. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 1918. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

total_cpu_time を、**total_disp_run_queue_time** モニター・エレメントと一緒に使用すると、CPU リソースの競合度合いを計算できます。その際、スケール 0 から 1 の範囲で測定し、数値が小さいほど、CPU リソースの競合が大きいことを意味します。CPU 速度と呼ばれるこの指標は、CPU にアクセスしてサービス・クラスで作業していた時間を、CPU にアクセスしていた、または CPU にアクセスするために待機していた合計時間で除算して算出します。CPU 速度は、サービス・クラスで実行されている作業が、CPU をまったく待機しないで実行できる場合に比べて、どれほど効率的に実行されているかを把握する手段となります。数式は、次のとおりです。

```
CPU velocity = total_cpu_time / (total_cpu_time + total_disp_run_queue_time)
```

total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント

このサービス・クラスで実行された要求が CPU にアクセスするために待機していた合計時間。この値はマイクロ秒単位で示されます。

表 1919. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 1919. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 1920. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

total_disp_run_queue_time モニター・エレメントを、**total_cpu_time** と一緒に使用すると、CPU リソースの競合度合いを計算できます。その際、スケール 0 から 1 の範囲で測定し、数値が小さいほど、CPU リソースの競合が大きいことを意味します。CPU 速度と呼ばれるこの指標は、CPU にアクセスしてサービス・クラスで作業していた時間を、CPU にアクセスしていた、または CPU にアクセスするために待機していた合計時間で除算して算出します。この指標によって、作業が CPU をまったく待機しないで実行された場合の効率に比べ、どれほど効率的に実行されているかを測定することができます。数式は、次のとおりです。

```
CPU velocity = total_cpu_time / (total_cpu_time + total_disp_run_queue_time)
```

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降のディスパッチャー実行キューの合計待機時間を表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降のディスパッチャー実行キューの合計待機時間を表します。

total_exec_time ステートメント実行の経過時間 : モニター・エレメント

SQL キャッシュ内の特定のステートメントの実行に要した合計時間 (秒およびマイクロ秒単位)。

表 1921. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを **num_executions** モニター・エレメントとともに使用して、ステートメントの平均経過時間を判別し、 SQL の調整によって最も望ましい影響を受ける SQL ステートメントを識別します。このエレメントの内容を評価する際は、**num_compilation** モニター・エレメントも考慮する必要があります。

注: DB2 システムが統計を収集する際の細分度の違いのために、**total_exec_time** モニター・エレメントの値が、**system_cpu_time** モニター・エレメントの値と **user_cpu_time** モニター・エレメントの値の合計と等しくならないことがあります。この場合、**system_cpu_time** モニター・エレメントと **user_cpu_time** モニター・エレメントの合計の方が実際の実行時間の合計を正確に反映しています。

total_extended_latch_wait_time - 拡張ラッチの合計待機時間のモニター・エレメント

拡張ラッチの待機に費やされた時間 (ミリ秒)。

表 1922. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: 入力として与えられた XML 文書内に含まれているエレメントをすべて報告する。
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: 入力として与えられた XML 文書内に含まれているエレメントをすべて報告する。
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: 入力として与えられた XML 文書内に含まれているエレメントをすべて報告する。

表 1922. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_EXTENDED_LATCH_WAITS	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1923. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE

表 1923. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE

使用法

- 拡張ラッチの待機時間を、合計待機時間に対するパーセンテージとして表すには、次の数式を使用します。この数式を使用すると、拡張ラッチの待機に費やした時間が、合計待機時間と比較して相対的に高いかどうかを判別できます。

  ```
  (TOTAL_EXTENDED_LATCH_WAIT_TIME / TOTAL_WAIT_TIME) * 100
  ```

- 次の数式を使用して、拡張ラッチの平均待機時間 (ミリ秒) を計算できます。

  ```
  TOTAL_EXTENDED_LATCH_WAIT_TIME / TOTAL_EXTENDED_LATCH_WAITS
  ```

total_extended_latch_waits - 拡張ラッチの合計待機回数のモニター・エレメント

拡張ラッチの待機回数。

表 1924. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1924. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_EXTENDED_LATCH_WAITS	REQUEST METRICS BASE

表 1925. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	uow (metrics.xml 文書に報告されます) uow_metrics	REQUEST METRICS BASE

表 1925. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	pkgcache (metrics.xml 文書に報告されます) pkgcache_metrics	ACTIVITY METRICS BASE

使用法

次の数式を使用して、拡張ラッチの平均待機時間 (ミリ秒) を計算できます。

```
TOTAL_EXTENDED_LATCH_WAIT_TIME / TOTAL_EXTENDED_LATCH_WAITS
```

total_move_time 合計エクステント移動時間 : モニター・エレメント

表スペースのリバランス・プロセス中に移動したすべてのエクステントの移動時間の合計 (ミリ秒)。

表 1926. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE

total_nested_invocations - ネストされた呼び出し合計のモニター・エレメント

1 より大きいネスト・レベルで呼び出されたルーチンの回数。

例えば、ルーチンまたはトリガーが、別のルーチンまたはトリガーのコンテキストで呼び出される場合です。

表 1927. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

total_routine_coord_time - ルーチン・コーディネーター合計時間のモニター・エレメント

コーディネーター・エージェントがルーチンの実行に費やした合計時間。

プロシージャー、コンパイル済み SQL 関数、コンパイル済みトリガー、および動的に準備されたコンパウンド SQL ステートメントの場合、このエレメントはルーチンで費やされた合計経過時間を表します。要求メトリックが無効な場合、この値はゼロになります。

表 1928. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	REQUEST METRICS BASE
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	REQUEST METRICS BASE

total_times_routine_invoked - ルーチン呼び出しの合計オカレンスのモニター・エレメント

ルーチンがメンバーで実行された回数。

表 1929. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

total_hash_joins ハッシュ結合の合計

実行されたハッシュ結合の合計数。

エレメント ID
total_hash_joins

エレメント・タイプ
カウンター

表 1930. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1931. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 データベースまたはアプリケーション・レベルで、この値と hash_join_overflows および hash_join_small_overflows を組み合わせて使用すると、ソート・ヒープ・サイズを適度に変更することがハッシュ結合に有効かどうかを判別できます。

total_hash_loops ハッシュ・ループの合計

ハッシュ結合の単一パーティションが、使用可能なソート・ヒープ・スペースよりも大きかったときの合計回数。

表 1932. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1933. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントの値は、ハッシュ結合が効率的に実行されていないことを示します。ソート・ヒープ・サイズが小さすぎるか、またはソート・ヒープしきい値が小さすぎることを示します。この値とその他のハッシュ結合変数を組み合わせて使用すると、ソート・ヒープ・サイズ (*sortheap*) とソート・ヒープしきい値 (*sheapthres*) の構成パラメーターを調整できます。

total_implicit_compilations - 暗黙的なコンパイル数の合計 : モニター・エレメント

total_implicit_compilations モニター・エレメントは、データベース・サーバーでの暗黙的なコンパイルの合計数を格納します。暗黙的なコンパイルとは、ユーザーが直接要求していないコンパイルのことです。

すなわち、バインド、再バインド、準備、または即時に実行などによるものではないということです。例えば、実行時にコンパイルされる必要がある、VALIDATE RUN オプションでバインドされたステートメントを実行すると、暗黙的なコンパイルが発生する可能性があります。

表 1934. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1934. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1935. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_implicit_compile_proc_time - 暗黙的なコンパイル処理時間の合計 : モニター・エレメント

total_implicit_compile_proc_time モニター・エレメントは、データベース・サーバーで暗黙的なコンパイルを実行するのに費やした処理時間 (待機時間ではない) の合計を格納します。暗黙的なコンパイルとは、ユーザーが直接要求していないコンパイルのことです。

すなわち、バインド、再バインド、準備、または即時に実行などによるものではないということです。例えば、実行時にコンパイルされる必要がある、VALIDATE

RUN オプションでバインドされたステートメントを実行すると、暗黙的なコンパイルが発生する可能性があります。値はミリ秒単位で示されます。

表 1936. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1937. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 1937. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_implicit_compile_time - 暗黙的なコンパイル時間の合計 : モニター・エレメント

total_implicit_compile_time モニター・エレメントは、データベース・サーバー上で暗黙的なコンパイルを実行するのにかかった時間の合計を格納します。暗黙的なコンパイルとは、ユーザーが直接要求していないコンパイルのことです。

すなわち、バインド、再バインド、準備、または即時に実行などによるものではないということです。例えば、実行時にコンパイルされる必要がある、VALIDATE RUN オプションでバインドされたステートメントを実行すると、暗黙的なコンパイルが発生する可能性があります。値はミリ秒単位で示されます。

表 1938. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 1938. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1939. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_load_proc_time - ロード処理時間の合計 : モニター・エレメント

データベース・サーバー上でロード処理を実行するのにかかった処理時間 (待機ではない) の合計。値はミリ秒単位で示されます。

表 1940. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1940. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1941. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_load_time - 合計ロード時間 : モニター・エレメント

データベース・サーバー上でロードを実行するのにかかった時間の合計。値はミリ秒単位で示されます。

表 1942. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1942. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1943. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_loads - ロード合計 : モニター・エレメント

データベース・サーバーで実行されたロードの合計数。

表 1944. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1945. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_log_available 使用可能なログの合計

非コミット・トランザクションによって使用されていない、データベース内のアクティブ・ログ・スペースの量 (バイト単位)。

表 1946. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1947. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法

このエレメントを total_log_used とともに使用して、ログ・スペースを使い果たすことを避けるために以下の構成パラメーターを調整する必要があるかどうかを判別します。

- logfilsiz
- logprimary
- logsecond

total_log_available の値が 0 まで下がった場合、SQL0964N が返されます。上記の構成パラメーターの値を大きくするか、あるいは COMMIT、ROLLBACK または FORCE APPLICATION によって最も古いトランザクションを終了する必要があります。

logsecond が -1 に設定されていると、このエレメントには SQLM_LOGSPACE_INFINITE が含まれます。

注: データベース・システム・モニター情報はバイト単位で示されますが、構成パラメーターは各 4K バイトのページ単位で設定されます。

total_log_used 使用されているログ・スペースの合計

データベースで現在使用されているアクティブ・ログ・スペースの合計量 (バイト単位)。

表 1948. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE

表 1949. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを total_log_available とともに使用して、ログ・スペースを使い果たすことを避けるために以下の構成パラメーターを調整する必要があるかどうかを判別します。

- logfilsiz
- logprimary
- logsecond

注: データベース・システム・モニター情報はバイト単位で示されますが、構成パラメーターは各 4K バイトのページ単位で設定されます。

total_move_time 合計エクステント移動時間 : モニター・エレメント

表スペースのリバランス・プロセス中に移動したすべてのエクステントの移動時間の合計 (ミリ秒)。

表 1950. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE

total_olap_funcs OLAP 関数の合計数 : モニター・エレメント

実行された OLAP 関数の合計数。

表 1951. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1952. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法

データベースまたはアプリケーション・レベルで、この値と olap_func_overflows を組み合わせて使用すると、ソート・ヒープ・サイズを適度に増やすことが、多くの割合の OLAP 関数に有効かどうかを判別できます。

total_peas - partial early aggregation の合計回数のモニター・エレメント

partial early aggregation 操作が実行された合計回数。

表 1953. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	REQUEST METRICS BASE

表 1954. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1955. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
接続	event_conn	-
ステートメント	event_stmt	-
トランザクション	event_xact	-
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

データベースまたはアプリケーション・レベルで、この値と **post_threshold_peas** を組み合わせて使用すると、ソート・ヒープ・サイズまたはソート・ヒープしきい値を大きくすることが、 partial early aggregation 操作に有効かどうかを判別できます。 **total_peas** に対する **post_threshold_peas** の比率が高い場合は、ソート・ヒープ・サイズかソート・ヒープしきい値、またはその両方を大きくすると、データベースまたはアプリケーションのパフォーマンスが向上する可能性があります。

total_peds - partial early distinct の合計回数のモニター・エレメント

partial early distinct 操作が実行された合計回数。

表 1956. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 1957. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1957. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1958. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE

表 1958. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
接続	event_conn	-
ステートメント	event_stmt	-
トランザクション	event_xact	-
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

データベースまたはアプリケーション・レベルで、この値と **disabled_peds** モニター・エレメントおよび **post_threshold_peds** モニター・エレメントを組み合わせて使用すると、ソート・ヒープ・サイズまたはソート・ヒープしきい値を大きくすることが、partial early distinct 操作に有効かどうかを判別できます。 **total_peds** モニター・エレメントに対する、**disabled_peds** モニター・エレメントおよび **post_threshold_peds** モニター・エレメントの比率が高い場合は、ソート・ヒープ・サイズまたはソート・ヒープしきい値、またはその両方を大きくすると、データベースまたはアプリケーションのパフォーマンスが向上する可能性があります。

total_reorg_proc_time - 再編成処理時間の合計 : モニター・エレメント

データベース・サーバー上で再編成の操作を実行するのにかかった処理時間 (待機ではない) の合計。値はミリ秒単位で示されます。

表 1959. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1959. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1960. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_reorg_time - 合計再編成時間 : モニター・エレメント

データベース・サーバー上で再編成の操作を実行するのにかかった時間の合計。値はミリ秒単位で示されます。

表 1961. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1962. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 1962. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_reorgs - 再編成の合計 : モニター・エレメント

データベース・サーバーに対して発行された再編成操作の数。

表 1963. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1964. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_rollback_proc_time - ロールバック処理時間の合計 : モニター・エレメント

データベース・サーバー上でロールバック操作を実行するのにかかった処理時間 (待機ではない) の合計。値はミリ秒単位で示されます。

表 1965. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1965. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1966. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_rollback_time - 合計ロールバック時間 : モニター・エレメント

データベース・サーバー上でロールバック操作を実行するのにかかった時間の合計。値はミリ秒単位で示されます。

表 1967. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 1967. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1968. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_routine_invocations - ルーチンの合計呼び出し数 : モニター・エレメント

ルーチンが呼び出された回数の合計。

表 1969. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1969. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1970. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE

表 1970. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_routine_non_sect_proc_time - 非セクション処理時間 : モニター・エレメント

このステートメントがルーチン内で非セクションの実行にかかった処理時間の合計。この値には、ルーチン内でユーザー・コードを実行した時間およびコミットやロールバックなど、非セクション操作を実行するのにかかった時間も含まれています。処理時間には待ち時間は含まれません。値はミリ秒単位で示されます。

表 1971. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1972. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

total_routine_non_sect_time - 非セクション・ルーチンの実行時間 : モニター・エレメント

このステートメントがルーチン内で非セクションを実行した時間の合計。この値には、ルーチン内でユーザー・コードを実行した時間およびコミットやロールバックなど、非セクション操作を実行するのにかかった時間も含まれています。値はミリ秒単位で示されます。

表 1973. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 1974. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

total_routine_time - 合計ルーチン時間 : モニター・エレメント

ルーチンの実行にかかった時間の合計。値はミリ秒単位で示されます。

表 1975. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 1975. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1976. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

収集レベルが BASE に設定されると、**total_routine_time** モニター・エレメントの値には NO SQL 節を使用して定義された関数の実行経過時間は含まれません。

収集レベルが EXTENDED に設定される場合、**total_routine_time** モニター・エレメントの値にはすべてのルーチンでの経過時間が含まれます。

total_routine_user_code_proc_time - ルーチンのユーザー・コード処理時間の合計 : モニター・エレメント

既知の DB2 時間以外の、ルーチン内 (通常はルーチンのユーザー・コード) の実行にかかった処理時間の合計。値はミリ秒単位で示されます。

表 1977. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1977. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1978. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

収集レベルが BASE に設定されると、このモニター・エレメントには、NO SQL 節を使用して定義された関数の実行処理経過時間は含まれません。代わりに、この時間は **total_section_proc_time** モニター・エレメントの値に含まれます。

収集レベルが EXTENDED に設定される場合、このモニター・エレメントの値にはすべてのルーチンの実行処理経過時間が含まれます。

total_routine_user_code_time - ルーチンのユーザー・コード時間の合計 : モニター・エレメント

既知の DB2 時間以外の、ルーチン内 (通常はルーチンのユーザー・コード) の実行にかかった時間の合計。値はミリ秒単位で示されます。

表 1979. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 1979. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1980. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

収集レベルが BASE に設定されると、このモニター・エレメントの値には、NO SQL 節を使用して定義された関数の実行経過時間は含まれません。代わりに、この時間は **total_section_time** モニター・エレメントの値に含まれます。

収集レベルが EXTENDED に設定される場合、このモニター・エレメントの値にはすべてのルーチンの実行経過時間が含まれます。

total_rqst_mapped_in - マッピングの際の要求の合計 : モニター・エレメント

再マップしきい値または作業アクション・セットによって、このサービス・サブクラスにマップする際の要求の合計数。

表 1981. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1982. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE

total_rqst_mapped_out - マッピングの際に除外された要求の合計 : モニター・エレメント

再マップしきい値または作業アクション・セットによって、このサービス・サブクラスからマッピングの際に除外された要求の合計数。

表 1983. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1984. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE

total_rqst_time - 合計要求時間 : モニター・エレメント

要求の処理にかかった時間の合計。この値はミリ秒単位で報告されます。

表 1985. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1986. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_runstats - ランタイム統計の合計 : モニター・エレメント

データベース・サーバーで実行された RUNSTATS 操作の合計数。

表 1987. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1988. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_runstats_proc_time - ランタイム統計処理時間の合計 : モニター・エレメント

データベース・サーバー上で RUNSTATS の操作を実行するのにかかった処理時間 (待機ではない) の合計。値はミリ秒単位で示されます。runstats ユーティリティーがスロットルされている時間は、runstats 処理時間には含まれません。

表 1989. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 1989. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1990. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_runstats_time - ランタイム統計時間の合計 : モニター・エレメント

データベース・サーバー上で RUNSTATS 操作を実行するのにかかった時間の合計。値はミリ秒単位で示されます。

表 1991. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 1991. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1992. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_sec_cons 2 次接続

サブエージェントがノードのデータベースに接続した数。

エレメント ID

total_sec_cons

エレメント・タイプ

カウンター

表 1993. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントと total_cons、db_conn_time、および db2start_time のモニター・エレメントを組み合わせて使用すると、各アプリケーションがデータベースに接続した頻度を計算できます。

total_section_proc_time - セクション処理時間の合計 : モニター・エレメント

エージェントがセクション実行にかかった処理時間の合計。処理時間には待ち時間は含まれません。値はミリ秒単位で示されます。

表 1994. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 1994. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1995. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

収集レベルが BASE に設定されると、**total_section_proc_time** モニター・エレメントの値には、NO SQL 節を使用して定義された関数の実行処理経過時間が含まれます。

収集レベルが EXTENDED に設定される場合には、こうした関数の実行処理経過時間は、**total_section_proc_time** モニター・エレメントの値には含まれません。この時間は、**total_routine_user_code_proc_time** モニター・エレメントの値に含まれます。

total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント

セクションの実行 (クライアント・アプリケーションにより発行される SQL ステートメントによって生成される、コンパイルされた照会プランの実行) 中にソートを実行するのにかかった処理 (待機以外の) 時間の合計。値はミリ秒単位で示されます。

表 1996. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1997. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

システム・レベルで、このエレメントと **total_section_sorts** モニター・エレメントを組み合わせて使用すると、セクションの実行中の平均ソート処理時間 (待機を含まない) を計算できます。これは、ソートがパフォーマンス上の問題となっているかどうかを表します。

アクティビティー・レベルでこのエレメントを使用すると、ソート時間の多いステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート時間を少なくすると効率が上がります。

total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント

セクションの実行 (クライアント・アプリケーションにより発行される SQL ステートメントによって生成される、コンパイルされた照会プランの実行) 中にソートを実行するのにかかった時間の合計。値はミリ秒単位で示されます。

表 1998. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 1998. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1999. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE

表 1999. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

システム・レベルで、このエレメントと **total_section_sorts** モニター・エレメントを組み合わせて使用すると、セクションの実行中の平均ソート時間を計算できます。これは、ソートがステートメントのパフォーマンス上の問題となっているかどうかを表します。

total_section_sort_time エレメントには、待機時間と処理時間の両方が含まれます。(total_section_sort_time - total_section_sort_proc_time) の値が大きい場合、ソートは待機に多くの時間を費やしていることになります。例えば、ソートが頻繁にディスクにスピルする場合、入出力待機のために **total_section_sort_time** モニター・エレメントの値が増加します。この時間は、ソートをアクティブに処理している時間のみをカウントする **total_section_sort_proc_time** モニター・エレメント値には含まれません。この場合、パフォーマンスを改善するためにソート・メモリーの調整を考慮できます。

アクティビティー・レベルでこのエレメントを使用すると、ソート時間の多いステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート時間を少なくすると効率が上がります。

total_section_sorts - セクションのソートの合計: モニター・エレメント

セクションの実行 (クライアント・アプリケーションにより発行される SQL ステートメントによって生成される、コンパイルされた照会プランの実行) 中に実行されたソートの合計回数。

表 2000. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 2000. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2001. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

表 2001. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このエレメントを **total_section_sort_time** モニター・エレメントと組み合わせて使用すると、セクションの実行中にソートの実行にかかった平均時間を計算します。

アクティビティーおよびパッケージ・キャッシュ・レベルでこのエレメントを使用すると、多数のソート操作を実行するステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート回数を少なくすると利点があります。また、EXPLAIN ステートメントを使用すると、 1 つのステートメントが実行するソートの回数を識別できます。

total_section_time - 合計セクション時間 : モニター・エレメント

エージェントがセクション実行にかかった時間の合計。値はミリ秒単位で示されます。

表 2002. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 2002. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2003. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

収集レベルが BASE に設定されると、**total_section_time** モニター・エレメントの値には、NO SQL 節を使用して定義された関数の実行経過時間が含まれます。

収集レベルが EXTENDED に設定される場合には、こうした関数の実行経過時間は、**total_section_time** モニター・エレメントの値には含まれません。代わりに、**total_routine_user_code_time** モニター・エレメントの値に含まれます。

total_sort_time - ソート時間合計 : モニター・エレメント

実行されたすべてのソートの合計経過時間。この値はミリ秒単位で報告されます。

表 2004. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	Sort
アプリケーション	appl	Sort
アプリケーション	stmt	Sort
動的 SQL	dynsql	Sort

スナップショット・モニターの場合、このカウンターはリセットできます。

表 2005. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	ステートメント、ソート

使用法

データベースまたはアプリケーションのレベルでこのエレメントと **total_sorts** を組み合わせて使用すると、平均ソート時間を計算できます。平均ソート時間は、ソートがパフォーマンス上の問題となっているかどうかを表します。

ステートメント・レベルでこのエレメントを使用すると、ソート時間の多いステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート時間を少なくすると効率が上がります。

このカウントには、関連操作のために作成される一時表のためのソート時間も含まれています。このカウントは、1 ステートメント、1 アプリケーション、または 1 つのデータベースにアクセスするすべてのアプリケーションについて情報を提供します。

経過時間を示すモニター・エレメントを使用するときは、次のことを考慮してください。

1. 経過時間は、システム負荷の影響を受けるので、実行する処理数が多くなると、この経過時間の値は大きくなる。
2. このモニター・エレメントをデータベース・レベルで計算する場合、データベース・システム・モニターはアプリケーション・レベルの時間を合計します。この場合、同時に複数のアプリケーション処理が実行されていることがあるので、データベース・レベルでは時間が二重に計算されます。

データベース・レベルで意味のあるデータを得るためには、データを低いレベルに正規化する必要があります。例えば、

```
total_sort_time / total_sorts
```

これは、ソート当たりの平均経過時間に関する情報を示しています。

total_sorts - ソート合計 : モニター・エレメント

実行されたソートの合計数。

表 2006. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2006. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2007. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 2008. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	ステートメント、ソート
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

データベースまたはアプリケーションのレベルでは、この値と **sort_overflows** を組み合わせて使用すると、ヒープ・スペースをさらに必要とするソートのパーセンテージを計算できます。さらに、**total_sort_time** と組み合わせると、平均ソート時間を計算できます。

ソート合計数に対してソートのオーバーフロー回数が少ない場合は、ソート・ヒープ・サイズを大きくしても、バッファー・サイズを極端に大きくしなければ、パフォーマンスに影響を与えることはほとんどありません。

ステートメント・レベルでこのエレメントを使用すると、多数のソート操作を実行するステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート回数を少なくすると利点があります。また、SQL EXPLAIN ステートメントを使用すると、 1 つのステートメントが実行するソートの回数を識別できます。

total_stats_fabrication_proc_time - 統計作成の合計処理時間のモニター・エレメント

リアルタイム統計収集により統計作成に費やされた、待機以外の時間の合計 (ミリ秒)。統計作成とは、照会をコンパイルする際に、統計を生成するのに必要な統計収集アクティビティーのことです。

表 2009. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2009. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2010. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_stats_fabrication_time - 統計作成の合計時間のモニター・エレメント

リアルタイム統計収集により統計作成で費やされた合計時間 (ミリ秒単位)。統計作成とは、照会をコンパイルする際に、統計を生成するのに必要な統計収集アクティビティーのことです。

表 2011. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT _TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 2011. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2012. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	uow_metrics	REQUEST METRICS BASE
パッケージ・キャッシュ	pkgcache	常に収集される

total_stats_fabrications - 統計作成の合計回数のモニター・エレメント

リアルタイム統計収集によって実行された統計作成の合計回数。統計作成とは、照会をコンパイルする際に、統計を生成するのに必要な統計収集アクティビティーのことです。

表 2013. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2014. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	常に収集される

total_sync_runstats_time - 同期 RUNSTATS の合計時間のモニター・エレメント

リアルタイム統計収集により起動される同期 RUNSTATS アクティビティーの合計時間 (ミリ秒単位)。同期 RUNSTATS アクティビティーは、照会のコンパイル中に発生します。

表 2015. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW - フォーマット設定された行ベースのコンポーネント時間の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される

表 2015. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2016. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	常に収集される

total_sync_runstats_proc_time - 同期 RUNSTATS の合計処理時間のモニター・エレメント

リアルタイム統計収集によって起動された同期 RUNSTATS アクティビティーに費やされた、待機以外の時間 (ミリ秒)。同期 RUNSTATS アクティビティーは、照会のコンパイル中に発生します。

表 2017. 表関数モニター情報

表関数	**モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)**
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2018. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

total_sync_runstats - 同期 RUNSTATS アクティビティーの合計回数のモニター・エレメント

リアルタイム統計収集によって起動された同期 RUNSTATS アクティビティーの合計回数。同期 RUNSTATS アクティビティーは、照会のコンパイル中に発生します。

表 2019. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 2019. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2020. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	常に収集される

total_sys_cpu_time ステートメントのシステム CPU の合計時間 : モニター・エレメント

SQL ステートメントに使われたシステム CPU 時間の合計。

表 2021. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントをステートメント実行の経過時間およびステートメントのユーザー CPU の合計とともに使用して、最もコストがかかるステートメントを評価します。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

total_sorts - ソート合計 : モニター・エレメント

実行されたソートの合計数。

表 2022. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 2022. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2023. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 2024. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	ステートメント、ソート
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

データベースまたはアプリケーションのレベルでは、この値と **sort_overflows** を組み合わせて使用すると、ヒープ・スペースをさらに必要とするソートのパーセンテージを計算できます。さらに、**total_sort_time** と組み合わせると、平均ソート時間を計算できます。

ソート合計数に対してソートのオーバーフロー回数が少ない場合は、ソート・ヒープ・サイズを大きくしても、バッファー・サイズを極端に大きくしなければ、パフォーマンスに影響を与えることはほとんどありません。

ステートメント・レベルでこのエレメントを使用すると、多数のソート操作を実行するステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート回数を少なくすると利点があります。また、SQL EXPLAIN ステートメントを使用すると、 1 つのステートメントが実行するソートの回数を識別できます。

total_usr_cpu_time ステートメントのユーザー CPU の合計時間 : モニター・エレメント

SQL ステートメントに使われたユーザー CPU 時間の合計。

表 2025. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントをステートメント実行の経過時間とともに使用して、最も実行時間が長いステートメントを評価します。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

total_wait_time - 合計待機時間 : モニター・エレメント

DB2 データベース・サーバー内での待機にかかった合計時間。値はミリ秒単位で示されます。

表 2026. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2027. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

データベース・サーバーがアクティブに要求を処理するのに費やす時間のパーセンテージを判別するには、次の比率を使用します。

```
(total_rqst_time - total_wait_time) / total_rqst_time
```

client_idle_wait_time モニター・エレメントの値は、**total_wait_time** モニター・エレメントの値に含まれません。 **total_wait_time** エレメントはデータベース・サーバーが要求を処理している間の待機にかかった時間のみを表します。

tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント

この接続で sqleseti API が発行された場合に、ロギングおよび診断の目的でターゲット・データベースに渡されたデータです。この接続、作業単位、またはアクティビティーの CLIENT_ACCTNG 特殊レジスターの現行値。

このモニター・エレメントは、**client_acctng** モニター・エレメントと同義です。**client_acctng** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_acc_str** モニター・エレメントは、表、ファイルおよびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 2028. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl	基本

表 2029. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
デッドロック	event_dlconn	-
トランザクション	event_xact	-

使用法

このエレメントは、問題判別およびアカウンティングの目的で使用します。

tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント

この接続で sqleseti API が発行された場合に、トランザクションを実行中のサーバー・トランザクション・プログラムを示します。この接続、作業単位、またはアクティビティーの CLIENT_APPLNAME 特殊レジスターの現行値。

このモニター・エレメントは、**client_applname** モニター・エレメントと同義です。 **client_applname** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_app** モニター・エレメントは、表、ファイルおよびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 2030. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl	基本

表 2031. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
デッドロック	event_dlconn	-
トランザクション	event_xact	-

使用法

このエレメントは、問題判別およびアカウンティングの目的で使用します。

tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント

sqleseti API が使用された場合は、トランザクション・マネージャーが生成してサーバーに提供したクライアント・ユーザー ID。この接続、作業単位、またはアクティビティーの CLIENT_USERID 特殊レジスターの現行値。

このモニター・エレメントは、**client_userid** モニター・エレメントと同義です。**client_userid** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_userid** モニター・エレメントは、表、ファイルおよびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 2032. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl	基本

表 2033. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
デッドロック	event_dlconn	-
トランザクション	event_xact	-

使用法

アプリケーション・サーバーまたはトランザクション処理モニター環境でこのエレメントを使用すると、トランザクションの実行対象になっているエンド・ユーザーを識別できます。

tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント

この接続で sqleseti API が発行された場合に、クライアントのシステムまたはワークステーション (CICS EITERMID など) を示します。この接続、作業単位、またはアクティビティーの CLIENT_WRKSTNNAME 特殊レジスターの現行値。

このモニター・エレメントは、**client_wrkstnname** モニター・エレメントと同義です。 **client_wrkstnname** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_wkstn** モニター・エレメントは、表、ファイルおよびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 2034. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl	基本

表 2035. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
デッドロック	event_dlconn	-
トランザクション	event_xact	-

使用法

このエレメントを使用すると、ノード ID、端末 ID、またはその他の同様の ID を使用して、ユーザーのマシンを識別できます。

tq_cur_send_spills オーバーフローした表キュー・バッファーの現在数 : モニター・エレメント

一時表内にある表キュー・バッファーの現在の数。

表 2036. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

使用法 表キューへの書き込みを行っているエージェントは、複数のリーダーに行を送信している可能性があります。書き込みエージェントが行を送信したときに、相手のエージェントが行を受け付けず、別のエージェントが処理のために行を必要とすると、書き込みエージェントはバッファーを一時表にオーバーフローします。一時表にオーバーフローすると、ライターとほかのリーダーがともに処理を続行できるようになります。

オーバーフローした行は、読み取りエージェントが行を受け付ける準備ができると読み取りエージェントに送信されます。

この数値が大きく、照会が sqlcode -968 で失敗し、さらに db2diad.log 内のメッセージにより TEMP 表スペースの一時スペースがなくなったことを示す場合は、表キューのオーバーフローが原因の可能性があります。この場合、ほかのノードで問題が発生していることを示します (ロッキングなど)。この照会のすべてのパーティション上でスナップショットを取って、調査してください。

データをパーティション化する方法が原因で、照会によって多数のバッファーをオーバーフローすることが必要になる場合もあります。このような原因がある場合は、 TEMPORARY 表スペースにさらにディスクを追加する必要があります。

tq_id_waiting_on ノード上の表キュー待機 : モニター・エレメント

データの送受信を待機している表キューの ID。

表 2037. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

使用法

これはトラブルシューティングに使用できます。

tq_max_send_spills 表キュー・バッファー・オーバーフローの最大数

一時表にオーバーフローした表キュー・バッファーの最大数。

エレメント ID

tq_max_send_spills

エレメント・タイプ
水準点

表 2038. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

表 2039. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	常に収集される

使用法 一時表に書き込まれた表キュー・バッファーの最大数を示します。

tq_node_waited_for 表キュー上のノード待機

サブセクション状況の ss_status が「*受信待ち*」または「*送信待ち*」で、tq_wait_for_any が FALSE の場合は、エージェントが待機中のノード番号を示します。

エレメント ID
tq_node_waited_for

エレメント・タイプ
情報

表 2040. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

使用法 これはトラブルシューティングに使用できます。サブセクションが待機しているノード上でアプリケーションのスナップショットが必要になる場合があります。例えば、アプリケーションがそのノード上でロック待機になっている場合です。

tq_rows_read 表キューから読み取られた行数

表キューから読み取られた合計行数。

エレメント ID
tq_rows_read

エレメント・タイプ
カウンター

表 2041. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

表 2042. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	常に収集される

使用法 この数が増加していることをモニターが示していない場合は、処理は進行していません。

ノードごとにこの数値が大きく異なる場合は、使用率が過剰なノードと過少なノードがあります。

この数値が大きい場合は、ノード間で転送されるデータ量が多く、最適化によりアクセス・プランを改善できることを示します。

tq_rows_written 表キューに書き込まれた行数

表キューに書き込まれた合計行数。

エレメント ID

tq_rows_written

エレメント・タイプ

カウンター

表 2043. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

表 2044. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	常に収集される

使用法 この数が増加していることをモニターが示していない場合は、処理は進行していません。

ノードごとにこの数値が大きく異なる場合は、使用率が過剰なノードと過少なノードがあります。

この数値が大きい場合は、ノード間で転送されるデータ量が多く、最適化によりアクセス・プランを改善できることを示します。

tq_sort_heap_rejections - 表キュー・ソート・ヒープ拒否のモニター・エレメント

表キューがソート・ヒープ・メモリーを要求したが、ソート・ヒープしきい値を超過するため拒否された回数。

表 2045. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 2045. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	REQUEST METRICS BASE

表 2046. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 2046. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2047. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

表 2047. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-
ステートメント	event_stmt	-
トランザクション	event_xact	-
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントと **tq_sort_heap_requests** モニター・エレメントを組み合わせて使用すると、表キューに十分なソート・ヒープ・メモリーが付与されているかを大まかに調べることができます。 **tq_sort_heap_requests** モニター・エレメントに対する **tq_sort_heap_rejections** モニター・エレメントの比率が高い場合、データベースのパフォーマンスは最適ではない可能性があります。ソート・ヒープ・サイズを大きくすることを検討してください。

tq_sort_heap_requests - 表キュー・ソート・ヒープ要求のモニター・エレメント

表キューがデータを格納するためにソート・ヒープ・メモリーを要求した回数。

表 2048. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 2048. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2049. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
接続	event_conn	-
ステートメント	event_stmt	-
トランザクション	event_xact	-
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントと **tq_sort_heap_rejections** モニター・エレメントを組み合わせて使用すると、表キューに十分なソート・ヒープ・メモリーが付与されているかを大まかに調べることができます。 **tq_sort_heap_requests** モニター・エレメントに対する **tq_sort_heap_rejections** モニター・エレメントの比率が高い場合、データベ

ースのパフォーマンスは最適ではない可能性があります。ソート・ヒープ・サイズを大きくすることを検討してください。

tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント

一時表にオーバーフローした表キュー・バッファーの合計数。

表 2050. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2050. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2051. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

表 2052. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
ステートメント	event_subsection	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

一時表に書き込まれた表キュー・バッファーの合計数を示します。詳しくは、**tq_cur_send_spills** モニター・エレメントを参照してください。

tq_wait_for_any 表キュー上のノード送信待機

このフラグは、サブセクションがノードからの行の受信を待っているためにサブセクションがブロックされていることを示すのに使用されます。

エレメント ID

tq_wait_for_any

エレメント・タイプ

情報

表 2053. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

使用法 ss_status が「*表キュー上でデータの受信待ち*」を示し、このフラグが TRUE の場合は、サブセクションはノードからの行の受信を待機中です。この場合は通常、 SQL ステートメントが待機中のエージェントにデータを渡すところまでまだ処理が進んでいないことを示します。例えば、書き込みエージェントがソートを実行中で、ソートが終了するまでは行を書き込まない場合があります。 db2expln の出力を使用して、エージェントが行の受信を待機している表キューに関連付けられたサブセクション番号を判別します。次に、サブセクションを実行している各ノードでスナップショットをとると、サブセクションの状況を確認できます。

ts_name - ロールフォワードされている表スペース : モニター・エレメント

現在ロールフォワードされている表スペースの名前。

エレメント ID

ts_name

エレメント・タイプ

情報

表 2054. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	基本

使用法 ロールフォワードが進行中の場合は、このエレメントは関係する表スペースを示します。

txn_completion_status - トランザクション完了状況

このエレメントは、トランザクションの状況を示します。

表 2055. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	TXNCOMPLETION	常に収集される

使用法

変更履歴イベント・モニターの場合、トランザクション状況は、以下のいずれかです。

C Commit

R ロールバック

S セーブポイントへのロールバック

uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント

実行された SQL UPDATE、INSERT、および DELETE ステートメントの数。

表 2056. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 2057. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、アプリケーション・レベルまたはデータベース・レベルのデータベース・アクティビティーのレベルを判別できます。

次の公式を使用すると、すべてのステートメントに対する UPDATE、INSERT および DELETE ステートメントの比率を計算できます。

```
  uid_sql_stmts
/ (static_sql_stmts + dynamic_sql_stmts )
```

この情報は、アプリケーションのアクティビティーおよびスループットの分析に役立ちます。

unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント

プリフェッチャー読み取りが使用されなかったページ数を示します。

表 2058. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 2059. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 2060. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
表スペース	event_tablespace	常に収集される
接続	event_conn	常に収集される

使用法

ページ数が多い場合、プリフェッチャーは、使用されないページをバッファー・プール内に読み込むことで不必要な入出力を行うことがあります。

uow_comp_status 作業単位完了状況

作業単位の状況およびそれが停止したときの状況。

エレメント ID

uow_comp_status

エレメント・タイプ

情報

表 2061. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位
DCS アプリケーション	dcs_appl	基本

表 2062. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	常に収集される

使用法 このエレメントを使用すると、作業単位が終了した原因がデッドロックによるものか、または異常終了によるものかを判別できます。次の原因が考えられます。

- コミット・ステートメントによりコミットされた。
- ロールバック・ステートメントによりロールバックされた。
- デッドロックによりロールバックされた。
- 異常終了によりロールバックされた。
- アプリケーションの正常終了によりコミットされた。
- 進行中であった作業単位に対する FLUSH EVENT MONITOR コマンドの結果が不明。

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル (*sqlmon.h*) を参照してください。

uow_completed_total - 完了済みの合計作業単位 : モニター・エレメント

コミットまたはロールバックされて完了した作業単位の合計数。

表 2063. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 2064. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	常に収集される
統計	event_wlstats (メトリック文書に報告されます)	常に収集される

使用法

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降に完了した作業単位の合計数を表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降に完了した作業単位の合計数を表します。

uow_elapsed_time 最新の作業単位の経過時間

最後に完了した作業単位の実行経過時間。

エレメント ID

uow_elapsed_time

エレメント・タイプ

time

表 2065. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ

表 2065. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

使用法

作業単位の完了にかかる時間の標識として、このエレメントを使用します。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

uow_id 作業単位 ID : モニター・エレメント

作業単位の ID。作業単位 ID は、アプリケーション・ハンドル内で固有です。

表 2066. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 2067. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
アクティビティー	event_activitystmt	常に収集される
アクティビティー	event_activityvals	常に収集される
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE

表 2067. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	ddlstmtexec txncompletion	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

さらにこのエレメントを **activity_id** および **appl_id** モニター・エレメントと一緒に使用すると、アクティビティーを一意的に識別できます。

uow_lifetime_avg - 作業単位の平均存続期間 : モニター・エレメント

作業単位の平均存続期間。ミリ秒で計測されます。

表 2068. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 2069. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	常に収集される
統計	event_wlstats (メトリック文書に報告されます)	常に収集される

使用法

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降の作業単位の平均存続期間を示します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が

実行されて以降の作業単位の平均存続期間を表します。

uow_lock_wait_time - ロック待機中の作業単位の合計時間 : モニター・エレメント

この作業単位がロックの待機に要した合計経過時間。値はミリ秒単位で示されます。

エレメント ID
uow_lock_wait_time

エレメント・タイプ
カウンター

表 2070. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位

使用法 このエレメントは、リソース競合問題の重大度を判別するときに利用できます。

uow_log_space_used - 使用されている作業単位ログ・スペース: モニター・エレメント

モニター対象のアプリケーションで現行作業単位に使用されているログ・スペースの量 (バイト単位)。

表 2071. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 2072. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位

表 2073. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	常に収集される
作業単位	-	常に収集される

使用法

このエレメントを使用すると、作業単位レベルでのロギングの所要量を把握することができます。

uow_start_time - 作業単位開始タイム・スタンプ : モニター・エレメント

作業単位が最初にデータベース・リソースを要求した日時。

表 2074. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 2075. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

表 2076. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
トランザクション	event_xact	-

使用法

このリソース要求は、その作業単位で SQL ステートメントを初めて実行したときに発生します。

- 最初の作業単位の場合は、**conn_complete_time** の後の最初のデータベース要求 (SQL ステートメントの実行) の時刻。
- その後の作業単位の場合は、前回の COMMIT または ROLLBACK の後の最初のデータベース要求 (SQL ステートメントの実行) の時刻。

注: 「*SQL リファレンス*」は、作業単位の境界を COMMIT または ROLLBACK のポイントとして定義します。

データベース・システム・モニターでは、 COMMIT/ROLLBACK とその作業単位定義から出される次の SQL ステートメントまでの経過時間を除外します。この測定方式により、データベース・マネージャーがデータベース要求の処理に要する時間を、その作業単位の最初の SQL ステートメント以前にアプリケーション・ロジック内で要する時間とは切り離して反映します。作業単位の経過時間には、作業単位内で SQL ステートメント間のアプリケーション・ロジックを実行する時間が含まれます。

このエレメントと **uow_stop_time** モニター・エレメントを組み合わせると、作業単位の合計経過時間を計算できます。**prev_uow_stop_time** モニター・エレメントと組み合わせると、作業単位間にアプリケーションで要した時間を計算できます。

uow_stop_time と **prev_uow_stop_time** モニター・エレメントを組み合わせると、*SQL リファレンス*の定義による作業単位の経過時間を計算できます。

uow_status 作業単位の状況

作業単位の状況。

エレメント ID

uow_status

エレメント・タイプ

情報

表 2077. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	常に収集される

使用法 このエレメントを使用すると、作業単位の状況を判別できます。 API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル sqlmon.h を参照してください。

uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント

最新の作業単位が完了した日時。これが起こるのはデータベースの変更がコミットまたはロールバックされたときです。

表 2078. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

表 2079. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-

使用法

このエレメントと **prev_uow_stop_time** モニター・エレメントを組み合わせて使用すると、COMMIT/ROLLBACK ポイント間の合計経過時間を計算できます。**uow_start_time** モニター・エレメントと組み合わせると、前回の作業単位の経過時間を計算できます。

タイム・スタンプの内容は、次のように設定されます。

- アプリケーションが 1 つの作業単位を完了し、(**uow_start_time** モニター・エレメントで定義されたように) 新しい作業単位をまだ開始していない場合、このエレメントは有効な非ゼロタイム・スタンプをレポートします。
- アプリケーションが作業単位を実行中の場合は、このエレメントがゼロをレポートします。
- アプリケーションがデータベースに初めて接続すると、このエレメントは **conn_complete_time** モニター・エレメントの値に設定されます。

新しい作業単位が開始すると、このエレメントの内容は、**prev_uow_stop_time** モニター・エレメントに移動します。

uow_throughput - 作業単位スループット : モニター・エレメント

1 秒単位の作業単位数で計測される、作業単位完了数。

表 2080. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 2081. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (メトリック文書に報告されます)	常に収集される
統計	event_wlstats (メトリック文書に報告されます)	常に収集される

使用法

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降の作業単位スループットを表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降の作業単位スループットを表します。

uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント

作業単位の存続時間の最高水準点 (ミリ秒)。

表 2082. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE

表 2082. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 2083. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	常に収集される
統計	event_scstats	常に収集される

使用法

このエレメントを使用すると、UOWTOTALTIME しきい値が有効であるかどうか、また、そのようなしきい値の構成方法を判別する助けになります。

サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、この最高水準点はワークロードが割り当てたサービス・クラス用に計算されます。アクティビティーのサービス・クラスを変更する作業アクション・セットのどのマッピングもこの最高水準点に影響しません。

update_sql_stmts 更新回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースに UPDATE ステートメントを発行した合計回数が含まれています。

表 2084. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、フェデレーテッド・サーバーまたはアプリケーションによりこのデータ・ソースに対して行われたデータベース・アクティビティーのレベルを判別できます。

このエレメントを使用すると、次の公式を使用して、フェデレーテッド・サーバーまたはアプリケーションによるこのデータ・ソースへの書き込みアクティビティーのパーセンテージも判別できます。

```
書き込みアクティビティー =
  (INSERT ステートメント + UPDATE ステートメント + DELETE ステートメント) /
  (SELECT ステートメント + INSERT ステートメント + UPDATE ステートメント +
  DELETE ステートメント)
```

update_time 更新応答時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの UPDATE に対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

表 2085. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

応答時間とは、フェデレーテッド・サーバーが UPDATE ステートメントをデータ・ソースにサブミットしてからデータ・ソースが UPDATE を処理したことをフェデレーテッド・サーバーに応答するまでの時間です。

使用法 このエレメントを使用すると、このデータ・ソースに対する UPDATE が処理されるのを待機するために生じた実際の時間を判別できます。この情報は、キャパシティー・プランニングおよびチューニングを行うときに便利です。

usage_list_last_state_change - 最後の状態変更のモニター・エレメント

usage_list_state モニター・エレメントの値が最後に変更された時刻を示すタイム・スタンプ。

表 2086. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

usage_list_last_updated - 使用リストの最終更新モニター・エレメント

特定のセクションが最後に更新された時刻を示すタイム・スタンプ。セクションは、**executable_id** と **mon_interval_id** モニター・エレメントの値によって表されます。

表 2087. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE

表 2087. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

usage_list_mem_size - 使用リスト・メモリー・サイズのモニター・エレメント

特定の使用リストに割り振られているメモリーの総量 (キロバイト単位)。

表 2088. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

usage_list_name - 使用リスト名のモニター・エレメント

使用リスト名。

表 2089. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

usage_list_num_references - 参照回数モニター・エレメント

特定のセクションが使用リストに追加されて以降、そのセクションが特定のオブジェクトを参照した合計回数。

表 2090. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

usage_list_num_ref_with_metrics - メトリックに関する参照回数のモニター・エレメント

特定のセクションが使用リストに追加され、統計が収集されて以降、そのセクションが特定のオブジェクトを参照した合計回数。

表 2091. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

usage_list_schema - 使用リスト・スキーマのモニター・エレメント

使用リストのスキーマの名前。

表 2092. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

usage_list_size - 使用リスト・サイズのモニター・エレメント

特定の使用リストに保持可能な項目の最大数。

表 2093. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

usage_list_state - 使用リストの状態のモニター・エレメント

特定の使用リストの状態。

可能な値は、以下のとおりです。

A アクティブ。

F アクティブ化に失敗しました。

I 非アクティブ。

P アクティブ化の処理中。

表 2094. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

usage_list_used_entries - 使用リストの使用された項目のモニター・エレメント

使用リストに現在含まれている項目の数。使用リストが 非アクティブ状態である場合、このモニター・エレメントは、この使用リストのモニターが最後にアクティブだったときに、使用リストに含まれていた項目の数を示します。

表 2095. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

usage_list_wrapped - 使用リスト折り返しインディケーターのモニター・エレメント

特定の使用リストが折り返されているかどうかを示すインディケーター。使用リストが満杯になった場合、デフォルトの動作として項目が折り返されます。つまり、最も古い項目は、最新の項目に置き換えられます。

可能な値は `Y` および `N` です。

表 2096. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE

user_cpu_time ユーザー CPU 時間 : モニター・エレメント

データベース・マネージャーのエージェント・プロセス、作業単位、またはステートメントで使用されたユーザー CPU 時間の合計 (秒およびマイクロ秒単位)。表に書き込むイベント・モニターの場合、このエレメントの値は、BIGINT データ・タイプを使用して、マイクロ秒単位で示されます。

ステートメント・モニター・スイッチまたはタイム・スタンプ・スイッチがオンになっていない場合は、このエレメントは収集されず、代わりに -1 が書き込まれます。

表 2097. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	常に収集される
トランザクション	event_xact	常に収集される

表 2097. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	常に収集される
アクティビティー	event_activity	常に収集される

使用法

このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

注: DB2 システムが統計を収集する際の細分度の違いのために、**total_exec_time** モニター・エレメントの値が、**system_cpu_time** モニター・エレメントの値と **user_cpu_time** モニター・エレメントの値の合計と等しくならないことがあります。この場合、**system_cpu_time** モニター・エレメントと **user_cpu_time** モニター・エレメントの合計の方が実際の実行時間の合計を正確に反映しています。

utility_dbname ユーティリティーで操作されるデータベース

ユーティリティーで操作されているデータベース。

表 2098. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_description ユーティリティー記述

ユーティリティーが実行している作業を簡潔に示す記述。例えば、rebalance 呼び出しに「Tablespace ID: 2」が含まれている場合、このリバランサーが ID 2 の表スペースに対して機能していることを示します。

このフィールドのフォーマットはユーティリティー・クラスに依存し、リリースごとに変更される可能性があります。

表 2099. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_detail - ユーティリティー詳細

このエレメントには、ユーティリティーが実行する処理の要旨が入っています。

表 2100. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILSTART	常に収集される

使用法

ユーティリティーが実行している作業の要旨。ユーティリティーに指定されている一部のオプションを含みます。例えば、REORG を起動するためのレコードには、部分的に再構成されたコマンド・ストリングが含まれます。これには、アクセス・モードなど、ユーティリティーが使用するさまざまなオプションの一部が含まれます。このフィールドのフォーマットはユーティリティーのタイプに依存し、リリースごとに変更される可能性があります。

utility_id ユーティリティー ID

ユーティリティー呼び出しに対応するユニーク ID。

表 2101. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 2102. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_invocation_id - ユーティリティー呼び出し ID

ユーティリティー呼び出しに対応するユニーク ID。

表 2103. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメントの収集レベル
変更履歴	changesummary utillocation utilphase utilstart utilstop	常に収集される

使用法

utility_invocation_id は、特定のユーティリティー呼び出しを一意的に識別するバイナリー・トークンです。 **utility_invocation_id** は、ユーティリティーを実行している全メンバーで同じです。 **utility_invocation_id** の固有性は、データベースの非アクティブ化、再活動化、およびメンバーのシャットダウンが行われても維持されます。このため、特定のユーティリティー呼び出しに対応するすべてのイベント・モニター・レコードを迅速に識別することができます。

utility_invoker_type - ユーティリティー呼び出し側タイプ

このエレメントは、ユーティリティーが起動された方法を説明しています。

表 2104. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

表 2105. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	utilstart	常に収集される

使用法

このエレメントを使用すると、ユーティリティーが呼び出された方法を判別できます。例えば、このエレメントを使用すると、ユーティリティーが DB2 によって自動的に呼び出されたか、またはユーザーによって呼び出されたかを判別できます。以下のリストにある、このエレメントの値は、sqlmon.h で定義されています。

API 定数	ユーティリティー
SQLM_UTILITY_INVOKER_USER	ユーティリティーはユーザーによって呼び出されました。
SQLM_UTILITY_INVOKER_AUTO	ユーティリティーは DB2 によって自動的に呼び出されました。

変更履歴イベント・モニターでは、このエレメントはユーティリティーが呼び出された方法を示します。

USER ユーティリティーはユーザーによって呼び出されました。

AUTO ユーティリティーは DB2 によって自動的に呼び出されました。

utility_operation_type - ユーティリティー操作のタイプ

ユーティリティー操作のタイプを指定します。

表 2106. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILSTART	常に収集される

使用法

変更履歴イベント・モニターでは、このエレメントには、開始されたユーティリティー・イベント (UTILITY_TYPE) に関する詳細が入っています。

UTILITY_TYPE が BACKUP の場合は、以下のいずれかです。

D 差分

I 増分

F 完全

UTILITY_TYPE が LOAD の場合は、以下のいずれか。

I 挿入

R 置換

S 再起動

T 終了

UTILITY_TYPE が MOVETABLE の場合は、以下のいずれか。

A 取り消し

C コピー

I 初期化

L クリーンアップ

M 移動

R 再生

S スワップ

V 検証

UTILITY_TYPE が REDISTRIBUTE の場合は、以下のいずれか。

A 打ち切り

C 続行

D デフォルト

T ターゲット・マップ

UTILITY_TYPE が REORG の場合は、以下のいずれか。

A すべての表索引の再編成

I 索引の再編成

N インプレース表再編成

R 表再利用エクステントの再編成

T 従来の表再編成

UTILITY_TYPE が RESTORE の場合は、以下のいずれかです。

A 自動増分

B 増分打ち切り

F 完全

M 手動増分

UTILITY_TYPE が ROLLFORWARD の場合は、以下のいずれかです。

E ログの最後

P 特定の時点

UTILITY_TYPE が RUNSTATS の場合は、以下のいずれかです。

A 表にあるすべての索引

I 索引

T 表

utility_phase_detail - ユーティリティー・フェーズ詳細

このエレメントは将来の利用のために予約済み。

表 2107. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILPHASE	常に収集される

utility_phase_type - ユーティリティー・フェーズ・タイプ

ユーティリティー・フェーズ・タイプを示します。

表 2108. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILPHASE	常に収集される

使用法

変更履歴イベント・モニターでは、utility_type エレメントが BACKUP の場合、フェーズ・タイプは以下です。

BACKUPTS
バックアップ表スペース

utility_priority ユーティリティー優先度

ユーティリティー優先度は、スロットルされたピアに関連したスロットル・ユーティリティーの相対的な重要度を指定します。優先度 0 は、ユーティリティーがスロットルされずに実行されることを意味します。非ゼロの優先度は 1 から 100 の範囲にする必要があります。 100 は最高の優先度、1 は最低の優先度です。

表 2109. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

表 2110. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	utilstart	常に収集される

utility_start_time ユーティリティー開始時刻

現在のユーティリティーがもともと呼び出された日付と時刻。

表 2111. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_start_type - ユーティリティー開始タイプ

このエレメントは、ユーティリティーが開始された方法を示します。

表 2112. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILSTART	常に収集される

使用法

変更履歴イベント・モニターの場合、ユーティリティー開始情報は以下のいずれかです。

- RESUME
- START

utility_state - ユーティリティー状態

このエレメントは、ユーティリティーの状態を示します。

エレメント ID

utility_state

エレメント・タイプ

情報

表 2113. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

使用法 このエレメントを使用して、アクティブ・ユーティリティーの状態を判別できます。以下のリストにある、このフィールドの値は、sqlmon.h で定義されています。

API 定数	説明
SQLM_UTILITY_STATE_EXECUTE	ユーティリティーは実行されています。
SQLM_UTILITY_STATE_WAIT	ユーティリティーは、進行を再開する前にイベントが発生するのを待機しています。
SQLM_UTILITY_STATE_ERROR	ユーティリティーは、エラーを検出しました。

utility_stop_type - ユーティリティー停止タイプ

このエレメントは、ユーティリティーが停止された方法を示します。

表 2114. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	UTILSTOP	常に収集される

使用法

変更履歴イベント・モニターでは、ユーティリティーは以下のいずれかの方法で停止されました。

- PAUSE
- STOP

valid セクション妥当性インディケーター : モニター・エレメント

動的 SQL ステートメント・セクションが有効かどうかを示します。 静的 SQL ステートメントの場合、このモニター・エレメントの値は常に Y です。

表 2115. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

使用法

このモニター・エレメントの有効値は Y および N です。システムが次回使用される時に、無効なセクションはシステムによって暗黙に準備されます。

utility_type ユーティリティー・タイプ

ユーティリティーのクラス。

表 2116. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

表 2117. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・エレメント収集レベル
変更履歴	changesummary utillocation utilphase utilstart utilstop	常に収集される

使用法

このエレメントの値は、`sqlmon.h` で定義された "SQLM_UTILITY_" の名前で始まる定数になります。

変更履歴イベント・モニターの場合、ユーティリティーのタイプは、以下のいずれかです。

- BACKUP
- LOAD
- MOVETABLE
- REDISTRIBUTE
- REORG
- RESTORE
- ROLLFORWARD
- RUNSTATS

valid セクション妥当性インディケーター : モニター・エレメント

動的 SQL ステートメント・セクションが有効かどうかを示します。 静的 SQL ステートメントの場合、このモニター・エレメントの値は常に Y です。

表 2118. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

使用法

このモニター・エレメントの有効値は Y および N です。システムが次回使用される時に、無効なセクションはシステムによって暗黙に準備されます。

vectored_ios - ベクトル化入出力要求数 : モニター・エレメント

ベクトル化入出力要求の数です。より厳密には、 DB2 がバッファー・プールのページ域へのページのプリフェッチを実行する回数です。

表 2119. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 2120. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール

使用法

このエレメントを使用すると、ベクトル化入出力の実行頻度を判別できます。ベクトル化入出力要求の数がモニターされるのは、プリフェッチの実行中だけです。

version モニター・データのバージョン

イベント・モニター・データ・ストリームを作成したデータベース・マネージャーのバージョン。

表 2121. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-

使用法

イベント・モニターが使用するデータ構造は、データベース・マネージャーのリリースによって異なっている場合があります。そのため、モニター・アプリケーションは、受信するデータを処理できるかどうかを判別するために、データ・ストリームのバージョンを確認する必要があります。

このリリースでは、このエレメントは API 定数の SQLM_DBMON_VERSION9_5 に設定されています。

virtual_mem_free - 空き仮想メモリーのモニター・エレメント

プロセスに割り振られていない、このホスト上の使用可能な仮想メモリーの量 (MB 単位)。

表 2122. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

virtual_mem_reserved - 予約済み仮想メモリーのモニター・エレメント

実行中のプロセスによって予約されている仮想メモリーの量 (MB 単位)。

表 2123. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

virtual_mem_total - 合計仮想メモリーのモニター・エレメント

このホスト上の使用可能な仮想メモリーの総量 (MB 単位)。

表 2124. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE

wl_work_action_set_id - ワークロード作業アクション・セット ID : モニター・エレメント

このアクティビティーがワークロード有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスが所属する作業クラス・セットに関連した作業アクション・セットの ID を示します。それ以外の場合、このモニター・エレメントは `0` の値を示します。

表 2125. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 2126. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このモニター・エレメントを **wl_work_class_id** モニター・エレメントと組み合わせて使用すると、アクティビティーのワークロード作業クラスが存在する場合にはそれを一意的に識別できます。

wl_work_class_id - ワークロード作業クラス ID : モニター・エレメント

このアクティビティーがワークロード有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 2127. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 2128. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このモニター・エレメントを **wl_work_action_set_id** モニター・エレメントと組み合わせて使用すると、アクティビティーのワークロード作業クラスが存在する場合にはそれを一意的に識別できます。

wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント

アクティビティーまたは接続が WLM しきい値によってキューに入れられた回数。

表 2129. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 2129. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2130. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

表 2130. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント

WLM キューイングしきい値の待機にかかった時間。この値はミリ秒単位で示されます。

表 2131. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE

表 2131. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2132. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

wlo_completed_total 完了したワークロード・オカレンスの合計 : モニター・エレメント

最後にリセットしてから完了するワークロード・オカレンスの数。

表 2133. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 2134. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-

使用法

このエレメントを使用すると、処理をシステムに移動させている特定のワークロードのオカレンスの数を判別できます。

work_action_set_id 作業アクション・セット ID : モニター・エレメント

この統計レコードが適用される作業アクション・セットの ID。

表 2135. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	常に収集される
統計	event_wcstats	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと一緒に使用すると、作業クラスの動作の分析をすることができます。

以下の条件が満たされている場合、このエレメントの値は 0 になります。

- このエレメントが、event_histogrambin 論理データ・グループでレポートされる。
- ヒストグラム・データが、処理クラスではないオブジェクトに関して収集される。

work_action_set_name 作業アクション・セット名 : モニター・エレメント

このイベントの一部として示された統計が関連付けられた作業アクション・セットの名前。

表 2136. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE

表 2137. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-
統計	event_wcstats	-

使用法

このエレメントと **work_class_name** エレメントを組み合わせて使用すると、統計がこのレコードに示されている作業クラスを一意的に識別したり、統計がこのレコードに示されているしきい値キューのドメインである作業クラスを一意的に識別できます。

work_class_id 作業クラス ID : モニター・エレメント

この統計レコードが適用される作業クラスの ID。

表 2138. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	常に収集される
統計	event_histogrambin	常に収集される

使用法

このエレメントを他の統計エレメントと一緒に使用すると、作業クラスの分析をすることができます。

以下の条件が満たされている場合、このエレメントの値は 0 になります。

- このエレメントが、event_histogrambin 論理データ・グループでレポートされる。
- ヒストグラム・データが、処理クラスではないオブジェクトに関して収集される。

work_class_name 作業クラス名 : モニター・エレメント

このイベントの一部として示された統計が関連付けられた作業クラスの名前。

表 2139. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE

表 2140. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-
統計	event_wcstats	-

使用法

このエレメントと **work_action_set_name** エレメントを組み合わせて使用すると、統計がこのレコードに示されている作業クラスを一意的に識別したり、統計がこのレコードに示されているしきい値キューのドメインである作業クラスを一意的に識別できます。

workload_id ワークロード ID : モニター・エレメント

ワークロードを一意的に識別する整数。

表 2141. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得	ACTIVITY METRICS BASE

表 2142. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本

表 2143. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	常に収集される
統計	event_wlstats	常に収集される
統計	event_histogrambin	常に収集される
アクティビティー	event_activity	常に収集される
しきい値違反	event_thresholdviolations	常に収集される

使用法

この ID を使用して、このアクティビティー、アプリケーション、ヒストグラム・ビン、またはワークロード統計レコードが所属するワークロードを一意的に識別します。

以下の条件が満たされている場合、このエレメントの値は 0 になります。

- このエレメントが、event_histogrambin 論理データ・グループでレポートされる。
- ワークロードではないオブジェクトのヒストグラム・データが収集される。

workload_name ワークロード名 : モニター・エレメント

ワークロードの名前。

表 2144. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 2145. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	常に収集される
統計	event_wlstats	常に収集される

使用法

統計イベント・モニターおよびワークロードの表関数において、ワークロード名は統計またはメトリックが収集されたり報告されるワークロードを識別します。 作業単位イベント・モニターおよび作業単位表関数の場合、ワークロード名は作業単位が関連付けられたワークロードを識別します。

ワークロード名を使用して、特定のワークロードに関する作業単位や情報のセットを識別します。

workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント

このアクティビティーが所属するワークロード・オカレンスの ID。

表 2146. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 2147. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される

使用法

これを使用すると、アクティビティーをサブミットしたワークロード・オカレンスを識別できます。

workload_occurrence_state - ワークロード・オカレンスの状態 : モニター・エレメント

ワークロード・オカレンスの状態。

表 2148. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE

表 2148. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

使用法

可能な値は以下のとおりです。

DECOUPLED
ワークロード・オカレンスにコーディネーター・エージェントが割り当てられていません (コンセントレーターのケース)。

DISCONNECTPEND
ワークロード・オカレンスはデータベースから切断中です。

FORCED
ワークロード・オカレンスは強制されました。

INTERRUPTED
ワークロード・オカレンスは中断されました。

QUEUED
ワークロード・オカレンス・コーディネーター・エージェントが、ワークロード管理キューイングしきい値によってキューに入れられています。パーティション・データベース環境では、この状態は、コーディネーター・エージェントがしきい値チケットを取得するために別のメンバーに対して RPC を行ったものの、まだ応答を受け取っていないことを示している可能性があります。

TRANSIENT
ワークロード・オカレンスは、サービス・スーパークラスにまだマップされていません。

UOWEXEC
ワークロード・オカレンスは要求を処理中です。

UOWWAIT
ワークロード・オカレンスはクライアントからの要求を待機中です。

x_lock_escals - 排他ロック・エスカレーション数 : モニター・エレメント

ロックが複数の行ロックから 1 つの排他表ロックにエスカレートされた回数。または行に対する排他ロックにより表ロックが排他ロックになった回数。

表 2149. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 2150. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される
トランザクション	event_xact	常に収集される

使用法

他のアプリケーションは排他ロックによって保留されているデータにアクセスすることができません。そのため、排他ロックはデータの並行性に影響を与える可能性があるため、それを追跡することは重要です。

アプリケーションが保留するロックの合計数がそのアプリケーションで使用可能なロック・リスト・スペースの最大量に達すると、ロックはエスカレートされます。使用可能なロック・リスト・スペースの量は、 **locklist** および **maxlocks** 構成パラメーターによって決まります。

1 つのアプリケーションが使用可能な最大ロック数に達して、エスカレートするロックがほかにない場合は、ほかのアプリケーションに割り振られているロック・リストのスペースが使用されます。ロック・リスト全体が満杯になるとエラーが起こります。

過度の排他ロック・エスカレーションが起こる場合の考えられる原因と対策については、『**lock_escals** モニター・エレメント』を参照してください。

共有ロックが十分にあるのに、アプリケーションは排他ロックを使用することがあります。共有ロックによってロック・エスカレーションの合計数を減らすことはできませんが、排他ロックのエスカレーションよりも共有ロックのエスカレーションのほうが望ましいと考えられます。

xda_object_pages XDA オブジェクト・ページ数

XML ストレージ・オブジェクト (XDA) データが消費するディスク・ページの数。

エレメント ID

xda_object_pages

エレメント・タイプ
情報

表 2151. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 2152. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	常に収集される

使用法 このエレメントを使用すると、特定の表中の XML ストレージ・オブジェクト (XDA) データが消費する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに XML ストレージ・オブジェクト・データが大きくなる比率を追跡できます。

xda_object_l_pages - XML ストレージ・オブジェクト (XDA) データ論理ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) データによって使用された、ディスク上の論理ページの数。

表 2153. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE

使用法

- この値は、オブジェクトに物理的に割り振られているスペース量より小さい場合があります。そのようになる可能性があるのは、TRUNCATE ステートメントの REUSE STORAGE オプションを使用した場合です。このオプションを指定すると、表に割り振られているストレージは引き続き割り振られますが、ストレージは空と見なされます。さらに、このモニター・エレメントの値は、オブジェクトに論理的に割り振られているスペース量よりも小さくなる可能性があります。その理由は、論理的に割り振られているスペースの合計には、量的にはわずかですが追加のメタデータが含まれているからです。

  オブジェクトの正確な論理サイズまたは物理サイズを取得するには、ADMIN_GET_TAB_INFO_V97 関数を使用します。この関数を使用すると、このモニター・エレメントに関して報告されるページ数とページ・サイズの積で得られる値よりも正確なオブジェクト・サイズに関する情報が得られます。

xid トランザクション ID

トランザクション・マネージャーが 2 フェーズ・コミットのトランザクションで生成した、 (すべてのデータベースにおいて) ユニークなトランザクション ID。

表 2154. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	作業単位

使用法 この ID を使用すると、トランザクション・マネージャーが生成したトランザクションと複数のデータベースに対して実行されたトランザクションを関連付けることができます。トランザクション・マネージャーに問題があったときに、 2 フェーズ・コミット・プロトコルが関与するデータベース・トランザクションとトランザクション・マネージャーが発信したトランザクションを対応させて、問題の診断に利用できます。

xmlid - XML ID モニター・エレメント

固有の文書 ID。この ID は、以下のように導出します。
<event_header>_<event_id>_<event_type>_<event_timestamp>_<partition>

表 2155. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
EVMON_FORMAT_UE_TO_TABLES プロシージャー - XML 文書のリレーショナル表への移動	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
EVMON_FORMAT_UE_TO_XML 表関数 - 未フォーマット・イベントの XML への変換	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する

表 2156. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング		
パッケージ・キャッシュ		

xquery_stmts - 試行された XQuery ステートメント

アプリケーションまたはデータベースに対して実行される XQuery ステートメントの数。

エレメント ID
xquery_stmts

エレメント・タイプ
カウンター

表 2157. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 2157. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 2158. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される
接続	event_conn	常に収集される

使用法 このエレメントを使用すると、固有の XQuery 言語要求のアクティビティーを測定することができます。これには xmlquery、xmltable、または xmlexist などの組み込み XQuery 言語要求は含まれません。

第 3 部 DB2 pureScale 環境におけるモニター

IBM DB2 pureScale Feature には、堅固で、可用性の高いデータベース処理環境が備わっています。DB2 pureScale インスタンスで 1 つ以上のホスト・システムの操作で問題が生じる場合であっても、通常はデータへのアクセスを中断しないでも対応が可能です。

皮肉にも、DB2 pureScale 環境におけるこの高可用性という特性は、パフォーマンスが最適でなくなることにつながる可能性のある問題を隠してしまうことがあります。DB2 pureScale 環境の特定の要素をモニターすると、そうした問題を把握して対処するのに役立ちます。

例えば、ハードウェア障害が原因となって、メンバーまたはクラスター・キャッシング・ファシリティー (CF とも呼ばれます) が別のホストに繰り返しフェイルオーバーするという事態が生じているとします。ただし、多くの場合、リカバリーは自動的に行われるので、問題に気付くことはないかもしれません。問題が検出されずに修正されないと、DB2 pureScale 環境が持っている潜在的パフォーマンスを完全には理解できません。

このため、DB2 pureScale インスタンスのある程度の操作モニターを継続的に行うことをお勧めします。これにより、以下のような質問に対する回答を得る助けになります。

- DB2 pureScale インスタンスのすべてのコンポーネントが実行されているか?
- メンバーまたは CF に障害はあった場合、正常に再始動できたか?
- 優先 1 次ホストで CF が実行されているか? あるいは、他のホストにフェイルオーバーされたか?
- 他の CF は、1 次 CF で障害が発生した場合に引き継ぐ準備が整っているか?

DB2 pureScale 環境の全体的な状況を把握したい場合、最初に行うべきなのは、ご使用のインスタンスにおけるホスト、メンバー、CF の操作状況を確かめることです。それらのエンティティーそれぞれについてレポートされている状態とアラートの情報を調べると、ご使用の DB2 pureScale インスタンスの機能状態に関する全体像を把握できます。

DB2 モニタリング・インフラストラクチャーを使用すると、DB2 pureScale インスタンスの全体的な状態を調べられるだけでなく、各モニター・エレメントを検査して、DB2 pureScale インスタンスの特定の面に関する情報も得られます。この情報を使用すると、全体的なシステム・パフォーマンスを低下させている可能性のある構成やアプリケーション設計について一層把握しやすくなります。

第 12 章 DB2 pureScale インスタンスの状況モニタリング

DB2 pureScale インスタンスのコンポーネントの全体的な状況を表示すると、どうすれば効率的に機能させられるかという全体像を把握するのに役立ちます。この情報は、表関数および管理ビューを使用すると表示できます。

また、DB2 コマンド行プロセッサー (CLP) コマンドとシステム・コマンドを使用して、インスタンスの操作状況を表示することもできます。

こうした表関数、ビュー、およびコマンドによって戻される情報から、ご使用の DB2 pureScale インスタンスの状態に関する全体的な概要を知ることができます。例えば、これらのインターフェースを使用すると、以下のような質問の回答を得られます。

- DB2 pureScale インスタンスを構成しているホストがアクティブかどうか?
- クラスター・キャッシング・ファシリティーまたはメンバーのサーバーとして機能しているホストは?
- メンバーおよびクラスター・キャッシング・ファシリティーの両方として機能しているホストがあるか?
- 複数のメンバーを実行しているホストがあるか?
- 優先ホーム・ホストで実行されているメンバーか?
- 複数のクラスター・キャッシング・ファシリティーによる構成の場合、1 次サーバーとして機能しているのはどれか? 他の 1 次以外のクラスター・キャッシング・ファシリティーの現在の状態は PEER、それとも CATCHUP か?
- インスタンスを構成しているメンバーの現在の *状態* は? 例えば、**db2stop** コマンドを使用して停止されたメンバーがあるか? あるいは、ホーム・ホストにフェイルバックするのを現在待機しているメンバーがあるか?
- いずれかのクラスター・キャッシング・ファシリティーまたは メンバーのサーバーが、調査が必要なアクティブ・アラート を示しているか?

こうした多くのインターフェースによって戻される情報には、*状態* とアラート についての情報が含まれています。ホスト、メンバー、またはクラスター・キャッシング・ファシリティーの状態は、当該オブジェクトの現在の操作能力を示しています。例えば、メンバーの状態としては、STARTED、STOPPED、RESTARTING、WAITING_FOR_FAILBACK、ERROR、UNKNOWN が考えられます。

アラートは、さらに調査が必要な対象があるかどうかを示すのに使用されます。インスタンスの 1 つ以上のクラスター・キャッシング・ファシリティーまたはメンバーでアラートが発生する場合、調査が必要が問題が生じている可能性があります。

DB2 pureScale インスタンスの状況情報を取得するためのインターフェース

DB2 pureScale インスタンスの種々のコンポーネントに関する全体的な状況を把握するには、多様な管理ビュー、表関数、CLP コマンドから選択可能です。

また、サーバー・コンポーネントが稼働していないときに役立つ、システム・コマンド・プロンプトまたはシェルから使用可能なコマンドもあります。

以下の方法で、ご使用の DB2 pureScale インスタンスについての状態およびアラート情報を表示できます。

- 『表関数および管理ビュー』
- 1751 ページの『CLP コマンド』
- 1751 ページの『システム・プロンプト (シェル) コマンド』.

表関数および管理ビュー

表関数には、システムに関する情報を取得するための柔軟なインターフェースが備わっています。ほとんどの表関数では、戻される情報の範囲を絞り込むためのパラメーターを受け入れます。例えば、特定のホストに関する情報を表示したい場合などがあります。

管理ビューでは、システム情報に素早く簡単にアクセスできます。表関数とは異なり、照会の範囲を絞り込むために管理ビューにパラメーターを渡すことはできません。通常、システム内でこのビューに関するすべてのオブジェクト (例えば、ホスト、クラスター・キャッシング・ファシリティー (CFとも呼ばれます)、または メンバーなど) に関する情報が戻されます。ただし、SQL を使用すると、いつでも管理ビューの出力をフィルター操作できます。

以下の表関数とその対応する管理ビューによって、DB2 pureScale インスタンスについての全体的な状況に関する情報が戻されます。

表 2159. DB2 pureScale インスタンスのコンポーネントに関する状況情報を表示する表関数と管理ビュー

インターフェース	説明
DB2_GET_CLUSTER_HOST_STATE 表関数 DB2_CLUSTER_HOST_STATE 管理ビュー	これらのインターフェースは、DB2 pureScale インスタンスを構成するホストについての基本情報を提供します。ホストのリストと、関連する状態情報が戻ります。
DB2_GET_INSTANCE_INFO 表関数 DB2_MEMBER 管理ビュー DB2_CF 管理ビュー	これらのインターフェースでは、DB2 pureScale インスタンスに関する詳細情報が提供されます。各ホストがインスタンスで果たす役割 (クラスター・キャッシング・ファシリティーまたはメンバー) に関する情報、それぞれのクラスター・キャッシング・ファシリティーまたは メンバーがホーム・ホストで実行されているかどうか、および各ホストの接続情報が戻されます。
DB2_INSTANCE_ALERTS 管理ビュー	このインターフェースでは、DB2 pureScale インスタンスにおけるアラート情報が提供されます。

注: 前述のそれぞれのインターフェースは、DB2 pureScaleインターフェースでも、その他の DB2 インスタンスでも使用できます。それぞれが戻す結果は異なる場合があります。例えば、DB2_MEMBER 管理ビューはどちらのタイプのインスタンスにも使用できますが、DB2 pureScale 環境以外のインスタンスの場合、返される情報には状態やアラートに関する情報は含まれません。関数やビューから返される結果を、照会しているインスタンスのタイプに応じて解釈する必要があります。詳細については、それぞれの表関数や管理ビューの参照トピックをご覧ください。

CLP コマンド

以下のコマンドを、DB2 コマンド行プロセッサー (CLP) から使用できます。

表 2160. DB2 pureScale インスタンスのコンポーネントに関する状況情報を表示する CLP コマンド

CLP コマンド	説明
LIST INSTANCE	このコマンドは、メンバー、ホスト、およびクラスター・キャッシング・ファシリティーの状態情報を戻します。
LIST INSTANCE SHOW DETAIL	このコマンドは LIST INSTANCE コマンドの拡張機能であり、メンバー、ホスト、クラスター・キャッシング・ファシリティーに関するパーティション番号および接続情報を含む追加情報を戻します。

システム・プロンプト (シェル) コマンド

以下のコマンドを、システム・プロンプトまたはシェル・プロンプトから使用できます。

表 2161. DB2 pureScale インスタンスのコンポーネントに関する情報を表示するシステム・プロンプト (シェル) コマンド

コマンド	説明
db2instance -list	このコマンドは、DB2 pureScale インスタンスにおけるメンバー、ホスト、クラスター・キャッシング・ファシリティーに関する状況情報を戻します。その時点でデータベース接続がない場合でも、あるいはインスタンスが停止している場合であっても、このコマンドは使用可能です。インスタンスが停止している場合、**db2instance -list** コマンドをクラスター・マネージャーとともに使用すると、DB2 pureScale インスタンスにおけるホストについての情報がレポートされます。メンバー、またはクラスター・キャッシング・ファシリティーのどちらかだけに対する出力を制限するために使用可能なオプションが幾つかあります。
db2cluster -list *options*	このコマンドを使用すると、DB2 pureScale インスタンスに関する情報を表示できます。このコマンドの場合、選択可能な幾つかの追加オプションがあります。**-list** オプションを指定する場合、コマンド出力に含める内容を指定する追加オプションも指定しなければなりません。

メンバーおよびクラスター・キャッシング・ファシリティーの状態とアラートの値

DB2 pureScale 環境におけるコンポーネントの状況を照会するために使用可能な表関数、管理ビュー、およびコマンドの多くは、*状態* とアラート についての情報を戻します。

ホスト、メンバー、 クラスター・キャッシング・ファシリティー (CF とも呼ばれます) の*状態* は、作動状況を示します。 ホスト、メンバー、 CF のアラート は、調査や介入が必要となる場合がある問題が生じていることを示します。

ホスト、メンバー、 クラスター・キャッシング・ファシリティーの状態

DB2 pureScale 環境におけるコンポーネントの状況を照会するために使用可能な表関数、管理ビュー、およびコマンドの多くによって、状態情報が戻されます。各コンポーネントの状態に関して考えられる値を、表 2162 に示します。

表 2162. ホスト、メンバー、クラスター・キャッシング・ファシリティーに関して考えられる状態

コンポーネント	考えられる状態	説明
ホスト	ACTIVE	ホストは使用可能です。これは、ホスト・システムが稼働中で、TCP/IP **ping** コマンドなどのオペレーティング・システム・コマンドやネットワーキング・コマンドに応答できることを意味します。
	INACTIVE	ホストは使用できません。これは、ホスト・システムが稼働していないので、システム・コマンドを使用したり、システム・コマンドに対して応答したりできないことを意味します。この状態になる原因としては、ホストの電力低下や、接続またはネットワーキングの問題など様々なことが考えられます。

表 2162. ホスト、メンバー、クラスター・キャッシング・ファシリティーに関して考えられる状態 (続き)

コンポーネント	考えられる状態	説明
クラスター・キャッシング・ファシリティー (CF)	STOPPED	CF が、管理者によって通常シャットダウンの一部として **db2stop** コマンドを使用して手動で停止されました。
	RESTARTING	CF が、**db2start** コマンドによって、または CF 障害の後に、開始処理中です。
	BECOMING_ PRIMARY	CF が開始された後、他の CF がインスタンスで 1 次 CF の役割を担っていない場合には、この役割を果たそうとします。
	PRIMARY	CF が、1 次 CF として通常どおり作動しています。
	CATCHUP (*n*%)	バックアップ CF が最初に開始されると、1 次 CF からの情報がまったく含まれません。 CATCHUP 状態の場合、バックアップ CF は 1 次 CF からの関連情報すべてのコピーを取得する処理中です。この情報によって、1 次 CF で障害が発生した場合に 1 次 CF としての役割を引き受けることが可能になります。 *n*% は、バックアップ CF における 1 次 CF からの情報のコピー・プロセスの進行状況を示します。このコピー・プロセスが完了すると、バックアップ CF は PEER 状態に移行します。 **注:** 非 1 次 CF の状態を **db2instance -list** コマンドを使用して表示すると、データベースに接続するまでは **CATCHUP** 状態になります。最初の接続が行われると、1 次 CF からのデータ・コピー処理が開始されます。
	PEER	バックアップ CF が、1 次 CF で障害が発生した場合に 1 次 CF としての役割を引き継ぐ準備が整っていることを示します。バックアップ CF が PEER 状態の間、二重化は継続されます。
	ERROR	DB2 クラスター・サービスが、CF を自動的に再始動できませんでした。CF が ERROR 状態を示す場合、ALERT フィールドは必ず YES になります。これは、管理者によって調査や介入が必要であることを示します。アラートがクリアされない限りは、DB2 クラスター・サービスは、ERROR 状態になった CFの再始動を試行しません。 また、CF への接続を確立して状態を照会できない場合にも、ERROR 状態が生じます。この場合、ALERT フィールドは YES になりません。問題が一時的である可能性があるからです。

表 2162. ホスト、メンバー、クラスター・キャッシング・ファシリティーに関して考えられる状態 (続き)

コンポーネント	考えられる状態	説明
メンバー	STARTED	インスタンスでメンバーが開始され、通常どおりに作動しています。すべてのデータベースが整合のある状態にあり、メンバーがデータベースへの接続を受け入れる準備が整ったか、既に受け入れています。メンバーで障害が発生し再開した場合には、プロセス・モデルを開始することは可能ですが、データベースのクラッシュ・リカバリーはまだ完了していません。いずれかのメンバーから **LIST UTILITIES SHOW DETAIL** コマンドを使用して、リカバリーの進行状況をモニターしてください。
	STOPPED	管理者によって通常シャットダウンの一部として、**db2stop** コマンドを使用してメンバーが手動で停止されました。
	RESTARTING	メンバーは開始または再始動の処理中です。現行ホストがホーム・ホストと同じ場合、ローカル・メンバーが再始動されています。現行ホストとホーム・ホストが異なる場合には、メンバーは現行ホストにフェイルオーバーし、light モードで再始動しています。
	WAITING_FOR_ FAILBACK	このメンバーのプロセス・モデルは、light モードで現行ホストにおいて正常に再始動されました。メンバーはホーム・ホストが使用可能になるのを待機していて、使用可能になった時点で、ホーム・ホストにフェイルバックします。任意のアクティブ・メンバーから、**LIST UTILITIES** コマンドに SHOW DETAIL オプションを指定して使用し、リカバリーの進行状況をモニターし、クラッシュ・リカバリーがすべてのデータベースで完了したかどうかを確認してください。メンバーは、新規の接続を受け入れませんし、トランザクションも処理しません。未確定トランザクションが存在する場合があります。
	ERROR	DB2 クラスター・サービスが、DB2 pureScale インスタンスのホーム・ホストまたはその他のホストで、メンバーを自動的に再始動できませんでした。メンバー が ERROR 状態を示す場合、ALERT フィールドは必ず YES に設定されます。これは、管理者による調査や介入が必要であることを示します。アラートがクリアされない限りは、DB2 クラスター・サービスは、ERROR 状態になった メンバーの再始動を試行しません。

ホスト、メンバー、 クラスター・キャッシング・ファシリティーのアラート

DB2 pureScale 環境におけるコンポーネントの状況を照会するコマンドは、状態情報を戻すだけでなく、アラート情報も戻します。すべてのコンポーネントに関して、考えられるアラートの値は、YES または NO です。一般に、アラートの値 NO は、通常どおりに実行されていることを示します。アラートの値が YES の場合、手操作による介入が必要となる場合がある問題が存在することを示します。場合によっては、アラート条件が一時的で、ホストがリブートするなどして、アラート・フィールドが自動的にクリアされることもあります。その他の場合には、管理者が問題を解決し、**db2cluster** コマンドに -clear -alert オプションを使用してアラート・フィールドを手動でリセットするまでは、アラート・フィールドはそのままです。

状況情報の解釈

状況情報についてホスト、メンバー、またはクラスター・キャッシング・ファシリティーに照会を実行すると、システムは、ご使用の DB2 pureScale 環境の種々のコンポーネントの状況を通知する状態およびアラート情報を表示します。通常、問題が発生すると、システム内で生じている事柄を把握するために状態とアラートの両方を調べる必要があります。

ホスト、メンバー、クラスター・キャッシング・ファシリティー (CF とも呼ばれます) の *状態* は、作動状況を示します。すべて通常どおりに作動している場合、ホスト、メンバー、およびクラスター・キャッシング・ファシリティー (CF とも呼ばれます) の状態に関してレポートされる様々な値によって、システムの状況の概要が分かります。例えば、メンバーでの状態が RESTARTING、または WAITING_FOR_FAILBACK の場合、問題があることは示されていません。メンバーが新しいホストにフェイルオーバーした理由、ホーム・ホストで再始動した理由 (例えば、保守のためにホストをオフラインにした場合など) について、考えられるもっともな原因がいくつかある可能性があります。メンバーが再三にわたりフェイルオーバーする場合には、さらなる調査が必要な問題が生じている場合があります。

ホスト、メンバー、 CF の *アラート* は、調査や介入が必要となる場合がある問題が生じていることを示します。特定のシステム・コンポーネントの状態に応じてアラートを調べると、問題の原因に関する追加情報を明らかにできます。以下のセクションでは、ホスト、メンバー、クラスター・キャッシング・ファシリティーに関して生じる可能性のある状態とアラート情報のさまざまな組み合わせについて、および状態とアラートの組み合わせによってそれぞれをどのように解釈するかについて略述しています。

要確認: この情報に関してレポートするインターフェースによって戻される状態情報とアラート情報が完全に揃うかどうかは、以下の要素によって異なります。

- 表関数、管理ビュー、またはコマンドを実行しているインスタンスのタイプ (例えば、DB2 pureScale インスタンスや、その他の DB2 インスタンスなど)
- 該当インスタンスにおいてサポートされているクラスター・マネージャーが導入されているかどうか。 DB2 pureScale Feature デプロイメントすべてにおいて、クラスター・マネージャーが使用されます。

詳しくは、 1759 ページの『データ共有とDB2 pureScale 環境以外の環境のレポート作成の相違点』を参照してください。

ホストの状況

さまざまなインターフェースを使用して、DB2 pureScale 環境におけるホストについての情報を表示できます。そうしたインターフェースの 1 つに、DB2_CLUSTER_HOST_STATE 管理ビューがあります。例えば、以下の SQL 照会について考慮してみましょう。

```
SELECT varchar(HOSTNAME,10) AS HOST,
       varchar(STATE,8) AS STATE,
       varchar(INSTANCE_STOPPED,7) AS STOPPED,
       ALERT
FROM SYSIBMADM.DB2_CLUSTER_HOST_STATE
```

前述の SQL ステートメントを実行すると、出力は以下のようになります。

```
HOST       STATE    STOPPED ALERT
---------- -------- ------- --------
HOSTD      ACTIVE   NO      NO
HOSTB      ACTIVE   NO      NO
HOSTA      ACTIVE   YES     NO
HOSTC      ACTIVE   NO      NO

  4 record(s) selected.
```

(前述の例の場合、STOPPED 列は、管理ビューが戻す INSTANCE_STOPPED 列に相当します。)

状態、instance_stopped、およびアラートの各列の値は、その時点での条件によって異なる値を取る可能性があります。表 2163 に、可能な値について要約しています。

表 2163. DB2 pureScale インスタンスのホスト・システムで考えられる状態、instance_stopped、およびアラートの組み合わせ

STATE	INSTANCE_ STOPPED	ALERT	説明
ACTIVE	NO	NO	ホストはアクティブで、通常どおりに作動しています。
		YES	ホストはアクティブですが (つまり、システム・コマンドに応答しますが)、DB2 pureScale インスタンスに参加するのを妨げる問題がある可能性があります。例えば、ファイル・システム上の問題、ネットワーク通信の問題、またはアイドル・プロセスがあり、DB2 pureScale Feature がフェイルオーバーを実行する必要があるものの実行できない場合があります。
	YES	NO	ホストはアクティブです。管理者が **db2stop instance on** *hostname* コマンドを使用して、このホストにおいてインスタンスを明示的に停止しました。
		YES	ホストはアクティブですが、ホストに関してアラートが存在し、クリアされません。管理者がインスタンスを明示的に停止しました。
INACTIVE	NO	NO	該当なし。 INSTANCE_STOPPED と ALERT の両方が NO に設定されている場合には、ホストを INACTIVE にすることはできません。
		YES	ホストがシステム・コマンドに応答しません。インスタンスは管理者によって明示的に停止されませんでしたが、アラートが存在します。状況情報のこの組み合わせは、ホストの異常シャットダウンを示します。そのようなシャットダウンは、ホストにおける電源障害などによって生じる可能性があります。
	YES	NO	インスタンスが管理者によって停止された場合の通常状態です。状況情報のこのような組み合わせは、ソフトウェアの更新インストールのためにホストがオフラインになった場合に生じることがあります。
		YES	ホストがシステム・コマンドに応答しません。ホストに関してアラートが存在し、クリアされていませんが、管理者によりインスタンスが明示的に停止されました (つまりシステムは、異常シャットダウンはしませんでした)。

ヒント: DB2_INSTANCE_ALERTS 管理ビューを使用すると、アラートの詳細を表示できます。

メンバーの状況

幾つかの異なるインターフェースを使用して、メンバーの状態とアラートを表示できます。そうしたインターフェースの 1 つに、DB2_MEMBER 管理ビューがあります。DB2_MEMBER 管理ビューは、DB2 pureScale インスタンスにおけるメンバーの状況情報を表示します。以下は、この管理ビューを使用してメンバー状況を取得する方法の例です。

```
SELECT ID,
       varchar(STATE,21) AS STATE,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       ALERT
FROM SYSIBMADM.DB2_MEMBER
```

状態およびアラートの列の値は、その時点での条件によって異なる値を取る可能性があります。表 2164 に、可能な値について要約しています。

表 2164. DB2 pureScale インスタンスにおけるメンバーで考えられる状態とアラートの組み合わせ

STATE	ALERT	説明
STARTED	NO	メンバーはインスタンスで通常どおりに開始されて、作動しています。
	YES	メンバーは、インスタンスで開始されました。ただし、いずれかの時点で、別のホストにフェイルオーバーする試行が正常に行われませんでした。フェイルオーバーする試行が失敗したので、メンバーは他のホストに正常にフェイルオーバーされたか、ホーム・ホストにフェイルバックされました。メンバーがホーム・ホストで実行されている場合、正常に実行されています。ゲスト・ホストで実行されている場合には、light モードで実行されています。何らかの方法で、アラートを調査し、生じた事柄を見極めてください。
STOPPED	NO	管理者が **db2stop** コマンドを使用して、メンバー を停止しました。
	YES	管理者が **db2stop** コマンドを使用して、メンバー を停止しましたが、アラート・フィールドはクリアされていません。
RESTARTING	NO	メンバー が開始されています。
	YES	メンバー が開始されています。ただし、いずれかの時点で、ホーム・ホストでメンバーを開始する試行、または別のホストにフェイルオーバーする試行が正常に行われませんでした。アラート・フィールドがまだクリアされていません。
WAITING_FOR_ FAILBACK	NO	メンバーが light モードでゲスト・ホストにおいて実行されていて、ホーム・ホストにフェイルバックするのを待機しています。ホーム・ホストの状況を調べて、メンバーがホーム・ホストにフェイルバックするのを妨げるものがあるかどうか (例えば、ネットワーク・アダプター障害など) を確認できます。
	YES	ホーム・ホストでメンバーを再始動する試行が失敗したか、自動フェイルバックが無効になっているか、クラッシュ・リカバリーが失敗した可能性があります。メンバーが自動的にホーム・ホストにフェイルバックできるようにするには、問題を解決してアラートを手動でクリアする必要があります。自動フェイルバックが無効になっている場合は、アラートを手動でクリアし、**db2cluster** コマンドを使用して自動フェイルバックを有効にします。
ERROR	YES	DB2 クラスター・サービスが、いずれのホストにおいてもメンバーを開始できませんでした。インスタンスを再始動するには、その前に問題を解決し、手動でアラートをクリアする必要があります。

ヒント: DB2_INSTANCE_ALERTS 管理ビューを使用すると、アラートの詳細を表示できます。

クラスター・キャッシング・ファシリティーの状況

DB2_GET_INSTANCE_INFO 表関数を使用して、DB2 pureScale インスタンスにおけるメンバーの状況情報を取得します。この表関数を使用する利点の 1 つとしては、パラメーターをこの関数に渡すことにより、戻される結果範囲を絞り込むことができます。例えば、DB2 pureScale インスタンスにおける CF の情報を取得するには、以下のような照会を作成できます。

```
SELECT ID,
       varchar(STATE,17) AS STATE,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       ALERT
FROM TABLE(DB2_GET_INSTANCE_INFO(NULL,'','','CF',NULL))
```

状態およびアラートの列の値は、その時点での条件によって異なる値を取る可能性があります。表 2165 に、可能な値について要約しています。

表 2165. DB2 pureScale インスタンスにおけるクラスター・キャッシング・ファシリティーで考えられる状態とアラートの組み合わせ

STATE	ALERT	説明
STOPPED	NO	クラスター・キャッシング・ファシリティー (CF とも呼ばれます) が **db2stop** コマンドにより手動で停止されました。
	YES	CF を 1 次 CF にする試行は正常に行われませんでした。クラスター・キャッシング・ファシリティーは、管理者が **db2stop** コマンドを使用してインスタンス内で手動で停止されました。
RESTARTING	NO	**db2start** コマンドを使用して、または 1 次 CF で障害が発生した結果として、CF が再始動されています。
	YES	CF が再始動されていますが、CF が 1 次役割を引き受ける試行が既に失敗したために保留アラートがあり、手動でクリアする必要があります。
BECOMING_ PRIMARY	NO	インスタンスで既に実行されている 1 次 CF が他にない場合には、該当の CF が 1 次CF の役割を引き受けます。
	YES	該当なし。 CF はアラート条件セットによって 1 次役割を引き受けることはできません。
PRIMARY	NO	CF が 1 次 CF の役割を引き受け、通常どおりに作動しています。
	YES	該当なし。 CF が、アラート条件セットによって 1 次 CF として作動できません。
CATCHUP(*n*%)	NO	この非 1 次 CF は、PEER モードで作動するために必要な、1 次 CF からの情報をコピー処理中です。 **注:** 非 1 次 CF の状態を **db2instance -list** コマンドを使用して表示すると、データベースに接続するまでは **CATCHUP** 状態になります。最初の接続が行われると、1 次 CF からのデータ・コピー処理が開始されます。
	YES	この非 1 次 CF は、PEER モードで作動するために必要な、1 次 CF からの情報をコピー処理中です。この CF が 1 次役割を引き受ける試行が既に失敗したために保留アラートがあり、手動でクリアする必要があります。

表 2165. DB2 pureScale インスタンスにおけるクラスター・キャッシング・ファシリティーで考えられる状態とアラートの組み合わせ (続き)

STATE	ALERT	説明
PEER	NO	この非 1 次 CF は、現行の 1 次 CF で障害が発生した場合に 1 次 CF の役割を担う準備が整っています。
	YES	この非 1 次 CF は、現行の 1 次 CF で障害が発生した場合に 1 次 CF の役割を担う準備が整っています。この CF が 1 次役割を引き受ける試行が既に失敗したために保留アラートがあり、手動でクリアする必要があります。
ERROR	YES	CF は、インスタンスのいずれのホストにおいても開始できませんでした。インスタンスを再始動するには、その前に問題を解決し、手動でアラートをクリアする必要があります。

ヒント: DB2_INSTANCE_ALERTS 管理ビューを使用すると、アラートの詳細を表示できます。

データ共有とDB2 pureScale 環境以外の環境のレポート作成の相違点

ホスト、メンバー、および クラスター・キャッシング・ファシリティーの状況データをレポートするさまざまな表関数、管理ビュー、コマンドはいずれも、DB2 pureScale インスタンス以外でも使用できます。ただし、こうしたインターフェースによって戻される結果は、DB2 pureScale インスタンスに表示される結果とは異なる場合があります。

サポート対象のクラスター・マネージャー (CM) を用いたクラスター・ファイル・システムを使用する構成 (「統合高可用性」や「統合 HA」とも呼ばれる構成) では、これらの状況レポート作成インターフェースのほとんどによって戻される結果は、DB2 pureScale インスタンスに表示される結果と似ています。例外は、DB2_GET_CLUSTER_HOST_STATE 表関数または DB2_CLUSTER_HOST_STATE 管理ビューを使用して、インスタンスのホストに関する情報を取得する場合です。統合 HA を用いたDB2 pureScale インスタンス以外の場合、こうしたインターフェースのいずれも INSTANCE_STOPPED 列を返しません。例えば、DB2_CLUSTER_HOST_STATE 管理ビューを使用した照会結果は、図 20 に示されている結果のようになります。

```
HOSTNAME STATE  INSTANCE_STOPPED ALERT
-------- ------ ---------------- -----
HOSTA    ACTIVE -                NO
HOSTB    ACTIVE -                NO
HOSTC    ACTIVE -                NO
HOSTD    ACTIVE -                NO
```

図 20. クラスター・マネージャーを用いるDB2 pureScale インスタンス以外の DB2_CLUSTER_HOST_STATE 管理ビューによって返される結果

別の例外は、クラスター・キャッシング・ファシリティーの状況に関して特定のレポートをするインターフェースです。DB2 pureScale 環境以外の場合、クラスター・キャッシング・ファシリティーが存在しないので状況はレポートされません。例えば、DB2 pureScale 環境以外の環境では、DB2_CF 管理ビューは以下のような

結果を返します。

```
ID     CURRENT_HOST        STATE      ALERT
------ ------------------- ---------- -----

     0 record(s) selected.
```

図 21. DB2 pureScale インスタンス以外で DB2_CF 管理ビューによって返される結果

状況レポート作成インターフェースが CM なしでインスタンスで使用されると、状況情報やアラート情報はまったく戻されません。例えば、DB2_CLUSTER__HOST_STATE 管理ビューを使用する照会結果は図 22 のようになります。

```
HOSTNAME STATE  INSTANCE_STOPPED ALERT
-------- ------ ---------------- -----
HOSTA    -      -                -
HOSTB    -      -                -
HOSTC    -      -                -
HOSTD    -      -                -

     4 record(s) selected.
```

図 22. クラスター・マネージャーを用いないDB2 pureScale インスタンス以外の DB2_CLUSTER_HOST_STATE 管理ビューによって返される結果

例: ホスト、メンバー、クラスター・キャッシング・ファシリティーの状況の表示

以下のトピックには、DB2 pureScale インスタンスのコンポーネントの状況を表示するための種々のインターフェースの使用例が含まれています。

DB2 pureScale インスタンスにおけるホストの状況情報の表示

DB2 pureScale インスタンスにおけるホストの全体的な状況を示す基本情報を取得できます。この情報によって、ホストがアクティブかどうか、インスタンスがホスト上で実行されているかどうか、および調査が必要なアラートが存在するかどうかが分かります。

このタスクについて

インスタンスの全体的な状況を表示するには、DB2 pureScale インスタンスにおけるホストの状況を表示することから始めます。

この状況を取得する 1 つの方法は、DB2_CLUSTER_HOST_STATE 管理ビューを使用することです。このビューは、インスタンスにおけるすべてのホストの状況情報を戻します。また、以下のインターフェースを使用しても、ホストの状況情報を取得できます。

- DB2_GET_CLUSTER_HOST_STATE 表関数。この表関数はホスト ID をパラメーターとして受け入れるので、この方法は特定のホストの状況を照会する場合に役立ちます。
- **LIST INSTANCE** コマンド。

- **db2instance** コマンド (**-list** パラメーターを指定)

手順

DB2 pureScale インスタンス内のホストの状況を表示するには、以下のようにします。

1. DB2_CLUSTER_HOST_STATE 管理ビューまたは DB2_GET_CLUSTER_HOST_STATE 表関数を使用して SQL ステートメントを作成します。 以下の例では、管理ビューを使用しています。

```
SELECT varchar(HOSTNAME,10) AS HOST,
       varchar(STATE,8) AS STATE,
       varchar(INSTANCE_STOPPED,7) AS STOPPED,
       ALERT
FROM SYSIBMADM.DB2_CLUSTER_HOST_STATE
```

2. この照会を実行します。

タスクの結果

前述の SQL ステートメントを実行すると、出力は以下のようになります。

```
HOST       STATE    STOPPED ALERT
---------- -------- ------- --------
HOSTD      ACTIVE   NO      NO
HOSTB      ACTIVE   NO      NO
HOSTA      ACTIVE   YES     NO
HOSTC      ACTIVE   NO      NO

  4 record(s) selected.
```

この例の場合、このインスタンスにはホストが 4 つあります。3 つはアクティブ、つまりシステムは電源オンの状態になっており、オペレーティング・システム・コマンドに応答できます。 1 つのホストは停止しています。つまり、管理者によってホスト上のインスタンスは明示的に停止されました。調査が必要なアラートはありません。

例

DB2_GET_CLUSTER_HOST_STATE 表関数を使用してホスト状況を取得する

また、DB2_GET_CLUSTER_HOST_STATE 表関数を使用しても、DB2 pureScale インスタンスにおけるホストについての状況情報を取得できます。この表関数を使用する利点の 1 つとしては、パラメーターをこの関数に渡すことにより、戻される結果範囲を絞り込むことができます。例えば、DB2 pureScale インスタンスにおけるホスト HOSTD についての情報を取得するには、以下のような照会を作成します。

```
SELECT varchar(HOSTNAME,10) as HOST,
       varchar(STATE,10) AS STATE,
       ALERT
FROM TABLE(DB2_GET_CLUSTER_HOST_STATE('HOSTD'))
```

結果:

```
HOST       STATE      ALERT
---------- ---------- --------
HOSTD      ACTIVE     NO

  1 record(s) selected.
```

DB2 pureScale インスタンスにおけるメンバーおよびクラスター・キャッシング・ファシリティーの状況情報の表示

DB2 pureScale インスタンスにおけるメンバーおよびクラスター・キャッシング・ファシリティー (CF とも呼ばれます) の操作状況についての詳細情報、例えば、CF が果たしている役割 (1 次またはピアなど)、および メンバーが別のホストにフェイルオーバーしたかどうかなどを表示できます。

このタスクについて

このタスクで示されている例では、**db2instance** システム・コマンドを使用して、DB2 pureScale インスタンスにおけるメンバーおよびクラスター・キャッシング・ファシリティーの状況についての情報を取得する方法が示されています。システム・コマンドを使用する利点は、データベース接続が必要ないことです。ただし、このコマンドは、インスタンス内のメンバー (CF ではない) であるホストから実行する必要があります。

手順

db2instance コマンドを使用してDB2 pureScale インスタンスにおいてメンバーおよび CF についての状況情報を取得するには、インスタンス内のいずれかのメンバーのシステム・プロンプトでこのコマンドに `-list` オプションを指定して入力します。

```
db2instance -list
```

db2instance コマンドは、以下の照会のような情報を戻します (表示上、出力は若干圧縮されています)。

```
ID        TYPE          STATE       HOME_HOST      CURRENT_HOST     ALERT   PARTITION_NUMBER      LOGICAL_PORT    NETNAME
--        ----          -----       ---------      ------------     -----   ----------------      ------------  ---------
0       MEMBER        STARTED           HOSTA            HOSTA        NO                  0                 0  OSTA-ib0
1       MEMBER        STARTED           HOSTB            HOSTB        NO                  0                 0  HOSTB-ib0
2       MEMBER        STARTED           HOSTC            HOSTC        NO                  0                 0  HOSTC-ib0
128     CF            PRIMARY           HOSTD            HOSTD        NO                  -                 0  HOSTD-ib0
129     CF            PEER              HOSTE            HOSTE        NO                  -                 0  HOSTE-ib0

HOSTNAME                 STATE               INSTANCE_STOPPED        ALERT
--------                 -----               ----------------        -----
   HOSTA                ACTIVE                             NO           NO
   HOSTC                ACTIVE                             NO           NO
   HOSTD                ACTIVE                             NO           NO
   HOSTE                ACTIVE                             NO           NO
   HOSTB                ACTIVE                             NO           NO
```

タスクの結果

戻される結果は、ご使用のDB2 pureScale インスタンスの構造によって異なります。この例では、結果は以下を示しています。

- メンバーとして、ホスト `HOSTA` から `HOSTC` が構成されています。
- 各メンバーが開始され、それ自体のホーム・ホストで実行されています。
- ホスト `HOSTD` と `HOSTE` 上で実行されているクラスター・キャッシング・ファシリティー CF が 2 つあります。

- 1 次 CF が HOSTD で実行されていて、もう 1 つの CF が HOSTE で PEER モードで実行されています。このモードは、1 次 CF で障害が発生した場合、1 次 CF としての役割を引き継ぐ準備が整っていることを示します。

例

また、以下のインターフェースを使用して、メンバーおよびクラスター・キャッシング・ファシリティーの状況情報を取得することもできます。

- DB2_MEMBER 管理ビューまたは DB2_CF 管理ビュー
- DB2_GET_INSTANCE_INFO 表関数
- **LIST INSTANCE** コマンド行プロセッサー (CLP) コマンド
- **db2cluster** システム・コマンド

以下の例は、これらのインターフェースの一部の使用法について示しています。

例 1: DB2_MEMBER 管理ビューを使用して状況情報を取得する

DB2_MEMBER 管理ビューは、DB2 pureScale インスタンスにおけるメンバーの状況情報を表示します。以下は、この管理ビューを使用してメンバー状況を取得する方法の例です。

```
SELECT ID,
       varchar(STATE,21) AS STATE,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       ALERT
FROM SYSIBMADM.DB2_MEMBER
```

結果:

```
ID     STATE                 HOME_HOST  CUR_HOST   ALERT
------ --------------------- ---------- ---------- --------
     0 WAITING_FOR_FAILBACK       HOSTA      HOSTB NO
     1 STARTED                    HOSTB      HOSTB NO
     2 STARTED                    HOSTC      HOSTC NO

  3 record(s) selected.
```

この例では、メンバー 0 がそのホーム・ホストで障害を発生し、HOSTB にフェイルオーバーしました。メンバー 0 は、ホーム・ホスト HOSTA にフェイルバックするのを待機しています。

例 2: DB2_GET_INSTANCE_INFO 表関数を使用して CF の状況情報を取得する

DB2_GET_INSTANCE_INFO 表関数を使用して、DB2 pureScale インスタンスにおけるメンバーの状況情報を取得します。この表関数を使用する利点の 1 つとしては、パラメーターをこの関数に渡すことにより、戻される結果範囲を絞り込むことができます。例えば、DB2 pureScale インスタンスにおける CF の情報を取得するには、以下のような照会を作成できます。

```
SELECT ID,
       varchar(STATE,17) AS STATE,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       ALERT
FROM TABLE(DB2_GET_INSTANCE_INFO(NULL,'','','CF',NULL))
```

結果:

```
ID     STATE             HOME_HOST  CUR_HOST   ALERT
------ ---------------- ---------- ---------- --------
   128 RESTARTING            HOSTD      HOSTD NO
   129 BECOMING_PRIMARY      HOSTE      HOSTE NO

  1 record(s) selected.
```

この例では、ホスト ID 128 の CF で障害は発生しました。ホスト ID 129 の CF は、1 次 CF としての引き継ぎの処理中です。

*例 3: **db2instance -list** コマンドによってレポートされたアラートを調査する*

この例では、**db2instance -list** コマンドの実行結果は以下のようになります。

```
$ db2instance -list
ID       TYPE            STATE        HOME_HOST           CURRENT_HOST           ALERT   PARTITION_NUMBER       LOGICAL_PORT    NETNAME
--       ----            -----        ---------           ------------           -----   ----------------       ------------    -------
0        MEMBER          STARTED           HostA                HostA             NO                 0                  0        -
1        MEMBER          STARTED           HostB                HostB             NO                 0                  1        -
2        MEMBER          STARTED           HostC                HostC             NO                 0                  2        -
128      CF              ERROR             HostD                HostD            YES                 -                  0        -
129      CF              ERROR             HostE                HostE            YES                 -                  0        -

HOSTNAME                  STATE              INSTANCE_STOPPED       ALERT
--------                  -----              ----------------       -----
   HostA                 ACTIVE                            NO          NO
   HostB                 ACTIVE                            NO          NO
   HostC                 ACTIVE                            NO          NO
   HostD                 ACTIVE                            NO         YES
   HostE                 ACTIVE                            NO         YES

現在、データ共有インスタンス内のメンバー、CF、またはホストに関するアラートがあります。
アラート、アラートの影響、およびアラートをクリアする方法の詳細については、次のコマンドを実行してください。'db2cluster -cm -list -alert'
```

この例の場合、インスタンス内の 2 つのクラスター・キャッシング・ファシリティーにアラートが存在します。また、これらの CF の状態は ERROR と表示されています。レポートの末尾のメッセージで提案されているように、**db2cluster** コマンドに **-cm -list -alert** オプションを指定して使用すると、アラートに関する詳細情報を表示できます。

```
$db2cluster -cm -list -alert

Alert: CF '128' failed to start the PRIMARY role on host 'HostD'. Check the cadiag*.log for failures related to CF '128' for
more information.

Action: This alert must be cleared manually with the command: 'db2cluster -cm -clear -alert'.

Impact: CF '128' on host 'HostD' will be unavailable to service requests from DB2 members until the alert is cleared.
```

例 4: DB2 pureScale インスタンスでメンバーの開始に失敗したことを示すメンバー・アラート

この例では、**db2instance -list** コマンドの実行結果は以下のようになります。

```
$ db2instance -list
ID       TYPE            STATE        HOME_HOST           CURRENT_HOST           ALERT   PARTITION_NUMBER       LOGICAL_PORT    NETNAME
--       ----            -----        ---------           ------------           -----   ----------------       ------------    -------
0        MEMBER          ERROR             HostA                HostA            YES                 0                  0        -
1        MEMBER          STARTED           HostB                HostB             NO                 0                  1        -
2        MEMBER          STARTED           HostC                HostC             NO                 0                  2        -
128      CF              PRIMARY           HostD                HostD             NO                 -                  0        -
129      CF                 PEER           HostE                HostE             NO                 -                  0        -

HOSTNAME                  STATE              INSTANCE_STOPPED       ALERT
--------                  -----              ----------------       -----
   HostA                 ACTIVE                            NO          NO
   HostB                 ACTIVE                            NO          NO
   HostC                 ACTIVE                            NO          NO
   HostD                 ACTIVE                            NO          NO
   HostE                 ACTIVE                            NO          NO

現在、データ共有インスタンス内のメンバー、CF、またはホストに関するアラートがあります。
アラート、アラートの影響、およびアラートをクリアする方法の詳細については、次のコマンドを実行してください。'db2cluster -cm -list -alert'
```

この例では、DB2 pureScale インスタンスにおいてメンバーを開始できませんでした。**db2cluster** コマンドに **-cm -list -alert** オプションを指定して実行すると、実行すべきアクションが提案され、DB2 pureScale インスタ

ンスにおいてこの障害が及ぼす影響の概略が示されます。

```
$db2cluster -cm -list -alert

Alert: DB2 member '0' failed to start on its home host 'HostA'. The cluster manager will attempt to restart the DB2 member in restart
light mode on another host. Check the db2diag.log for messages concerning failures on host 'HostA' for member '0'."

Action: This alert must be cleared manually with the command: 'db2cluster -cm -clear -alert'.

Impact: DB2 member '%0' will not be able to service requests until this alert has been cleared and the DB2 member returns to its home host.
```

例 5: 2 次 CF が CATCHUP フェーズに失敗したことを示す CF エラー

この例では、**db2instance -list** コマンドの実行結果は以下のようになります。

```
$ db2instance -list
ID        TYPE            STATE         HOME_HOST              CURRENT_HOST           ALERT   PARTITION_NUMBER       LOGICAL_PORT    NETNAME
--        ----            -----         ---------              ------------           -----   ----------------       ------------    -------
0       MEMBER          STARTED            HostA                   HostA              NO                 0                  0         -
1       MEMBER          STARTED            HostB                   HostB              NO                 0                  1         -
2       MEMBER          STARTED            HostC                   HostC              NO                 0                  2         -
128     CF              PRIMARY            HostD                   HostD              NO                 -                  0         -
129     CF                ERROR            HostE                   HostE             YES                 -                  0         -

HOSTNAME                 STATE              INSTANCE_STOPPED       ALERT
--------                 -----              ----------------       -----
   HostA                ACTIVE                            NO          NO
   HostB                ACTIVE                            NO          NO
   HostC                ACTIVE                            NO          NO
   HostD                ACTIVE                            NO          NO
   HostE                ACTIVE                            NO          NO

現在、データ共有インスタンス内のメンバー、CF、またはホストに関するアラートがあります。
アラート、アラートの影響、およびアラートをクリアする方法の詳細については、次のコマンドを実行してください。'db2cluster -cm -list -alert'
```

この例では、2 次 CF で CATCHUP フェーズが失敗しました。**db2cluster** コマンドに **-cm -list -alert** オプションを指定して実行すると、実行すべきアクションが提案され、DB2 pureScale インスタンスにおいてこの障害が及ぼす影響の概略が示されます。

```
$db2cluster -cm -list -alert

Alert: CF '129' failed to complete CATCHUP on host 'HostE'. Check the db2diag.log for failure messages pertaining to CATCHUP on CF '129'.

Action: Contact IBM support to determine the reason for the failure. To re-attempt CATCHUP, restart the failed CF with the commands:
'db2stop 129; db2start 129'. This alert will clear itself when the CF is restarted.

Impact: CF '129' on host 'HostE' will not be available until it can undergo CATCHUP successfully.
```

例 6: ホスト「HostA」でネットワーク接続が失われたことを示すホスト・アラート

この例では、**db2instance -list** コマンドの実行結果は以下のようになります。

```
$ db2instance -list
ID        TYPE                STATE         HOME_HOST              CURRENT_HOST           ALERT   PARTITION_NUMBER       LOGICAL_PORT    NETNAME
--        ----                -----         ---------              ------------           -----   ----------------       ------------    -------
0       MEMBER  WAITING_FOR_FAILBACK            HostA                   HostA              NO                 0                  0         -
1       MEMBER              STARTED            HostB                   HostB              NO                 0                  1         -
2       MEMBER              STARTED            HostC                   HostC              NO                 0                  2         -
128     CF                  PRIMARY            HostD                   HostD              NO                 -                  0         -
129     CF                     PEER            HostE                   HostE              NO                 -                  0         -

HOSTNAME                 STATE              INSTANCE_STOPPED       ALERT
--------                 -----              ----------------       -----
   HostA               INACTIVE                            NO         YES
   HostB                ACTIVE                            NO          NO
   HostC                ACTIVE                            NO          NO
   HostD                ACTIVE                            NO          NO
   HostE                ACTIVE                            NO          NO

 現在、データ共有インスタンス内のメンバー、CF、またはホストに関するアラートがあります。
アラート、アラートの影響、およびアラートをクリアする方法の詳細については、次のコマンドを実行してください。'db2cluster -cm -list -alert'
```

この例では、ホスト「HostA」でネットワーク接続が失われました。**db2cluster** コマンドに **-cm -list -alert** オプションを指定して実行すると、実行すべきアクションが提案され、DB2 pureScale インスタンスにおいてこの障害が及ぼす影響の概略が示されます。

```
$db2cluster -cm -list -alert

Alert: Host 'HostA' is INACTIVE. Ensure the host is powered on and connected to the network.

Action: This alert will clear itself when the host is ACTIVE.

Impact: While the host is INACTIVE, the DB2 members on this host will be in restart light mode on other hosts and will be in the
WAITING_FOR_FAILBACK state. Any CF defined on the host will not be able to start, and the host will not be available as a target
for restart light.
```

メンバーの再開状況の確認

メンバーで障害が発生したことが分かる場合、原因となった電源障害やその他のハードウェア障害などを修正した後、正常に再始動されたかどうかを確認したい場合があります。

このタスクについて

DB2_MEMBER 管理ビューを使用すると、DB2 pureScale インスタンスにおけるすべてのメンバーの操作状況を調べることができます。また、DB2_GET_INSTANCE_INFO 表関数を使用して、特定のホストに対する照会オプションを指定することもできます。

メンバーの再始動状況を確認するための詳細なプロセスは、 1762 ページの『DB2 pureScale インスタンスにおけるメンバーおよびクラスター・キャッシング・ファシリティーの状況情報の表示』の例 1 に示されています。特に、以下の列の値を取得するために、DB2_MEMBER 管理ビュー (または DB2_GET_INSTANCE_INFO 表関数) を使用する SQL 照会を作成してください。

- ID
- HOME_HOST
- CURRENT_HOST
- STATE
- ALERT

手順

1. 用いやすいどちらかのインターフェースを使用して SQL 照会を作成します。以下の例では、DB2_MEMBER 管理ビューを使用します。

   ```
   SELECT ID,
          varchar(STATE,21) AS STATE,
          varchar(HOME_HOST,10) AS HOME_HOST,
          varchar(CURRENT_HOST,10) AS CUR_HOST,
          ALERT
   FROM SYSIBMADM.DB2_MEMBER
   ```

2. この照会を実行します。 戻される結果は、以下のようになります。

   ```
   ID     STATE                HOME_HOST  CUR_HOST   ALERT
   ------ -------------------- ---------- ---------- --------
        0 STARTED              HOSTA      HOSTA      NO
        1 STARTED              HOSTB      HOSTB      NO
        2 STARTED              HOSTC      HOSTC      NO

     3 record(s) selected.
   ```

   前述の例の場合、すべてのメンバーがそれぞれのホストで実行されていて、アラートはありません。

タスクの結果

メンバーの再始動状況を調べて、以下を確認します。

- STATE 列の値が RESTARTING または STARTED のどちらであるかを確認します。前者の場合、メンバーは再始動の処理中であり、後者の場合には正常に再始動されたことを示します。状態が RESTARTING の場合、数分後にもう一度状況を確認し、状態が STARTED に変更されたかどうかを確かめてください。
- CUR_HOST の値が HOME_HOST の値と同じであることを確認します。これは、メンバーがホーム・ホストで実行されていることを示します。
- 対象のメンバーのアラート列に YES の値がないことを確認します。

CUR_HOST が HOME_HOST と異なる場合、状態が RESTARTING から移行しない場合、WAITING_FOR_FAILBACK のままの場合、またはアラート列の値が YES の場合、問題がある可能性があるためさらに調査が必要となります。

例

例 1: 再始動の処理中にメンバーで障害が生じた

この例では、メンバー 0 はホーム・ホスト HOSTA で再始動の処理中です。

```
ID     STATE                HOME_HOST  CUR_HOST   ALERT
------ -------------------- ---------- ---------- --------
     0 RESTARTING           HOSTA      HOSTA      NO
     1 STARTED              HOSTB      HOSTB      NO
     2 STARTED              HOSTC      HOSTC      NO

  3 record(s) selected.
```

再始動が最終的に正常に行われたかどうかを確認するため、数秒後にもう一度この照会を実行します。

例 2: メンバーが再始動できずに障害が生じた

この例の場合、メンバー 0 はホーム・ホストにフェイルバックするのを待機しています。現在、HOSTB 上において light モードで実行されています。

```
ID     STATE                HOME_HOST  CUR_HOST   ALERT
------ -------------------- ---------- ---------- --------
     0 WAITING_FOR_FAILBACK HOSTA      HOSTB      NO
     1 STARTED              HOSTB      HOSTB      NO
     2 STARTED              HOSTC      HOSTC      NO

  3 record(s) selected.
```

この場合、HOSTA のホスト状況を確認し、問題がないかどうかを確認できます。 DB2_CLUSTER_HOST_STATE 管理ビューを使用すると、以下のような結果が戻されます。

```
HOST       STATE    STOPPED ALERT
---------- -------- ------- --------
HOSTD      ACTIVE   NO      NO
HOSTB      ACTIVE   NO      NO
HOSTA      INACTIVE NO      YES
HOSTC      ACTIVE   NO      NO

  4 record(s) selected.
```

このレポートでは、HOSTA にアラートがあり、このホストが非アクティブであることを示しています。ただし、インスタンスは **db2stop** コマンドによって停止されていません。さらに調査すると、このホストに対する電力不足などが原因の場合があります。ホストに関する問題を解決してから、再始動状況をもう一度確認し、メンバーを再始動できるかどうかを確かめてください。

アラートに関する詳細情報の表示

いずれかのメンバーまたはクラスター・キャッシング・ファシリティーでアラートがレポートされる場合、 DB2_INSTANCE_ALERTS 管理ビューを使用してアラートの詳細情報を表示できます。または、**db2cluster** コマンドに **-cm -list -alert** パラメーターを指定して使用することもできます。

このタスクについて

このタスクでは、DB2 pureScale インスタンスのホストの 1 つで生じたアラートを既に判別しているとします。例えば、**LIST INSTANCE** コマンドを使用すると、いずれかのメンバーに関してアラートが表示される場合があります。

```
ID     TYPE    STATE   HOME_HOST CURRENT_HOST ALERT
------ ------- ------- --------- ------------ -----
     0 MEMBER  STARTED hostA     hostA        YES
     1 MEMBER  STARTED hostB     hostB        NO
     2 MEMBER  STARTED hostC     hostC        NO
     3 MEMBER  STARTED hostD     hostD        NO
   128 CF      PRIMARY hostE     -            NO
   129 CF      PEER    hostF     -            NO

HOSTNAME  STATE  INSTANCE_STOPPED ALERT
--------- ------ ---------------- -----
hostA     ACTIVE NO               NO
hostB     ACTIVE NO               NO
hostC     ACTIVE NO               NO
hostD     ACTIVE NO               NO
hostE     ACTIVE NO               NO
hostF     ACTIVE NO               NO
```

この場合、メンバー 0 にアラートが表示されています。

手順

インスタンスのアラートに関する詳細を調べるには、以下のようにします。

1. DB2_INSTANCE_ALERTS 管理ビューを使用する SQL ステートメントを作成します。

   ```
   SELECT * FROM SYSIBMADM.DB2_INSTANCE_ALERTS
   ```

2. この SQL ステートメントを実行します。

タスクの結果

DB2_INSTANCE_ALERTS 管理ビューが戻す情報は、メンバー 0 での問題の性質によって異なります。例えば、以下のようなメッセージを受け取る場合があります。

```
MESSAGE
-------------------------------------------------------------------------------------------------------------
Could not restart light DB2 member '0' on hosts 'hostA'. Check the db2diag.log for messages concerning a
restart light or database crash recovery failure on the indicated hosts for DB2 member '0'.

ALERT_ACTION
-------------------------------------------------------------------------------------------
This alert must be cleared manually with the command: 'db2cluster -clear -alert -member 0'

IMPACT
-------------------------------------------------------------------------------------------------------------
DB2 member '0' will not be able to restart light on host 'hostC' until this alert has been cleared.
```

例

*例 1: **db2cluster** コマンドを使用してアラート情報を表示する*

この例では、**db2instance -list** コマンドは以下の情報を戻します。

```
$ db2instance -list
ID        TYPE            STATE          HOME_HOST              CURRENT_HOST           ALERT   PARTITION_NUMBER       LOGICAL_PORT    NETNAME
--        ----            -----          ---------              ------------           -----   ----------------       ------------    -------
0       MEMBER          ERROR              HostA                  HostA                YES               0                  0         -
1       MEMBER          STARTED            HostB                  HostB                 NO               0                  1         -
2       MEMBER          STARTED            HostC                  HostC                 NO               0                  2         -
128     CF              PRIMARY            HostD                  HostD                 NO               -                  0         -
129     CF              PEER               HostE                  HostE                 NO               -                  0         -

HOSTNAME                  STATE              INSTANCE_STOPPED       ALERT
--------                  -----              ----------------       -----
   HostA                ACTIVE                           NO         YES
   HostB                ACTIVE                           NO          NO
   HostC                ACTIVE                           NO          NO
   HostD                ACTIVE                           NO          NO
   HostE                ACTIVE                           NO          NO

現在、データ共有インスタンス内のメンバー、CF、またはホストに関するアラートがあります。
アラート、アラートの影響、およびアラートをクリアする方法の詳細については、次のコマンドを実行してください。'db2cluster -cm -list -alert'
```

1757 ページの表 2164 の情報から、**db2instance -list** の出力によって メンバー 0 がいずれのホストにおいても開始できなかったことが分かります (ゲスト・ホストで開始したとすると、状況は STARTED になり、現行ホストには実行しているホストの名前が示されることなります)。

db2cluster -cm -list -alert コマンドを使用すると、以下のメッセージが表示されます。

```
$ db2cluster -cm -list -alert

Alert: DB2 member '0' failed to start on its home host 'Host A'. The cluster manager will
attempt to restart the DB2 member
in restart light mode on another host. Check the db2diag.log for messages concerning failures
on hosts 'HostA' for member '0'.

Action: This alert must be cleared manually with the command: 'db2cluster -cm -clear -alert'.

Impact: DB2 member '0' will not be able to service requests until this alert has been cleared
and the DB2 member returns to its home host.
```

次のタスク

調査を行うか、DB2_INSTANCE_ALERTS 管理ビューまたは **db2cluster** コマンドによって戻された情報で指定されたアクションを実行します。

第 13 章 DB2 pureScale 環境におけるイベント、リアルタイム・データベース、およびシステムのモニター

DB2 pureScale インスタンスのコンポーネントの全体的な状況を表示できるだけでなく、DB2 モニター・インフラストラクチャーを使用すると、クラスター・キャッシング・ファシリティーとメンバーの作動状況の特定の面について調べることができます。

こうした情報は、モニター表関数およびモニター管理ビューを使用して表示できます。また、選択したイベント・モニターを使用して、イベントの発生時にイベントをキャプチャーすることも可能です。

DB2 V9.7 では、DB2 製品用のモニター・インフラストラクチャーに多数の機能拡張が導入されました。こうした機能拡張の 1 つは、多数のメモリー内 のモニター・エレメントにアクセスできる一連の表関数で、これを使用すると、特定時点でのデータベース環境の状態を照会できます。別の機能拡張では、生じたロック、作業単位、およびアクティビティーなどに関する情報をキャプチャーするためのイベント・モニターが改良されました。

DB2 pureScale Feature では、 DB2 データベースに組み込まれているモニター機能が拡張されており、DB2 pureScale インスタンスにおけるクラスター・キャッシング・ファシリティー (CF とも呼ばれます) とメンバーの作動状況の特定の面について取り上げたデータを表示できるモニター・エレメントが備わっています。ただし、DB2 pureScale インスタンスのモニターと他の DB2 インスタンスのモニターには若干の相違点があるので、以下の点に注意してください。

- 『DB2 メンバーに加えて CF をモニターする機能』
- 1772 ページの『DB2 pureScale インスタンスのモニター・エレメントのレポート方法』
- モニター・エレメントのレポート作成に関するコンポーネント障害の影響。

DB2 メンバーに加えて CF をモニターする機能

DB2 pureScale 環境において、メンバーとは異なる役割を担う CF によって、これまでとは別のモニターを行う必要性が生じました。例えば、DB2 pureScale インスタンス以外の DB2 インスタンスでは、バッファー・プールのヒット率をモニター対象とする場合があります。これは、ディスクから読み取る必要のあるページ数とメモリー内にあるページ数とを比較した数値です。一般的には、バッファー・プールのヒット率が高いほど、パフォーマンスが良いということになります。必要なページをメモリーに転送するための入出力が少なくなるので、パフォーマンスが良くなるわけです。DB2 pureScale 環境では、ディスクからの物理ページの読み取りすべてはメンバーによって実行されますが、これが実行されるのは、まず CF を使用して、グループ・バッファー・プールに使用可能な有効なページのある他のメンバーのレコードがあるかどうかをチェックしてからになります。そのため、DB2 pureScale 環境以外の DB2 環境ではローカル・バッファー・プールだけを調整するのに対して、DB2 pureScale 環境では CF のグループ・バッファー・プールのバッファー・プール・ヒット率のモニターも重要になります。ローカル・バッファー・

プールまたはグループ・バッファー・プール (GBP) で該当ページが見つかる回数が多ければ、それだけディスクから読み取らなければならない回数が減ります。

GBP に加えて、グローバル・ロック・マネージャー (GLM) も CF のモニター対象にできる別のコンポーネントです。GLM は、DB2 pureScale インスタンスにおけるすべてのメンバーのオブジェクトのロックを管理します。DB2 pureScale Feature は、メンバー間のロックをモニターするためのモニター・エレメントを追加します。

DB2 pureScale インスタンスのモニター・エレメントのレポート方法

通常、DB2 pureScale インスタンスにおけるモニターのメカニズムは、他の DB2 インスタンスにおけるモニター・メカニズムと似ています。例えば、データベース内の表スペースについての情報を戻す MON_GET_TABLESPACE 表関数は、DB2 pureScale インスタンスと他の DB2 インスタンスで同じように作動します。DB2 pureScale インスタンスの場合、一部のモニター・エレメントの有効範囲は特定のメンバーに限定され、他の有効範囲はすべてのメンバーにおいてグローバルとなります。例えば、**direct_reads**、または **pool_data_l_reads** などのモニター・エレメントからのデータは、メンバーによって実行される読み取りアクティビティーに限定的となります。一方、表スペースの物理属性を表す **tbsp_total_pages** などのモニター・エレメントはすべてのメンバーで同じです。すべてのメンバーが同じ表スペースを共有しているからです。例えば次の照会を考えてみましょう。

```
SELECT VARCHAR(TBSP_NAME, 30) AS TBSP_NAME,
       MEMBER, POOL_DATA_L_READS,
       TBSP_TOTAL_PAGES
FROM TABLE(MON_GET_TABLESPACE('USERSPACE1',-2))
```

この照会の結果は、以下の例のようになります。

```
TBSP_NAME                      MEMBER POOL_DATA_L_READS    TBSP_TOTAL_PAGES
------------------------------ ------ -------------------- --------------------
USERSPACE1                          1                    0                 4096
USERSPACE1                          2                    0                 4096
USERSPACE1                          3                    0                 4096
USERSPACE1                          0                   36                 4096

  4 record(s) selected.
```

この例では、ローカル・バッファー・プールからの論理読み取り数はメンバーごとに異なります。それぞれのメンバーは他のメンバーとは別個に読み取りを実行しているからです。ただし、表スペースの合計ページはすべてのメンバーで同じです。すべてのメンバーが同じインスタンス USERSPACE1 から作業しているためです。

モニター・エレメントのレポート作成に関するコンポーネント障害の影響

DB2 pureScale 環境内のホスト、メンバー、または CF で障害が発生した場合、DB2 pureScale インスタンス全体がダウンしたのではない限りは、インスタンスからモニター・エレメントを取得できます。ただし、障害が発生したコンポーネントでは統計は生成されません。これは、『DB2 pureScale インスタンスのモニター・エレメントのレポート方法』に示されていた最初の例などの照会を実行すると明ら

かです。そうした照会では、それぞれのメンバーのデータは個別に表示されます。メンバー間の情報を集約する照会を使用すると、あるメンバーのデータが欠落していることに気付かない場合があります。

銘記しておくべき別の点は、モニター・エレメントのデータ収集時にメンバーで障害が生じると、障害が発生したメンバーとの通信問題が検出されるまで、または TCP/IP のタイムアウト期間が経過するまでは、データ収集プロセスが一時停止します。この場合、データはレポートされますが、障害の発生したメンバーからは情報が収集されません。

最後に、メンバーで障害が発生した場合には、そのモニター・エレメントで集計された統計はすべて 0 にリセットされることにも注意してください。

クラスター・キャッシング・ファシリティーメモリーと CPU の使用量モニターの概要

クラスター・キャッシング・ファシリティーの運用効率の基本的な指標は、メモリーと CPU が構成された最大容量に対してどの程度まで常時使用されているかという点です。

メモリー使用量

クラスター・キャッシング・ファシリティー (CF とも呼ばれます) は、以下の目的で種々のメモリー・ヒープを使用します。

グループ・バッファー・プール・メモリー
グループ・バッファー・プール・メモリーは、DB2 pureScale インスタンスのグループ・バッファー・プールに使用されます。このタイプのメモリーが構成された最大容量まで常時使用されると、パフォーマンスに悪影響を及ぼす可能性があります。ただし、メモリーが容量まで使用される場合があるということ自体が、パフォーマンスに影響を及ぼす指標となるわけではありません。パフォーマンスが低下しているかどうかを確かめるには、グループ・バッファー・プールのヒット・レートを調べてください。グループ・バッファー・プール・メモリーの使用量が多い一方でヒット・レートが低い場合には、このタイプのメモリーを増やす必要があることを示している可能性があります。このタイプのメモリーは、**cf_gbp_sz** 構成パラメーターで構成します。

ロック・メモリー
ロック・メモリーは、DB2 pureScale インスタンスにおけるページ・ロックを管理します。 CF でロック用に使用可能な十分なメモリーがない場合には、以下の 1 つ以上の状態が生じる場合があります。

- ロック・エスカレーションが生じる可能性があります。これにより、関係するオブジェクトの並行性が低下します。
- ロック要求が拒否される場合があります。その結果、SQL0912 メッセージが戻されます。

このタイプのメモリーは、**cf_lock_sz** 構成パラメーターで構成します。

共用通信域 (SCA) メモリー
SCA メモリーには、表、索引、表スペース、およびカタログのデータベース規模の情報が含まれます。それぞれのデータベースの CF には、独自の

SCA メモリーがあります。このメモリーは、いずれかの DB2 メンバーで最初にデータベースがアクティブにされると割り振られ、データベースがドロップされるか、CF が停止されるまでは解放されません。表パーティショニングを使用している場合は、 CF とメンバーの間で表パーティショニング・データを同期するために必要な情報も SCA メモリーに保管されます。

このタイプのメモリーが容量まで使用されると、表はロードに失敗し、エラーが戻される場合があります。このタイプのメモリーは、**cf_sca_sz** 構成パラメーターで構成します。

CF 全体メモリー

CF 全体メモリーは、CF で使用可能な物理メモリーの合計です。**cf_mem_size** 構成パラメーターで設定します。グループ・バッファー・プールのメモリー、ロック用のメモリー、および共用通信域用のメモリーは、すべてこのメモリー・プールから割り振られます。そのため、特定のタイプのメモリーに割り振られるメモリーの合計量が、 **cf_mem_size** 構成パラメーターで構成されたメモリー量を超えることはできません。

デフォルトでは、これらのタイプのそれぞれのメモリーの構成が自動的に行われます。DB2 pureScale Feature には、システムによって現在使用されているこれらのタイプのそれぞれのメモリー量を調べることが可能なモニター・エレメントが備わっています。また、各タイプのメモリーの最大サイズ、およびメモリーのサイズ変更操作が進行中かどうかを判別できる関連エレメントもあります。

特定のタイプの CF メモリーの使用量に関するレポートを作成するモニター・エレメントの他に、ENV_CF_SYS_RESOURCES 管理ビューを使用して、CF で使用可能な物理ストレージと仮想ストレージの合計量を調べることもできます。

CPU ロード

CF における CPU ロードは、プロセッサーにどれほどの負荷がかかっているかを示す指標です。CF が実行されているホスト上のプロセッサーがほとんど常に最大能力で稼働している場合には、CF が実行されているホストの性能が不足していることを示す可能性があります。プロセッサーを追加するか、より高性能のシステムにアップグレードできます。

ENV_CF_SYS_RESOURCES 管理ビューを使用すると、DB2 pureScale インスタンスで CF として作動しているホストの全体的な CPU ロードを確認できます。

注: CPU ロードについてレポートされる値には、CF によって実行される実際の処理用の CPU と、CF からの処理以外のホスト処理用の CPU の合計使用量が反映されます。

クラスター・キャッシング・ファシリティーのメモリー使用量を表示するためのモニター・エレメント

IBM DB2 pureScale Feature では、クラスター・キャッシング・ファシリティーのメモリー使用量に関してレポートする数々のモニター・エレメントが提供されています。

モニター・エレメント

次のモニター・エレメントにより、割り振られているCF メモリー・ヒープとその使用に関する情報が提供されます。

- 909 ページの『configured_cf_gbp_size - 構成済みクラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント』
- 956 ページの『current_cf_gbp_size - 現行クラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント』
- 1583 ページの『target_cf_gbp_size - ターゲット・クラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント』
- 909 ページの『configured_cf_lock_size - 構成済みクラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント』
- 956 ページの『current_cf_lock_size - 現行クラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント』
- 1583 ページの『target_cf_lock_size - ターゲット・クラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント』
- 910 ページの『configured_cf_mem_size - 構成済みクラスター・キャッシング・ファシリティーのメモリー・サイズ : モニター・エレメント』
- 957 ページの『current_cf_mem_size - 現行クラスター・キャッシング・ファシリティーのメモリー・サイズ :モニター・エレメント』
- 909 ページの『configured_cf_sca_size - 構成済みクラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント』
- 957 ページの『current_cf_sca_size - 現行クラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント』
- 1583 ページの『target_cf_sca_size - ターゲット・クラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント』

ヒント: いずれの場合も、上記のそれぞれのモニター・エレメントに関してレポートされる値は 4 K ページ単位で表現されます。そのため、例えば **current_gbp_size** モニター・エレメントを照会し、値 350 が戻る場合、実際に GBP で現在使用されているメモリー量は 350 ×4096 バイト = 1,433,600 バイトとなります。

こうしたタイプのメモリーのほとんどに関して、メモリーの構成方法を異なった角度から表した、照会可能な 3 つのモニター・エレメントがあります。

現行 現行メモリー・サイズ (例えば、current_cf_gbp_size、current_cf_mem_size など) は、システム内で現在使用されているそのタイプのメモリー量を表します。

構成済み
構成済みメモリー・サイズ (例えば、configured_cf_sca_size、configured_cf_mem_size など) は、現在データベースでこのタイプのメモリーが構成されている最大合計量を表します。現行メモリー値が、構成済みメモリー値を超えることはあり得ません。

ターゲット

ターゲット・メモリー・サイズ (例えば、target_cf_sca_size など) は、そのタイプのメモリーに関して新たに構成された最大値を表します。通常は、ターゲット・サイズは構成済みサイズと同じです。ただし、ターゲット・サイズと構成済みサイズが異なる場合、それは、その特定タイプのメモリーの構成済みサイズに関するオンライン変更がその時点で行われていることを示します。その後、メモリーの割り振り処理が行われます。このサイズ変更処理のいずれの時点においても、構成済みメモリーはその特定の時点で使用可能なそのタイプのメモリーの最大量を表します。最終的には、構成済みメモリーは、ターゲット・メモリーと同じになります。

各モニター・エレメントに関する参照トピックをご覧になり、そのモニター・エレメントに関連したデータを調べることが可能なモニター・インターフェースを確認してください。

クラスター・キャッシング・ファシリティーのメモリー使用量モニター・エレメントからの情報の取得

MON_GET_CF 表関数を使用すると、DB2 pureScale インスタンスにおけるクラスター・キャッシング・ファシリティーでのメモリー使用量についてレポートする種々のモニター・エレメントを取得できます。

このタスクについて

クラスター・キャッシング・ファシリティー (CF とも呼ばれます) においてメモリーがどの程度使用されているかを理解していると、メモリー割り振りを調整するかどうかを判断する上で役立ちます。例えば、クラスター・キャッシング・ファシリティーが割り振られているバッファー・プール・メモリーのほぼ最大量を使用している場合、グループ・バッファー・プールのヒット・レートは可能な最大レートよりも低くなる可能性があります。または、ロック・メモリーが最大限に使用されている場合、ロック・エスカレーションの数が予想されていたよりも多いことがあります。

手順

クラスター・キャッシング・ファシリティーにおけるメモリー使用量についての情報を取得するには、以下のようにします。

1. 取得するモニター・エレメントを判別します。 例えば、ロック・メモリー使用量を表示する場合には、以下の 1 つ以上のモニター・エレメントを選択できます。
   - current_cf_lock_size
   - configured_cf_lock_size
   - target_cf_lock_size
2. MON_GET_CF 表関数を使用して照会を作成します。 ステップ 1 の例を使用すると、ステートメントは以下の例のようになります。

```
SELECT SUBSTR(HOST_NAME,1,10) AS HOST,
       ID as HOSTID,
       CURRENT_CF_LOCK_SIZE,
       CONFIGURED_CF_LOCK_SIZE,
       TARGET_CF_LOCK_SIZE
  FROM TABLE( MON_GET_CF( NULL ) )
```

3. この照会を実行します。 この例を用いると、前述の照会の出力は以下の例のようになります。

```
HOST       HOSTID CURRENT_CF_LOCK_SIZE CONFIGURED_CF_LOCK_SIZE
---------- ------ -------------------- ----------------------
HOSTA         128               133852                 564224
HOSTB         129               133852                 564224

TARGET_CF_LOCK_SIZE
-------------------
             564224
             564224


  2 record(s) selected.
```

注: 両方のクラスター・キャッシング・ファシリティーのロック・メモリーは通常は同じです。片方のクラスター・キャッシング・ファシリティーがもう一方に情報と構成を二重化するからです。 1762 ページの『DB2 pureScale インスタンスにおけるメンバーおよびクラスター・キャッシング・ファシリティーの状況情報の表示』で説明されているいずれかのインターフェースを使用すると、2 つのクラスター・キャッシング・ファシリティーのどちらが 1 次であるかを確認できます。そうしたインターフェースの例としては、**LIST INSTANCE** コマンドと **db2cluster** コマンドの 2 つがあります。

タスクの結果

前述の例では、使用されているロック・メモリーの現行量は 170,258 4k ブロック、つまり 697,376,768 バイトです。使用可能なロック・メモリーの最大量は 564,224 4k ブロック、つまり 2,311,061,504 バイトです。

例

例 1: グループ・バッファー・プールのメモリー使用量データを取得する

以下の照会によって、システム上のすべてのクラスター・キャッシング・ファシリティーのグループ・バッファー・プールのメモリー・サイズについての情報が表示されます。

```
SELECT SUBSTR(HOST_NAME,1,20) AS HOST,
       ID as HOSTID,
       CURRENT_CF_GBP_SIZE,
       CONFIGURED_CF_GBP_SIZE,
       TARGET_CF_GBP_SIZE
  FROM TABLE( MON_GET_CF(NULL) )
```

前述の照会によって戻される出力は、以下の例のようになります。

```
HOST                HOSTID CURRENT_CF_GBP_SIZE CONFIGURED_CF_GBP_SIZE TARGET_CF_GBP_SIZE
------------------- ------ ------------------- ---------------------- -------------------
HOSTA                  128              367611                 500224              500224
HOSTB                  129              367611                 500224              500224

  2 record(s) selected.
```

この例では、現行の GBP サイズは 367,611 4k ページ、つまり 1,505,734,656 バイトです。GBP に割り振られているメモリーは 500,224 4k ページ、つまり 2,048,917,504 バイトです。

例 2: 特定のホストに関する共用通信域 (SCA) メモリー使用量データを取得する

以下の照会によって、ID 128 のクラスター・キャッシング・ファシリティーの SCA メモリー・サイズについての情報が表示されます。

```
SELECT SUBSTR(HOST_NAME,1,8)AS HOST,
       ID as HOSTID,
       CURRENT_CF_SCA_SIZE,
       CONFIGURED_CF_SCA_SIZE,
       TARGET_CF_SCA_SIZE
FROM TABLE(MON_GET_CF(128))
```

前述の照会によって戻される出力は、以下の例のようになります。

```
HOST     HOSTID CURRENT_CF_SCA_SIZE CONFIGURED_CF_SCA_SIZE
-------- ------ ------------------- ----------------------
HOSTA       128                  43                  16128

TARGET_CF_SCA_SIZE
------------------
             23280

   1 record(s) selected.
```

この例では、現在使用されている SCA メモリーは 43 4k ページです。これらのモニター・エレメントが取得された時点で、最大の SCA メモリー・サイズは 16,128 4k ページでした。ただし、構成済みサイズとターゲット・サイズは異なります。つまり、SCA メモリーはサイズが増加し、以前の構成済み最大サイズから 23,280 ページになったということを意味します。

次のタスク

前述のこれらすべての例において、メモリー使用量に関して戻される値はメモリー使用量に関する概要しか提供していないということに注意してください。これらの値自体は、メモリー構成を変更するかどうかの決定を下すうえで、必ずしも十分な情報をもたらすわけではありません。例えば、グループ・バッファー・プール (GBP) のサイズ自体は、DB2 pureScale インスタンスに対して十分な大きさかどうかを示すものではありません。それを判別するには、バッファー・プールのアクティビティーに関してレポートするモニター・エレメントを使用して、バッファー・プール・ヒット・レートを計算することを考慮してください。ヒット・レートによって、バッファー・プールのサイズを調整する必要があるかどうかが分かります。ロック・メモリーの場合には、生じているロック・エスカレーション数を調べると、十分なロック・メモリーが割り振られているかどうかを把握できます。ロック・エスカレーションのレートが高いと、ロック・メモリーを増やす必要があることを示す可能性があります。

クラスター・キャッシング・ファシリティーのプロセッサー・ロードの表示

ENV_CF_SYS_RESOURCES 管理ビューを使用すると、DB2 pureScale インスタンスにおける 1 次クラスター・キャッシング・ファシリティーの全体的な CPU ロードを表示できます。

始める前に

DB2 pureScale インスタンスで実行中のデータベースに接続する必要があります。

このタスクについて

ENV_CF_SYS_RESOURCES 管理ビューは、DB2 pureScale インスタンスにおけるすべてのクラスター・キャッシング・ファシリティー (CF とも呼ばれます) についての情報を戻します。複数の CF が構成されているインスタンスでは、1 つが 1 次として動作し、その他は PEER モードで実行されるバックアップ CF として作動します。ENV_CF_SYS_RESOURCES 管理ビューは、すべての CF に関する情報を戻します。

注: CPU ロードについてレポートされる値には、CF によって実行される実際の処理用の CPU と、CF からの処理以外のホスト処理用の CPU の合計使用量が反映されます。

手順

DB2 pureScale インスタンスにおける CF での CPU ロードを判別するには、以下のようにします。

1. ENV_CF_SYS_RESOURCES 管理ビューを使用する SQL ステートメントを作成します。 例えば、

```
SELECT  VARCHAR(NAME,20) AS HOST_ATTRIBUTE,
        VARCHAR(VALUE,25) AS VALUE,
        VARCHAR(UNIT,8) AS UNIT
FROM SYSIBMADM.ENV_CF_SYS_RESOURCES
```

2. ステートメントを実行します。 この照会は、以下の出力を返します。

```
HOST_ATTRIBUTE       VALUE                     UNIT
-------------------- ------------------------- --------
HOST_NAME            HOSTA                       -
MEMORY_TOTAL         24108                     MB
MEMORY_FREE          3504                      MB
MEMORY_SWAP_TOTAL    4102                      MB
MEMORY_SWAP_FREE     4063                      MB
VIRTUAL_MEM_TOTAL    28211                     MB
VIRTUAL_MEM_FREE     7568                      MB
CPU_USAGE_TOTAL      96                        PERCENT
HOST_NAME            HOSTB                     -
MEMORY_TOTAL         24108                     MB
MEMORY_FREE          3342                      MB
MEMORY_SWAP_TOTAL    4102                      MB
MEMORY_SWAP_FREE     4063                      MB
VIRTUAL_MEM_TOTAL    28211                     MB
VIRTUAL_MEM_FREE     7406                      MB
CPU_USAGE_TOTAL      97                        PERCENT

  16 record(s) selected.
```

   この出力では、HOSTA と HOSTB の両方の結果が記されています。これは、CF として作動するように構成されたホストが 2 つあることを示します。

3. どちらのホストが 1 次 CF として動作するかを判別するには、DB2_CF 管理ビューを使用できます。

```
SELECT VARCHAR(CURRENT_HOST,12) AS HOST,
       ID,
       STATE
FROM SYSIBMADM.DB2_CF
```

   この照会は、以下の出力を返します。

```
HOST         ID     STATE
------------ ------ --------------------------------
HOSTA           128 PRIMARY
HOSTB           129 PEER

  2 record(s) selected.
```

この場合、1 次ホストは HOSTA で、ステップ 2 (1779 ページ) で使用したコマンドからの出力からすると、 1 次 CF の CPU ロードは 96% であると推測できます。

次のタスク

CPU 使用量が最大容量で実行されていることが分かる場合、プロセッサーを追加するか、クラスター・キャッシング・ファシリティーをアップグレードすると、システム・スループットが向上する場合があります。

注: 複数の論理プロセッサーがあるホストの場合、使用量の数値が 100% を超える可能性があります。例えば、8 個のプロセッサーで構成されるホストの場合、プロセッサー使用量が 800% に達する可能性があります。

DB2 pureScale 環境におけるバッファー・プールのモニター

メンバーによって要求されるデータ・ページがグループ・バッファー・プールまたはローカル・バッファー・プールに見つかる回数を、ディスクから読み取る必要のある回数と比較して調べると、入出力に関連するパフォーマンス上の問題がある場所を特定できます。

一般的に、バッファー・プールが大きければ、必要なデータのページがメモリー内に見つかる可能性も高くなります。

バッファー・プール・アクティビティーに関連したモニター・エレメントの確認および比較を行うと、クラスター・キャッシング・ファシリティーのグループ・バッファー・プール (GBP) と、それぞれのメンバーのローカル・バッファー・プール (LBP) が、どの程度システムのディスク入出力を削減しているのかを理解するのに役立ちます。

DB2 pureScaleのバッファー・プール・アクティビティーを表示するためのモニター・エレメント

IBM DB2 pureScale Feature では、DB2 pureScale インスタンスにおけるバッファー・プール・アクティビティーに関してレポートする多数のモニター・エレメントを使用できます。

グループ・バッファー・プールのモニター・エレメント

以下のモニター・エレメントは、1 次 CF におけるグループ・バッファー・プール (GBP) についての情報を提供します。

- 1292 ページの『pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント』
- 1294 ページの『pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント』

- 1290 ページの『pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』
- 1329 ページの『pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント』
- 1331 ページの『pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント』
- 1327 ページの『pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント』
- 1396 ページの『pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント』
- 1398 ページの『pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント』
- 1394 ページの『pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』
- 1269 ページの『pool_async_data_gbp_indep_pages_found_in_lbp - 非同期 EDU によってローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
- 1270 ページの『pool_async_data_gbp_l_reads - 非同期のグループ・バッファー・プール・データの論理読み取り : モニター・エレメント』
- 1270 ページの『pool_async_data_gbp_p_reads - 非同期のグループ・バッファー・プール・データの物理読み取り : モニター・エレメント』
- 1269 ページの『pool_async_data_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント』
- 1275 ページの『pool_async_index_gbp_indep_pages_found_in_lbp - 非同期 EDU によってローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
- 1276 ページの『pool_async_index_gbp_l_reads - 非同期のグループ・バッファー・プール索引の論理読み取り : モニター・エレメント』
- 1276 ページの『pool_async_index_gbp_p_reads - 非同期のグループ・バッファー・プール索引の物理読み取り : モニター・エレメント』
- 1275 ページの『pool_async_index_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効な索引データ・ページ : モニター・エレメント』
- 1282 ページの『pool_async_xda_gbp_indep_pages_found_in_lbp - 非同期 EDU によってローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属 XML ストレージ・オブジェクト (XDA) ページのモニター・エレメント』
- 1283 ページの『pool_async_xda_gbp_l_reads - グループ・バッファー・プール XDA データの非同期論理読み取り要求 : モニター・エレメント』
- 1283 ページの『pool_async_xda_gbp_p_reads - グループ・バッファー・プール XDA データの非同期物理読み取り要求 : モニター・エレメント』
- 1282 ページの『pool_async_xda_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント』

注: これらのモニター・エレメントは、それぞれのメンバーのバッファー・プールに関するデータを個別にレポートし、集約は実行されません。ローカル・バッファー・プールに関してバッファー・プール使用量情報を集約する場合 (例えば、すべてのローカル・バッファー・プールについての平均ヒット・レートを計算する場合など) には、SUM 集約関数を使用します。

各モニター・エレメントに関する参照トピックをご覧になり、そのモニター・エレメントに関連したデータを調べることが可能なモニター・インターフェースを確認してください。

ローカル・バッファー・プールのためのモニター・エレメント

以下のモニター・エレメントは、DB2 pureScale インスタンス内のそれぞれのメンバーのローカル・バッファー・プールについての情報を提供します。

- 1288 ページの『pool_data_gbp_indep_pages_found _in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属データ・ページのモニター・エレメント』
- 1296 ページの『pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』
- 1326 ページの『pool_index_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール非従属索引ページのモニター・エレメント』
- 1333 ページの『pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント』
- 1393 ページの『pool_xda_gbp_indep_pages _found_in_lbp - ローカル・バッファー・プールで検出されたグループ・バッファー・プール XDA 非従属ページのモニター・エレメント』
- 1403 ページの『pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』
- 1271 ページの『pool_async_data_lbp_pages_found - 非同期のローカル・バッファー・プールの検出データ・ページ : モニター・エレメント』
- 1277 ページの『pool_async_index_lbp_pages_found - 非同期のローカル・バッファー・プールの検出索引ページ : モニター・エレメント』
- 1284 ページの『pool_async_xda_lbp_pages_found - 非同期のローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント』

次の 3 つのモニター・エレメントでレポートされる値は、指定のメンバーのローカル・バッファー・プールに対するディスクからのすべての読み取りを示します。

- 1300 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』
- 1337 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』
- 1405 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

最後の 3 つのモニター・エレメントは、DB2 pureScale 環境*以外* の環境で使用されるモニター・エレメントと同じです。DB2 pureScale 環境のメンバーのローカ

ル・バッファー・プールは、DB2 pureScale 環境以外の環境のデータベースのバッファー・プールと同等なので、これらのエレメントも同じになります。

重要:

- 最後の 3 つのモニター・エレメントでレポートされる値は、GBP で検出されなかったためにローカル・バッファー・プールが GBP 従属ページ (つまり、メンバーが GBP から要求したページ) をディスクから読み取った回数に関連しています。また、一時ページなど GBP 非従属ページ (つまり、メンバーに対してローカルで、メンバーが GBP に従属関係を持たないページ) をディスクから読み取った回数も関係します。
- DB2 pureScale 環境における LBP と GBP 間の関係性のため、バッファー・プール・ヒット率を計算するための数式は、DB2 pureScale 環境以外で使用される数式とは異なります。詳しくは、1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

各モニター・エレメントに関する参照トピックをご覧になり、そのモニター・エレメントに関連したデータを調べることが可能なモニター・インターフェースを確認してください。

DB2 pureScale 環境におけるバッファー・プールのヒット・レートとヒット率

メンバーが必要としたページがディスクではなくメモリー内で見つかる頻度を測定する 1 つの方法は、バッファー・プールのヒット率 を計算することです。バッファー・プール・ヒット率は、要求されたページをデータベース・マネージャーがバッファー・プール内で見つけた回数 (ヒット・レート とも呼ばれます) を、ディスクから読み取る必要があった回数と比較したものです。DB2 pureScale 環境では、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートおよびヒット率はどちらも、全体のアクセス・パフォーマンスにとって重要な要因となります。

ローカル・バッファー・プール (LBP) ヒット率は、メンバーが必要とするページが、ローカル・バッファー・プールで有効な状態で見つかる頻度を示します メンバーの LBP にあるページが LBP にロードされて以降、他のメンバーによって変更されていなければ、そのページは有効な状態にあると見なされます。他のメンバーがそのページを変更した場合 (これは、ページがディスクにキャストされる前に生じる場合があります)、そのページは*無効* であると見なされます。無効なページを持つメンバーで、トランザクションを実行するためにページが必要になった場合、メンバーは CF にアクセスして、有効なバージョンの新規ページを要求する必要があります。

LBP のヒット率が低いと、ページがローカルには見つからず、CF から要求しなければならなかったということを示します。

ただし、DB2 pureScale 環境では、LBP ヒット率を調べても、バッファー・プールに関する一面しか把握できません。ページを取得するためのグループ・バッファー・プール (GBP) が果たす役割と、GBP 自体のヒット率についても考慮する必要があります。メンバーが LBP 内に有効なページ・コピーを見つけることができな

いと、そのページの有効なコピーを GBP で検索するように CF に対して要求を行います。GBP は、以下のいずれかのアクションを実行します。

- ページの有効なコピーがある場合には、要求を出したメンバーに GBP がそれを提供します。
- ない場合には、GBP はディスクからページを読み取る必要があることを要求メンバーに通知します。

また、LBP の使用に関して考慮すべきもう 1 つの事柄は、 GBP 非従属ページという概念です。GBP 非従属ページとは、メンバーの LBP を介してアクセスされるだけで、GBP に存在することがないページのことです。ページを使用する操作、またはページが属するオブジェクトを使用する操作が、ローカル・メンバーのみを使用してアクセスされる場合、そのページは GBP 非従属であると言えます。

グループ・バッファー・プールのヒット率は、有効なローカル・コピーがないページに関してメンバーが要求したそのページがグループ・バッファー・プールで見つかった頻度を、ディスクから読み取る必要があった頻度と比較して示したものです。GBP のヒット率が低ければ、インスタンスにおいてメンバーが必要としたページが GBP で使用可能であった頻度が比較的まれであったことを示します GBP のサイズを増やすと、ヒット・レートおよび全体のパフォーマンスが改善される可能性があります。そのため、メンバーのローカル・バッファー・プール (LBP) におけるデータ・ページのヒット率を計算する場合、メンバーが LBP からページを読み取ろうとした回数を、LBP 内で有効なページが見つからなかった読み取り試行回数と比較して考慮する必要があります。LBP および GBP モニター・エレメントを使用して GBP ヒット・レートを計算する方法の詳細については、 1785 ページの『バッファー・プール・ヒット率の計算数式』を参照してください。

ヒント: ヒット率は様々な要因によって変わります。例えば、データベース内のデータの性質、実行した照会、およびハードウェアやソフトウェアの構成などの要因があります。一般的に、バッファー・プールのヒット率が高ければ、照会パフォーマンスも良好であることを示しています。ヒット率が低いことが分かる場合、または徐々に低下している場合、バッファー・プールのサイズを増やすと役立ちます。グループ・バッファー・プールのサイズを増やすには、CF で **cf_gbp_sz** 構成パラメーターを調整してください。ローカル・バッファー・プールを調整するには、補正が必要なバッファー・プールが属しているメンバーで、**ALTER BUFFERPOOL** ステートメントを実行します。

バッファー・プール・モニター・エレメントのレポート

DB2 pureScale 環境では、他の DB2 環境と同様、それぞれのメンバーが独自のローカル・バッファー・プールに関してレポートを作成します。メンバー間でデータが集計されることはありません。どのメンバーを対象とするのかを考慮し、それに応じてデータを解釈する必要があります。場合によっては、特定のメンバーのヒット率を計算することもできます。あるいは、すべてのメンバーのデータを一緒に調べて、DB2 pureScale 環境全体としてのヒット・レートとヒット率の概要を把握することもできます。

例えば、MON_GET_BUFFERPOOL 表関数を使用して、データ・ページが (**pool_data_gbp_p_reads** モニター・エレメントを使用して) GBP で見つからなかったためにディスクからローカル・バッファー・プールに読み込まれた回数について

のデータを戻す照会を送信する場合、戻すメンバーを指定しないと、以下のような結果が表示されます。

```
MEMBER BP_NAME             POOL_DATA_GBP_P_READS
------ ------------------- --------------------
     0 IBMDEFAULTBP                        408
     0 IBMSYSTEMBP4K                         0
     0 IBMSYSTEMBP8K                         0
     0 IBMSYSTEMBP16K                        0
     0 IBMSYSTEMBP32K                        0
     1 IBMDEFAULTBP                        108
     1 IBMSYSTEMBP4K                         0
     1 IBMSYSTEMBP8K                         0
     1 IBMSYSTEMBP16K                        0
     1 IBMSYSTEMBP32K                        0
     2 IBMDEFAULTBP                        112
     2 IBMSYSTEMBP4K                         0
     2 IBMSYSTEMBP8K                         0
     2 IBMSYSTEMBP16K                        0
     2 IBMSYSTEMBP32K                        0

  15 record(s) selected.
```

重要: 前述の例の場合、一時バッファー・プールに関してレポートされたデータがすべてゼロであることが分かります。これは偶然ではありません。DB2 pureScale インスタンスにおいて、一時オブジェクトと表スペースは関連するメンバーに対してローカルであるためです。CF の GBP は使用しません。

すべてのメンバーにおける結果を把握したい場合には、SUM 集約関数を使用して、すべてのメンバーの数値を一緒に加算します。

```
SELECT  VARCHAR(BP_NAME,15) AS BP_NAME,
        SUM(POOL_DATA_GBP_P_READS) AS TOTAL_P_READS
FROM TABLE(MON_GET_BUFFERPOOL('', -2))
GROUP BY BP_NAME
```

前述の照会によって、以下の出力のような結果が戻ります。

```
BP_NAME         TOTAL_P_READS
--------------- --------------------
IBMDEFAULTBP                     310
IBMSYSTEMBP16K                     0
IBMSYSTEMBP32K                     0
IBMSYSTEMBP4K                      0
IBMSYSTEMBP8K                      0

  5 record(s) selected.
```

バッファー・プール・ヒット率の計算数式

バッファー・プール・ヒット率は、照会で必要とされたデータを外部ストレージから読み取る必要があった頻度と比較して、メモリー内で検出された頻度を示します。ヒット・レートおよびヒット率は、バッファー・プール・モニター・エレメントに基づいた数式を使用して計算できます。

ローカル・バッファー・プール

表 2166. ローカル・バッファー・プール・ヒット率の数式 : これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファー・プール・ヒット率の計算式
データ・ページ	(pool_data_lbp_pages_found - pool_async_data_lbp_pages_found - pool_temp_data_l_reads) / (pool_data_l_reads) × 100
索引ページ	((pool_index_lbp_pages_found - pool_async_index_lbp_pages_found - pool_temp_index_l_reads) / (pool_index_l_reads) × 100

表 2166. ローカル・バッファー・プール・ヒット率の数式 (続き)： これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファー・プール・ヒット率の計算式
一時データ・ページ	((pool_temp_data_l_reads - pool_temp_data_p_reads) / pool_temp_data_l_reads) × 100
一時索引ページ	((pool_temp_index_l_reads - pool_temp_index_p_reads) / pool_temp_index_l_reads) × 100
XML ストレージ・オブジェクト (XDA) ページ	((pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads) × 100
全体のヒット率	((pool_data_lbp_pages_found + pool_index_lbp_pages_found + pool_xda_lbp_pages_found - pool_async_data_lbp_pages_found - pool_async_index_lbp_pages_found - pool_async_xda_lbp_pages_found) / (pool_data_l_reads + pool_index_l_reads + pool_xda_l_reads + pool_temp_data_l_reads + pool_temp_xda_l_reads + pool_temp_index_l_reads)) × 100

ローカル・バッファー・プール (DB2 pureScale 環境)

DB2 pureScale 環境で LBP のヒット率を計算するときには、GBP に依存しないデータ・ページのヒット率も考慮する必要があります。データ・ページがメンバーから要求されると、まず LBP が調べられます。ページが LBP で検出された場合は、**pool_data_lbp_pages_found** モニター・エレメントが増分されます。ページが LBP で検出されなかった場合は、ディスクから直接読み取られ、**pool_data_p_reads** モニター・エレメントが増分されます。ただし、 GBP に依存するページがローカル・バッファー・プールで検出された場合には、 **pool_data_lbp_pages_found** モニターのカウンターも増分されます。このような場合にこれらのページ・アクセス・カウントを区別する唯一の方法として、新規モニター・エレメント **pool_data_gbp_indep_pages_found_in_lbp** が用意されています。

表 2167. GBP には依存しない (LBP) ヒット率の数式： これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファー・プール・ヒット率の計算式
データ・ページ	(pool_data_gbp_indep_pages_found_in_lbp) / (pool_data_gbp_indep_pages_found_in_lbp + pool_data_p_reads + pool_temp_data_p_reads - pool_data_gbp_p_reads) × 100
索引ページ	(pool_index_gbp_indep_pages_found_in_lbp) / (pool_index_gbp_indep_pages_found_in_lbp + pool_index_p_reads + pool_temp_index_p_reads - pool_index_gbp_p_reads) × 100
XML ストレージ・オブジェクト (XDA) ページ	(pool_xda_gbp_indep_pages_found_in_lbp / (pool_xda_gbp_indep_pages_found_in_lbp + pool_xda_p_reads + pool_temp_xda_p_reads - pool_xda_gbp_p_reads) × 100

GBP に依存しないページの場合は、 LBP サイズを調整するとヒット率に影響が生じます。一時ページ・アクセスや NOT LOGGED INITIALLY 操作のような GBP に依存しない特定のタイプの操作では、 GBP に依存しないページ・ヒット率をモニターすることができるでしょう。

別の方法として、GBP に依存するページ・エージェントのローカル・バッファー・プール・ヒット率を計算するために、以下の数式を使用することもできます。

表 2168. GBP に依存する (LBP) ヒット率の数式： これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファー・プール・ヒット率の計算式
データ・ページ	((pool_data_lbp_pages_found - pool_data_gbp_indep_pages_found_in_lbp) / (pool_data_l_reads - pool_data_gbp_indep_pages_found_in_lbp - (pool_data_p_reads - pool_data_gbp_p_reads))) × 100

表 2168. GBP に依存する (LBP) ヒット率の数式 (続き): これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファー・プール・ヒット率の計算式
索引ページ	((pool_index_lbp_pages_found - pool_index_gbp_indep_pages_found_in_lbp) / (pool_index_l_reads - pool_index_gbp_indep_pages_found_in_lbp - (pool_index_p_reads - pool_index_gbp_p_reads))) × 100
XML ストレージ・オブジェクト (XDA) ページ	((pool_xda_lbp_pages_found - pool_xda_gbp_indep_pages_found_in_lbp) / (pool_xda_l_reads - pool_xda_gbp_indep_pages_found_in_lbp - (pool_xda_p_reads - pool_xda_gbp_p_reads))) × 100

両方のバッファー・プールの調整方法を決定したり、 GBP および LBP の調整後に結果を妥当性検査したりする場合に、 LBP ヒット率および GBP ヒット率の両方を比較する必要があります。

グループ・バッファー・プール (DB2 pureScale 環境)

DB2 pureScale 環境におけるグループ・バッファー・プール・ヒット率を計算するための数式は、他の DB2 環境で使用されるヒット率の数式とは異なります。このような相違が生じるのは、クラスター・キャッシング・ファシリティーのおけるグループ・バッファー・プールが、データのページを取得するためにそれぞれのメンバー内のローカル・バッファー・プールを扱う方法に起因しています。バッファー・プール・モニター・エレメントに基づく以下の数式を使用すると、ローカル・バッファー・プールとグループ・バッファー・プールの両方に関するデータ・ページ、索引ページ、XML ストレージ・オブジェクト・ページのヒット率を計算できます。

表 2169. グループ・バッファー・プール (GBP) ヒット率の数式: これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファー・プール・ヒット率の計算式
データ・ページ	((pool_data_gbp_l_reads - pool_data_gbp_p_reads) / pool_data_gbp_l_reads) × 100
索引ページ	((pool_index_gbp_l_reads - pool_index_gbp_p_reads) / pool_index_gbp_l_reads) × 100
XML ストレージ・オブジェクト (XDA) ページ	((pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads) × 100
全体のヒット率	((pool_data_gbp_l_reads + pool_index_gbp_l_reads + pool_xda_gbp_l_reads - pool_data_gbp_p_reads - pool_index_gbp_p_reads - pool_xda_gbp_p_reads) / (pool_data_gbp_l_reads + pool_index_gbp_l_reads + pool_xda_gbp_l_reads)) x 100

バッファー・プール・ヒット率を計算するための前述の数式に加え、下記の数式を使用すると、プリフェッチされた時間ページが GBP で検出されたパーセンテージが示されます。

データ・ページのプリフェッチ

((pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads) / pool_async_data_gbp_l_reads) × 100

索引ページのプリフェッチ

((pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads) / pool_async_index_gbp_l_reads) × 100

XML ストレージ・オブジェクト (XDA) ページのプリフェッチ

((pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads) / pool_async_xda_gbp_l_reads) × 100

DB2 pureScale 環境におけるバッファー・プール・ヒット率の計算

DB2 pureScale インスタンスのバッファー・プール・ヒット率を計算すると、入出力効率が向上するようにバッファー・プールを調整する箇所を把握するのに役立ちます。

始める前に

対象となる率を判別します。インスタンスにおけるすべてのメンバーの率を確認する場合には、SUM 集約関数を使用してすべてのメンバーのデータを集約する SQL の作成を考慮してください。特定のメンバーのデータのみを確認する場合には、MON_GET_BUFFERPOOL 表関数でデータを確認するメンバーを指定できます。

手順

バッファー・プール・ヒット率を計算するには、以下のステップに従ってください。

1. 必要なモニター・エレメントに関する情報を取得します。 以下の例では、MON_GET_BUFFERPOOL 表関数を使用して、GBP のデータ・ページのヒット率を計算するのに必要な値が含まれるモニター・エレメント、つまり **pool_data_gbp_l_reads** と **pool_data_gbp_p_reads** を取り出します。

   ```
   SELECT varchar(bp_name,20) AS bp_name,
          pool_data_gbp_l_reads,
          pool_data_gbp_p_reads,
          member
   FROM TABLE(MON_GET_BUFFERPOOL('',-2))
   ```

   前述の照会によって、以下の例のようなデータが戻ります。

   ```
   BP_NAME              POOL_DATA_GBP_L_READS POOL_DATA_GBP_P_READS MEMBER
   -------------------- --------------------- --------------------- ------
   IBMDEFAULTBP                       1814911                456990      1
   IBMSYSTEMBP4K                            0                     0      1
   IBMSYSTEMBP8K                            0                     0      1
   IBMSYSTEMBP16K                           0                     0      1
   IBMSYSTEMBP32K                           0                     0      1
   IBMDEFAULTBP                       1807959                455287      3
   IBMSYSTEMBP4K                            0                     0      3
   IBMSYSTEMBP8K                            0                     0      3
   IBMSYSTEMBP16K                           0                     0      3
   IBMSYSTEMBP32K                           0                     0      3
   IBMDEFAULTBP                       1813932                455225      2
   IBMSYSTEMBP4K                            0                     0      2
   IBMSYSTEMBP8K                            0                     0      2
   IBMSYSTEMBP16K                           0                     0      2
   IBMSYSTEMBP32K                           0                     0      2
   IBMDEFAULTBP                       1113396                278845      0
   IBMSYSTEMBP4K                            0                     0      0
   IBMSYSTEMBP8K                            0                     0      0
   IBMSYSTEMBP16K                           0                     0      0
   IBMSYSTEMBP32K                           0                     0      0

     20 record(s) selected.
   ```

   重要: 前述の例の場合、一時バッファー・プールに関してレポートされたデータがすべてゼロであることが分かります。これは偶然ではありません。DB2 pureScale インスタンスにおいて、一時オブジェクトと表スペースは関連するメンバーに対してローカルであるためです。CF の GBP は使用しません。
2. これらのモニター・エレメントに関して戻された値を使用して、ヒット率を計算します。 GBP のヒット率を計算する数式は、以下のとおりです (パーセンテージ表記)。

((pool_data_gbp_l_reads - pool_data_gbp_p_reads) ÷ pool_data_gbp_l_reads) × 100

ステップ 1 (1788 ページ) のモニター・エレメントに関して戻されたデータを使用すると、以下のようになります。

```
(((1,814,911+1,807,959  + 1,813,932+1,113,396) - (456,990+455,287 + 455,225+278,845)) ÷ (1,814,911+1,807,959 + 1,813,932+1,113,396)) × 100
= ((6,550,198 - 1,646,347) ÷ 6,550,198) × 100
= 74.9%
```

この例の場合、GBP のヒット率は 74.9% です。

注: 照会の出力で表示されている値は、例示のみを目的としています。

例

例 1: すべてのメンバーにおける全体的なヒット・レートを検出する

この例は前述の手順と似ていますが、すべてのメンバーにおける全体的なヒット・レートを提供するために集約関数が使用されています。

```
SELECT VARCHAR(BP_NAME,20) AS BP,
       SUM(POOL_DATA_GBP_L_READS) AS POOL_DATA_GBP_L_READS,
       SUM(POOL_DATA_GBP_P_READS) AS POOL_DATA_GBP_L_READS
FROM TABLE(MON_GET_BUFFERPOOL('',-2))
GROUP BY BP_NAME
```

結果:

```
BP                   POOL_DATA_GBP_L_READS POOL_DATA_GBP_L_READS
-------------------- --------------------- ---------------------
IBMDEFAULTBP                       6550198               1646347
IBMSYSTEMBP16K                           0                     0
IBMSYSTEMBP32K                           0                     0
IBMSYSTEMBP4K                            0                     0
IBMSYSTEMBP8K                            0                     0

  5 record(s) selected.
```

例 2: すべてのデータ・ページ、索引ページ、XML ストレージ・オブジェクト (XDA) ページの GBP ヒット率を判別する

この例では **MON_GET_BUFFERPOOL** 表関数を使用して、必要なモニター・エレメントに含まれるデータを取得し、各メンバーのヒット率を計算します。すべてのデータ・ページ、索引ページ、XDA ページの GBP ヒット率を計算するには、以下の数式を使用します。

((pool_data_gbp_l_reads + pool_index_gbp_l_reads+pool_xda_gbp_l_reads)
- (pool_data_gbp_p_reads + pool_index_gbp_p_reads+pool_xda_gbp_p_reads))
÷ (pool_data_gbp_l_reads + pool_index_gbp_l_reads+pool_xda_gbp_l_reads) × 100

```
WITH BPMETRICS AS (
     SELECT BP_NAME,
            POOL_DATA_GBP_L_READS +
              POOL_INDEX_GBP_L_READS +
              POOL_XDA_GBP_L_READS
            AS LOGICAL_READS,
            POOL_DATA_GBP_P_READS +
              POOL_INDEX_GBP_P_READS +
              POOL_XDA_GBP_P_READS
            AS PHYSICAL_READS,
            MEMBER
     FROM TABLE(MON_GET_BUFFERPOOL('',-2)) AS METRICS)
     SELECT VARCHAR(BP_NAME,20) AS BP_NAME,
            LOGICAL_READS,
            PHYSICAL_READS,
```

```
     CASE WHEN LOGICAL_READS > 0
          THEN DEC(((
            FLOAT(LOGICAL_READS) - FLOAT(PHYSICAL_READS)) /
            FLOAT(LOGICAL_READS))
            * 100,5,2)
          ELSE NULL END AS HIT_RATIO,
          MEMBER
     FROM BPMETRICS
```

結果:

```
BP_NAME             LOGICAL_READS        PHYSICAL_READS       HIT_RATIO MEMBER
------------------- -------------------- -------------------- --------- ------
IBMDEFAULTBP                  5730213               617628     89.22      1
IBMSYSTEMBP4K                       0                    0         -      1
IBMSYSTEMBP8K                       0                    0         -      1
IBMSYSTEMBP16K                      0                    0         -      1
IBMSYSTEMBP32K                      0                    0         -      1
IBMDEFAULTBP                  5724845               615395     89.25      3
IBMSYSTEMBP4K                       0                    0         -      3
IBMSYSTEMBP8K                       0                    0         -      3
IBMSYSTEMBP16K                      0                    0         -      3
IBMSYSTEMBP32K                      0                    0         -      3
IBMDEFAULTBP                  5731714               615814     89.25      2
IBMSYSTEMBP4K                       0                    0         -      2
IBMSYSTEMBP8K                       0                    0         -      2
IBMSYSTEMBP16K                      0                    0         -      2
IBMSYSTEMBP32K                      0                    0         -      2
IBMDEFAULTBP                  5024809               409159     91.85      0
IBMSYSTEMBP4K                       0                    0         -      0
IBMSYSTEMBP8K                       0                    0         -      0
IBMSYSTEMBP16K                      0                    0         -      0
IBMSYSTEMBP32K                      0                    0         -      0

  20 record(s) selected.
```

例 3: SUM 集約関数を使用して全体的なヒット率を計算する

以下のように SUM 集約関数を使用しても、すべてのメンバーにおける全体的なヒット率を計算できます。

```
WITH BPMETRICS AS (
     SELECT SUM(POOL_DATA_GBP_L_READS) +
       SUM(POOL_INDEX_GBP_L_READS) +
       SUM(POOL_XDA_GBP_L_READS)
     AS LOGICAL_READS,
     SUM(POOL_DATA_GBP_P_READS) +
       SUM(POOL_INDEX_GBP_P_READS) +
       SUM(POOL_XDA_GBP_P_READS)
     AS PHYSICAL_READS
     FROM TABLE(MON_GET_BUFFERPOOL('',-2)) AS METRICS)
     SELECT LOGICAL_READS,
            PHYSICAL_READS,
     CASE WHEN LOGICAL_READS > 0
     THEN DEC(((FLOAT(LOGICAL_READS) - FLOAT(PHYSICAL_READS)) /
       FLOAT(LOGICAL_READS))
       * 100,5,2)
     ELSE NULL END AS HIT_RATIO
FROM BPMETRICS
```

結果:

```
LOGICAL_READS        PHYSICAL_READS       HIT_RATIO
-------------------- -------------------- ---------
             22211581              2255996     89.84

  1 record(s) selected.
```

次のタスク

ヒット率が低いように思われる場合、または徐々に低下している場合、メンバーまたは CF の、あるいはその両方のバッファー・プール・サイズを増やすことができます。DB2 pureScale インスタンスにある LBP に対するヒット・レートが予想されるよりも全体的に低い場合、各メンバーのバッファー・プールのサイズは異なる可能性があるので、それぞれのメンバーのヒット・レートを別個に調べてください。

あるメンバーの LBP のサイズが小さいと、そのインスタンスの平均ヒット・レートに大きな影響を与える可能性があります。

ヒント: ヒット率は様々な要因によって変わります。例えば、データベース内のデータの性質、実行した照会、およびハードウェアやソフトウェアの構成などの要因があります。一般的に、バッファー・プールのヒット率が高ければ、照会パフォーマンスも良好であることを示しています。ヒット率が低いことが分かる場合、または徐々に低下している場合、バッファー・プールのサイズを増やすと役立ちます。グループ・バッファー・プールのサイズを増やすには、CF で **cf_gbp_sz** 構成パラメーターを調整してください。ローカル・バッファー・プールを調整するには、補正が必要なバッファー・プールが属しているメンバーで、**ALTER BUFFERPOOL** ステートメントを実行します。

DB2 pureScale 環境におけるロック・モニターの概要

従来の DB2 環境と同様、データ整合性と高水準の並行性を維持するには DB2 pureScale 環境におけるロック管理が不可欠です。

DB2 pureScale 環境におけるメンバー間のロックは、クラスター・キャッシング・ファシリティーのグローバル・ロック・マネージャー (GLM) コンポーネントによって管理されます。DB2 pureScale 環境でのロックのモニターでは、特定のメンバーで保持されている可能性のあるロックだけを検討するのではなく、メンバー間におけるロック待機も検討対象です。

DB2 pureScale 環境では、同じデータを異なるメンバーが作業対象とすることがあるため、別のタイプのデータとの競合が起こり得ます。2 つのメンバーが同じオブジェクトを更新しようとするとこれが生じます。メンバーがオブジェクトのロックを必要とする場合、そのメンバーのローカル・ロック・マネージャー (LLM) コンポーネントがグローバル・ロック・マネージャー (GLM) と協働します。LLM が該当のオブジェクトのロックをまだ保持していない場合、LLM は GLM にロックを要求します。このように、GLM は他のメンバーからのロック要求を調整します。

DB2 pureScale インスタンスをグローバル・レベルで表示すると、**locks_held**、または **lock_wait_time** などのモニター・エレメントは、インスタンス内のロック、メンバー内のロックとメンバー間のロックのすべての ロックに関するデータをレポートします。特に、DB2 pureScale Feature に追加されたモニター・エレメントを使用すると、メンバー間のロック待機のみを調べることができます。

メンバー間におけるロック要求

DB2 pureScale 環境では、あるメンバーのアプリケーションが、別のメンバーによって現在ロックされているオブジェクトに関するロックを要求する場合があります。DB2 pureScale Feature では、メンバー間におけるロックについての特定の情報を報告するモニター・エレメントが導入されました。

従来の DB2 環境と同様に、DB2 pureScale 環境のメンバー内 で処理を行うと、あるアプリケーションが他のアプリケーションによってその時点で実行されている操作と両立しない操作を実行しようとすると、オブジェクトにロックが掛かる可能性があります。これは、例えば、2 つのアプリケーションが同時に同じデータ行を更新しようとすると生じ得ます。ロック関連情報を表示するロック・イベント・モニ

ターを使用すると、特定のメンバー内でこの種のロックが生じる頻度をモニターできます。また DB2 pureScale 環境では、あるメンバーが別のメンバーのアプリケーションによってその時点でロックされているオブジェクトに関するロックを要求すると、メンバー*間* におけるロック待機が生じる場合があります。そのため、DB2 pureScale 環境では、個々のメンバーのロックを調べるだけではなく、メンバー間のロック情報も調べたい場合があります。

メンバー間のロック待機

以下のモニター・エレメントでは、アプリケーションが別のメンバーが保持しているロックを待機している時間だけ を報告します。

- lock_wait_time_global
- lock_wait_time_global_top
- lock_waits_global
- lock_timeouts_global

これらのモニター・エレメントによって報告される時間は、**lock_wait_time** と **lock_wait_time_top** のモニター・エレメントが DB2 pureScale インスタンス全体としての観点で表示されるときに、それら2 つの全体的なモニター・エレメントの一部として含まれます。同様に、ロック待機とタイムアウトのカウントも、**lock_waits** と **lock_timeouts** のモニター・エレメントがインスタンス全体のレベルで表示されるときに、それら 2 つのモニター・エレメントの一部として報告されます。

以下のシナリオは、これらのメンバー間の関係、またはグローバル・ロック待機モニター・エレメントとメンバー内のロックについて報告するモニター・エレメントの間の関係について示しています。

1. メンバー 1 のアプリケーション 1 が行で共有 (S) ロックを保持しています。
2. メンバー 2 のアプリケーション 2 が同じ行で排他的 (X) ロックを要求しています。アプリケーション 2 は、アプリケーション 1 がその行を現在ロックしているので待機させられます。
3. 2 ミリ秒後、メンバー 2 のアプリケーション 3 は同じ行で排他的 (X) ロックを要求します。アプリケーション 3 も待機させられます。
4. 8 ミリ秒後、アプリケーション 1 はロックを解放し、アプリケーション 2 がロックを獲得します。
5. 5 ミリ秒後、アプリケーション 2 はロックを解放し、アプリケーション 3 がロックを獲得します。

lock_wait_time によって報告される合計ロック待機時間は 23 ミリ秒です。アプリケーション 2 は合計で 10 ミリ秒待機し、アプリケーション 3 は合計で 13 ミリ秒待機しました。ただし、メンバー*間* での合計ロック待機時間 **lock_wait_time_global** は 10 ミリ秒です。この待機時間は全体のロック待機時間の一部でしかなく、あるメンバーが別のメンバーによって保持されているロックを待機していた時間だからです。

同様に、**lock_waits_global** によって報告されるロック保持数のカウントは 1 です。アプリケーション 1 に対するアプリケーション 2 の待機は、1 つのメンバーが別のメンバーを待機しているものとしてカウントされるからです。アプリケーシ

ョン 3 が待機時間の一部としてメンバー 1 のアプリケーションによって待機されている場合であっても、このロック待機はメンバー間のロック待機としてカウントされません。メンバー 2 のローカル・ロック・マネージャーからロックを取得したからです。

ロックを保持しているアプリケーションに関するレポート作成

ロック・イベント・モニターは、ロックを保持しているアプリケーションに関する情報を表示します。通常、ロック・イベント・モニターは、アプリケーションを実行しているメンバーに関わりなくアプリケーション情報をレポートします。ただし、アプリケーションがリモート・メンバーで実行されていると、ロックを保持しているアプリケーションが判別できないということが稀に生じます。次の例を考慮してください。

```
ALTER WORKLOAD finance COLLECT LOCK WAIT DATA WITH HISTORY AND VALUES
   FOR LOCKS WAITING MORE THAN 5 SECONDS
```

このステートメントにより、『finance』 ワークロードで保持されているロックが 5 秒を超える場合、ロック・イベント・モニターに記録されるようになります。以下のシナリオについて考慮します。

- ロックが、メンバー 1 のアプリケーション 1 によって保持されています。
- 5 秒後、システムは、ロック・イベント・モニターにアプリケーションとメンバーに関する情報も含めロックについての情報を書き込もうとします。ただし、ロックがアプリケーション 1 によって解放され、同じオブジェクトに対するロックが別のメンバーの異なるアプリケーションに対してすぐに付与されると、ロックを保持しているアプリケーションは「収集にロック・ホルダーが使用できませんでした (Lock holder was unavailable for collection)」と記録される場合があります。

またロックを保持しているメンバーで障害が発生すると、ロック・イベント・モニターは、必要なデータに関してロックを保持しているメンバーで実行されているアプリケーションについての情報をレポートできません。

さらに、非推奨のモニタリング・フィーチャーを使用する場合、ロックを保持しているアプリケーションについてのレポート作成に関していくつかの制限があります。詳しくは、 1801 ページの『スナップショット・モニターによるロックのモニター』を参照してください。

メンバー間のロックを表示するためのモニター・エレメント

IBM DB2 pureScale Feature では、多数の新しいモニター・エレメントが追加され、それを DB2 pureScale 環境で使用すると、ロックをモニターできます。こうしたモニター・エレメントは、DB2 pureScale インスタンスのメンバー間におけるロック待機についての特定の情報をレポートします。

ロック関連のモニター・エレメント

以下のモニター・エレメントは、メンバー間のロック待機とロック・エスカレーションについての情報を取得するために使用できます。

- 1162 ページの『lock_waits_global - グローバル・ロック待機 : モニター・エレメント』

- 1157 ページの『lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント』
- 1159 ページの『lock_wait_time_global_top - 最長グローバル・ロック待機時間 : モニター・エレメント』
- 1151 ページの『lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント』
- 1138 ページの『lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント』
- 1136 ページの『lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント』
- 1135 ページの『lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント』

以下のエレメントは、ロックに直接関連するわけではありませんが、DB2 pureScale インスタンス内のステートメント、作業単位、ワークロードなどが、どの程度クラスター・キャッシング・ファシリティーで待機しているかに関する情報を取得するのに使用できます。

- 886 ページの『cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント』
- 887 ページの『cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント』

cf_wait_time エレメントは、クラスター・キャッシング・ファシリティーが要求にサービス提供するためにステートメント、作業単位、ワークロードが待機しなければならなかった期間を示します。このエレメントを cf_waits (cf_wait_time ÷ cf_waits) と組み合わせて使用して、要求ごとの平均待機時間が分かります。一般に、クラスター・キャッシング・ファシリティーがサービス提供する要求の平均待機時間は、数ミリ秒程度です。それよりもかなり長い場合 (つまり、桁が違う場合)、InfiniBand のセットアップに問題がある可能性があります。あるいは、クラスター・キャッシング・ファシリティーに要求が殺到し、処理しきれないのかもしれません。

各モニター・エレメントに関する参照トピックをご覧になり、そのモニター・エレメントに関連したデータを調べることが可能なモニター・インターフェースを確認してください。

また、ロック・イベント・モニターによってサポートされているその他のモニター・エレメントを使用すると、メンバー内 のロックに関する情報を表示できます。

ページ再利用

DB2 pureScale インスタンスにおいてメンバーが異なると、他のメンバーが既に使用しているデータのページに対するアクセスが必要となる場合があります。あるメンバーが要求を行い、別のメンバーが既に使用しているページに関して権限付与するプロセスは、*ページ再利用* と呼ばれます。

特定のデータ・ページに異なるメンバーがアクセスする必要がある場合、クラスター・キャッシング・ファシリティーがそのページにアクセスするメンバーとタイミングを管理します。場合によっては、クラスター・キャッシング・ファシリティー

(CF とも呼ばれます) は、あるメンバーに、他のメンバーのページに関して、その他のメンバーが処理を終える前であっても、*再利用* する許可を与えることがあります。以下に、ページ再利用が生じる方法の例を示します。

M1 と M2 という 2 つのメンバーがあり、どちらも同じデータ・ページ上にある異なる 2 つの行を更新しようとしています。

1. M1 はデータ・ページの R1 という行で更新を行っています。そのデータ行が含まれるページに対する排他的アクセスが権限付与されています。
2. M2 が、行 R2 を更新するために同じページに対する排他的アクセスを必要とします。この要求を、CF に渡します。 M2 は、要求が処理されるのを待機します。
3. CF は、メンバー M1 が既にこのページに対する排他的アクセスを持っていることを把握します。そのため、M1 にページを再利用する要求を発行します。その間、M2 は待機します。
4. M1 は再利用要求を処理するために、そのページを GBP に再び書き込み、ページを解放します (M1 が行または表に関するロックを持っている場合には、それを保持します)。
5. CF はそのページに対するアクセス権限を M2 に付与します。M2 は GBP からそのページを読み取り、そのページを必要とする操作を実行します。

手順 4 に記されている点に注意してください。更新のために行または表に関してあるメンバーが持っているロックは、他のメンバーが再利用し、該当の作業単位が終了する前にそのページの使用を開始する場合であっても、その作業単位が完了するまでは保持されます。このようにして、異なるメンバーが同一のデータ・ページを処理する場合であっても、ロック整合性が失われることなく作業が可能です。2 つのメンバーが同一のデータ行に対して両立しない行ロックを必要とする場合、ロック管理がどちらかのメンバーによって行われる場合と同様に、一方のメンバーが処理を完了しないと、もう一方のメンバーは処理を続行できません。

ページ再利用のためのモニター・エレメント

IBM DB2 pureScale Feature インスタンスでは、DB2 pureScale でページ再利用が生じた頻度をモニターするのに使用可能な新しいモニター・エレメントが多数追加されています。

ページ再利用関連のモニター・エレメント

以下のモニター・エレメントを使用して、DB2 pureScale インスタンスでページ再利用が生じた頻度に関する情報を取得できます。

- 1252 ページの『page_reclaims_x - ページ再利用の排他的アクセス : モニター・エレメント』
- 1253 ページの『page_reclaims_initiated_s - 共有アクセスで開始されたページ再利用 : モニター・エレメント』
- 1515 ページの『spacemappage_page_reclaims_x - スペース・マップ・ページ再利用の排他的アクセス : モニター・エレメント』
- 1516 ページの『spacemappage_page_reclaims_s - スペース・マップ・ページ再利用の共有アクセス : モニター・エレメント』

- 1253 ページの『page_reclaims_initiated_x - 排他的アクセスで開始されたページ再利用 : モニター・エレメント』
- 1253 ページの『page_reclaims_initiated_s - 共有アクセスで開始されたページ再利用 : モニター・エレメント』
- 1516 ページの『spacemappage_page_reclaims_initiated_x - 排他的アクセスで開始されたスペース・マップ・ページ再利用 : モニター・エレメント』
- 1517 ページの『spacemappage_page_reclaims_initiated_s - 共有アクセスで開始されたスペース・マップ・ページ再利用 : モニター・エレメント』
- 1441 ページの『reclaim_wait_time - 再利用待機時間 : モニター・エレメント』
- 1517 ページの『spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント』

各モニター・エレメントに関する参照トピックをご覧になり、そのモニター・エレメントに関連したデータを調べることが可能なモニター・インターフェースを確認してください。

メンバー間のページ再利用のモニター

特定のアプリケーションまたはステートメントが時間を費やしている箇所を調べる際、ロック待機に費やした時間に加えて、あるページが別のメンバーによって使用されている場合にはそのページが使用可能になるのを DB2 pureScale 環境内のアプリケーションまたはステートメントが待機しなければならない場合もあることを念頭に置いてください。

ページ再利用モニター・エレメントを使用すると、このタイプの待機がシステムのスループットにどの程度影響を及ぼしているかを確認できます。

このタスクについて

ページ再利用統計を表示するには、MON_GET_PAGE_ACCESS_INFO 表関数を使用します。この表関数は、他のメンバーが現在使用しているページをメンバーが要求する頻度と、メンバーが他のメンバーが要求した際にそうしたページを解放した頻度の両方に関するオブジェクト・レベルの情報を戻します。また、関係する待機時間も取得できます。

手順

1. 結果を表示する対象となるページのタイプを判別します。 以下の例の場合、page_reclaims_x モニター・エレメントと page_reclaims_s モニター・エレメントを使用して、すべてのデータ・ページと索引ページに関して再利用されたページの回数情報を取得します。
2. MON_GET_PAGE_ACCESS_INFO 表関数を使用する SQL ステートメントを作成します。 例えば、すべてのメンバーに関するデータ・ページと索引ページの情報を取得するには、以下のようなステートメントを作成します。

```
SELECT MEMBER,
       VARCHAR(TABNAME,30) AS TABLE,
       VARCHAR(OBJTYPE,8) AS OBJTYPE,
       PAGE_RECLAIMS_X,
```

```
       PAGE_RECLAIMS_S
FROM TABLE(MON_GET_PAGE_ACCESS_INFO('DTW','', -2))
WHERE PAGE_RECLAIMS_X !=0 OR  PAGE_RECLAIMS_S !=0
ORDER BY MEMBER ASC, PAGE_RECLAIMS_X ASC
```

3. この照会を実行します。 この場合、戻される結果は以下のようになります。

```
MEMBER TABLE                          OBJTYPE  PAGE_RECLAIMS_X      PAGE_RECLAIMS_S
------ ------------------------------ -------- -------------------- --------------------
     0 CUSTOMER                       TABLE                     196                    0
     0 STOCK_1_250                    TABLE                     213                    0
     0 STOCK_1251_1500                TABLE                     237                    0
     0 STOCK_251_500                  TABLE                     239                    0
     0 STOCK_501_750                  TABLE                     245                    0
     0 STOCK_1751_2000                TABLE                     253                    0
     0 STOCK_2001_2250                TABLE                     254                    0
     0 STOCK_751_1000                 TABLE                     259                    0
     0 STOCK_1501_1750                TABLE                     269                    0
     0 STOCK_2251_2500                TABLE                     274                    0
     0 STOCK_251_500                  INDEX                     276                 2934
     0 STOCK_1001_1250                TABLE                     280                    0
     0 STOCK_1501_1750                INDEX                     284                 3070
     0 STOCK_501_750                  INDEX                     294                 3029
     0 STOCK_1_250                    INDEX                     296                 2916
     0 STOCK_751_1000                 INDEX                     301                 3056
     1 STOCK_1001_1250                TABLE                     247                    0
     1 STOCK_501_750                  TABLE                     255                    0
     1 STOCK_751_1000                 TABLE                     257                    0
     1 STOCK_1501_1750                TABLE                     257                    0
     1 STOCK_251_500                  INDEX                     287                 2921
     1 STOCK_1_250                    INDEX                     292                 2916
     1 STOCK_751_1000                 INDEX                     316                 3190
     1 STOCK_501_750                  INDEX                     319                 2956
     1 ORDERS                         INDEX                   42434                 1416
     1 ORDER_LINE                     INDEX                  116107                 3731
     2 CUSTOMER                       TABLE                     180                    0
     2 STOCK_2001_2250                TABLE                     221                    0
     .                                  .                       .                      .
     .                                  .                       .                      .
     .                                  .                       .                      .
     2 STOCK_1501_1750                TABLE                     240                    0
     2 STOCK_2251_2500                TABLE                     247                    0
     2 STOCK_1251_1500                TABLE                     268                    0
     2 STOCK_251_500                  INDEX                     276                 2976
     2 STOCK_1_250                    INDEX                     284                 2846
     2 STOCK_501_750                  TABLE                     285                    0
     2 STOCK_501_750                  INDEX                     293                 3143
     2 DISTRICT                       TABLE                   18402                    0
     2 ORDERS                         INDEX                   41581                 1474
     2 ORDER_LINE                     INDEX                  114442                 3815
     3 CUSTOMER                       TABLE                     159                    0
     3 STOCK_251_500                  TABLE                     226                    0
     3 ORDERS                         INDEX                   42192                 1340
     3 ORDER_LINE                     INDEX                  115459                 3871

  112 record(s) selected.
```

注: この照会の長い出力の一部は除外されていて、垂直方向の省略符号によってそのことが示されています。

タスクの結果

前述の例の場合、データ・ページと索引ページに関する情報が別個に戻されていることが分かります。また、スキーマが指定されているので、レポート対象から、SYSIBM スキーマに関連したオブジェクトのデータが除外されています。

例

例 1: ページ再利用の待機時間を取得する

以下の SQL によって、すべてのメンバーにおいて再利用された合計ページ数と合計待機時間が取得されます。

```
SELECT
     SUM(PAGE_RECLAIMS_X+PAGE_RECLAIMS_S+SPACEMAPPAGE_PAGE_RECLAIMS_X
       +SPACEMAPPAGE_PAGE_RECLAIMS_S)AS PAGE_RECLAIMS,
     SUM(RECLAIM_WAIT_TIME) AS RECLAIM_WAIT_TIME
FROM TABLE(MON_GET_PAGE_ACCESS_INFO('','', -2))
```

この照会の結果は、以下の例のようになります。

```
PAGE_RECLAIMS        RECLAIM_WAIT_TIME
-------------------- --------------------
                 156                   91

  1 record(s) selected.
```

(待機時間は、ミリ秒単位でレポートされます)

例 2: 再利用されたページ数が多い上位 10 個の表を表示する

この例では、ページ再利用に関係する表オブジェクトを確認する方法を示しています。

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA,
       SUBSTR(TABNAME,1,20) AS TABNAME,
       RECLAIM_WAIT_TIME,
       MEMBER,
       SUBSTR(OBJTYPE,1,10) AS OBJTYPE
FROM TABLE(MON_GET_PAGE_ACCESS_INFO(NULL,NULL,-2))
WHERE RECLAIM_WAIT_TIME > 0
 ORDER BY RECLAIM_WAIT_TIME DESC
 FETCH FIRST 10 ROWS ONLY
```

結果:

```
TABSCHEMA TABNAME              RECLAIM_WAIT_TIME    MEMBER OBJTYPE
--------- -------------------- -------------------- ------ ----------
DTW       ORDER_LINE                        1307192      1 INDEX
DTW       ORDER_LINE                        1250134      2 INDEX
DTW       ORDER_LINE                        1249452      0 INDEX
DTW       ORDER_LINE                        1159741      3 INDEX
DTW       DISTRICT                           827598      0 TABLE
DTW       DISTRICT                           785354      2 TABLE
DTW       DISTRICT                           767148      1 TABLE
DTW       DISTRICT                           687608      3 TABLE
DTW       ORDERS                             556538      0 INDEX
DTW       ORDERS                             539858      2 INDEX

  10 record(s) selected.
```

(待機時間は、ミリ秒単位でレポートされます)

例 3: 再利用されたページ数が多くなった原因となる 10 個のステートメントを表示する

この照会は、前述の例の変化形です。この場合、照会は再利用されたページ数が多い 10 個のステートメントを戻します。

```
SELECT SUBSTR(STMT_TEXT,1,50) AS STMT_TEXT,
       RECLAIM_WAIT_TIME
FROM TABLE(MON_GET_PKG_CACHE_STMT('D',NULL,NULL,-2))TABLE
       WHERE RECLAIM_WAIT_TIME > 0
ORDER BY RECLAIM_WAIT_TIME DESC
FETCH FIRST 10 ROWS ONLY
```

結果:

```
STMT_TEXT                                          RECLAIM_WAIT_TIME
-------------------------------------------------- --------------------
Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03                796668
Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03                785863
Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03                746521
Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03                623461
Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?                610602
Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?                522899
Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?                518076
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID =                 419022
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID =                 406028
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID =                 406006

  10 record(s) selected.
```

(待機時間は、ミリ秒単位でレポートされます)

例 4: 再利用されたページ数が多くなった原因となる 10 個のステートメントを、各ステートメントの実行の平均待機時間とともに表示する

前述の例の場合、待機時間は、ステートメントごとの全体の値として示されました。この照会の場合、ステートメントの中には何度も実行されるものがあるかもしれないという事実は考慮に入れていません。以下の例では、上位 10 個のステートメントそれぞれの実行に関する平均待機時間を調べる方法が示されています。

```
SELECT SUBSTR(STMT_TEXT,1,75) AS STMT_TEXT,
       NUM_EXECUTIONS,
       RECLAIM_WAIT_TIME,
       DEC(FLOAT(RECLAIM_WAIT_TIME)/FLOAT(NUM_EXECUTIONS),10,8)
         AS AVG_WAIT_PEREXEC
FROM TABLE(MON_GET_PKG_CACHE_STMT('D',NULL,NULL,-2))TABLE
WHERE RECLAIM_WAIT_TIME > 0
ORDER BY AVG_WAIT_PEREXEC DESC
FETCH FIRST 10 ROWS ONLY
```

結果:

```
STMT_TEXT                                                                NUM_EXECUTIONS       RECLAIM_WAIT_TIME    AVG_WAIT_PEREXEC
------------------------------------------------------------------------ -------------------- -------------------- ----------------
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?                    157173               419497       2.66901439
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?                    155752               397870       2.55450973
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?                    155352               385613       2.48218883
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?                    155151               347847       2.24199006
Insert into ORDERS values (?, ?, ?, ?, ?, ?, ?, ?)                                  157173               259076       1.64834927
Insert into ORDERS values (?, ?, ?, ?, ?, ?, ?, ?)                                  155752               253548       1.62789562
Insert into ORDERS values (?, ?, ?, ?, ?, ?, ?, ?)                                  155352               232300       1.49531386
Insert into ORDERS values (?, ?, ?, ?, ?, ?, ?, ?)                                  155151               219607       1.41544044
Delete from NEW_ORDER where NO_W_ID = ? and NO_D_ID = ? and NO_O_ID = ?              152968               106525       0.69638747
Delete from NEW_ORDER where NO_W_ID = ? and NO_D_ID = ? and NO_O_ID = ?              152591               101367       0.66430523

  10 record(s) selected.
```

(待機時間は、ミリ秒単位でレポートされます)

この照会の若干異なるバージョンでは、各ステートメントの実行に要した期間が示されています。

```
SELECT SUBSTR(STMT_TEXT,1,75) AS STMT_TEXT,
       NUM_EXECUTIONS,
       RECLAIM_WAIT_TIME,
       DEC(FLOAT(RECLAIM_WAIT_TIME)/FLOAT(NUM_EXECUTIONS),10,8) AS AVG_EXEC_TIME
FROM TABLE(MON_GET_PKG_CACHE_STMT('D',NULL,NULL,-2))TABLE
WHERE RECLAIM_WAIT_TIME > 0
ORDER BY RECLAIM_WAIT_TIME DESC
FETCH FIRST 10 ROWS ONLY
```

結果:

```
STMT_TEXT                                                                    NUM_EXECUTIONS       RECLAIM_WAIT_TIME    AVG_EXEC_TIME
---------------------------------------------------------------------------- -------------------- -------------------- -------------
Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03, S_DIST_04, S_DIST_05, S             1555470               755544    0.48573357
Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03, S_DIST_04, S_DIST_05, S             1554405               754231    0.48522167
Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03, S_DIST_04, S_DIST_05, S             1570256               741047    0.47192750
Select S_QUANTITY, S_DIST_01, S_DIST_02, S_DIST_03, S_DIST_04, S_DIST_05, S             1550835               707148    0.45597887
Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)                           1554392               508568    0.32718130
Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)                           1555454               497197    0.31964751
Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)                           1570245               493692    0.31440444
Insert into ORDER_LINE values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)                           1550813               465049    0.29987432
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?                        157145               419283    2.66812816
Update DISTRICT set D_NEXT_O_ID = ? where D_W_ID = ? and D_ID = ?                        155719               397364    2.55180164

  10 record(s) selected.
```

(待機時間は、ミリ秒単位でレポートされます)

第 14 章 DB2 pureScale 環境における非推奨のモニター・フィーチャーの使用

IBM DB2 pureScale Feature では、 DB2 モニター・インフラストラクチャーが拡張されており、DB2 pureScale インスタンスに関する情報を取得するのに使用できる多様な一連のモニター・エレメントが備わっています。ただし、非推奨のモニター・インターフェースを使用する場合には、モニター・データの取得と解釈を行う際に注意すべき制約事項が幾つか存在します。

このトピックでは、以下の非推奨フィーチャーを使用する場合に注意すべき制限事項について取り上げます。

- 『スナップショット・モニターによるロックのモニター』
- 1802 ページの『デッドロック・イベント・モニター』
- 1805 ページの『LIST TABLESPACES コマンドと LIST TABLESPACE CONTAINERS コマンド』

スナップショット・モニターによるロックのモニター

スナップショット・モニター・コマンド、関数、ビューを使用してメンバー間のロックに関する情報を調べる場合、スナップショットが取られているのと同じメンバーでアプリケーションが実行されていると、表示されるのはロックを保持しているアプリケーションの詳細情報だけになります。つまり、ロックを保持しているアプリケーションの ID は REMOTE APPLICATION としてレポートされ、アプリケーション ID やロック・モードなどのその他の情報は省略されます。そのため、すべてのメンバーの情報が戻るように、グローバル・スナップショットを取ることを考慮してください。

例えば、図 23 は **GET SNAPSHOT FOR APPLICATION** コマンドの出力を示しています。この例の場合、ロックを保持しているアプリケーションはコマンドを実行しているのと同じメンバー上にあります。

```
ID of agent holding lock                = 73
Application ID holding lock             = *N0.user1.080616184956
Database partition lock wait occurred on = 0
Lock name                               = 0x02000400000000000000000054
Lock attributes                         = 0x00000000
Release flags                           = 0x00000000
Lock object type                        = Table
Lock mode                               = Exclusive Lock (X)
Lock mode requested                     = Share Lock (S)
Name of tablespace holding lock         = USERSPACE1
Schema of table holding lock            = USER1
Name of table holding lock              = T1
Data Partition Id of table holding lock  = 0
Lock wait start timestamp               = 06/16/2009 14:50:26.744694
```

図 23. GET SNAPSHOT FOR APPLICATION コマンドの出力 - ロックを保持しているメンバーで実行されるコマンド： この例では、ロックを保持しているアプリケーションは、**GET SNAPSHOT** コマンドを実行しているのと同じメンバー上で実行されています。

ただし、ロックがリモート・メンバー上のアプリケーションによって保持されている場合には、図 24 に示されているようなレポートになります。

```
Application ID holding lock             = REMOTE APPLICATION
Database partition lock wait occurred on = 0
Lock name                               = 0x020004000000000000000000054
Lock attributes                         = 0x00000000
Release flags                           = 0x00000000
Lock object type                        = Table
Lock mode requested                     = Share Lock (S)
Name of tablespace holding lock         = USERSPACE1
Schema of table holding lock            = USER1
Name of table holding lock              = T1
Data Partition Id of table holding lock  = 0
Lock wait start timestamp               = 06/16/2009 14:50:26.744694
```

図 24. GET SNAPSHOT FOR APPLICATION コマンドの出力 - ロックを保持しているメンバー以外のメンバーで実行されるコマンド： この例では、ロックを保持しているアプリケーションは、**GET SNAPSHOT** コマンドを実行しているのとは別のメンバーで実行されています。**ID of agent holding lock** と **Lock mode** という行が省略されています。また、**Application ID holding lock** には REMOTE APPLICATION と表示されています。

グローバル・スナップショットを取ると、すべてのメンバーのデータが戻されます。スナップショット出力でロック名を調べると、どこでロックを保持しているかを判別できます。各メンバーのレポートを流し読みすると、該当のロックを保持しているアプリケーションを素早く見つけることができます。

デッドロック・イベント・モニター

デッドロックが生じると、デッドロックを追跡するイベント・モニターによって使用される情報を、デッドロック検出機能が生成します。非推奨の **CREATE EVENT MONITOR ... FOR DEADLOCKS** コマンドでは、DB2 pureScale 環境におけるメンバー間のデッドロックに関する一部の詳細情報が表示されない場合があります。図 25および図 26で示しているように、デッドロック・イベント・レポート作成の場合、デッドロックの影響を受けたアプリケーションに関する詳細情報が、**dbevmon** ツールの出力には表示されないことがあります。

```
3) Deadlock Event ...
  Deadlock ID:   1
  Deadlock node: 0
  Number of applications deadlocked: 2
  Deadlock detection time: 06/17/2009 14:46:22.543136
  Rolled back Appl participant no: 2
```

図 25. **db2evmon** *出力例 (DB2 pureScale インスタンス)*

```
3) Deadlock Event ...
  Deadlock ID:   1
  Deadlock node: 0
  Number of applications deadlocked: 2
  Deadlock detection time: 06/17/2009 14:46:22.543136
  Rolled back Appl participant no: 2
  Rolled back Appl Id: *N0.finance.081217170042
  Rolled back Appl seq number: : 0001
  Rolled back Appl handle: 66
```

図 26. **db2evmon** *出力例 (その他のすべてのタイプの DB2 インスタンス)*

(アプリケーション詳細が表示されるかどうかは、ユーザーが制御不能な幾つかの要因によって決まります。) ただし、デッドロックに関係するアプリケーションを判別することはできます。そのためには、デッドロック ID とロールバック・アプリケーション参加者番号に関してレポートされたデータを、**dbevmon** 出力の Deadlocked Connection セクションの情報と関連付けて導き出します。

同様に、出力の Deadlocked Connection セクションには、要求されたロックを保持しているアプリケーションのアプリケーション ID、シーケンス番号、またはロック・モードは含まれていません。ただし、ロックを保持しているアプリケーションのデッドロック ID、メンバー ID、ロック名、ロック・タイム・スタンプ、参加者番号は表示されます (メンバー ID は、**dbevmon** ツールの出力の「Deadlock node」として表示されます)。この情報を使用して、競合の原因となっているアプリケーションを導き出すことができます。以下の、**db2evmon** ツールからの出力例は、この動作について示しています。デッドロックに関連するアプリケーションを導き出すために使用する情報には下線が付されています。

```
5) Deadlock Event ...
 Deadlock ID:   1
 Deadlock node: 0
 Number of applications deadlocked: 2
 Deadlock detection time: 12/17/2008 12:01:12.735436
 Rolled back Appl participant no: 2
 Rolled back Appl Id: *N0.finance.081217170042
 Rolled back Appl seq number: : 0001
 Rolled back Appl handle: 66

6) Connection Header Event ...
 Appl Handle: 66
 Appl Id: *N0.finance.081217170042
 Appl Seq number: 00001
 DRDA AS Correlation Token: *N0.finance.081217170042
 Program Name    : db2bp
 Authorization Id: FINANCE
 Execution Id    : finance
 Codepage Id: 1208
 Territory code: 1
 Client Process Id: 7201
 Client Database Alias: A
 Client Product Id: SQL09070
 Client Platform: Unknown
 Client Communication Protocol: Local
 Client Network Name: so2.torolab.ibm.com
 Connect timestamp: 12/17/2008 12:00:42.176747

 7) Deadlocked Connection ...
 Deadlock ID:   1
 Deadlock Node: 0
 Participant no.: 2
 Participant no. holding the lock: 1
 Appl Id: *N0.finance.081217170042
 Appl Seq number: 00001
 Appl Id of connection holding the lock: REMOTE_APPLICATION
 Lock wait start time: 12/17/2008 12:01:01.607230
 Lock Name       : 0x02000500040000010000000052
 Lock Attributes : 0x00000000
 Release Flags   : 0x00000000
 Lock Count      : 0
 Hold Count      : 0
 Current Mode    : none
 Deadlock detection time: 12/17/2008 12:01:17.730069
 Table of lock waited on      : T2
 Schema of lock waited on     : FINANCE
 Data partition id for table  : 0
 Tablespace of lock waited on : USERSPACE1
 Type of lock: Row
 Mode application requested on lock: NS  - Share (CS/RS)
 Node lock occured on: 2
 Lock object name: 16777220
 Application Handle: 66
 Deadlocked Statement:
   Type     : Dynamic
```

```
  Operation: Fetch
  Section  : 201
  Creator  : NULLID
  Package  : SQLC2G17
  Cursor   : SQLCUR201
  Cursor was blocking: FALSE
  Text     : select * from t2

 List of Locks:
...

     Database partition          : 0
     Lock Name                   : 0x020004000100FFFFFF81000000000052
     Lock Attributes             : 0x00000008
     Release Flags               : 0x40000000
     Lock Count                  : 1
     Hold Count                  : 0
     Lock Object Name            : 8454145
     Object Type                 : Row
     Tablespace Name             : USERSPACE1
     Table Schema                : FINANCE
     Table Name                  : T1
     Data partition id           : 0
     Mode                        : X   - Exclusive

     Database partition          : 0
     Lock Name                   : 0x02000500000000000000000054
     Lock Attributes             : 0x00000000
     Release Flags               : 0x00000001
     Lock Count                  : 1
     Hold Count                  : 0
     Lock Object Name            : 5
     Object Type                 : Table
     Tablespace Name             : USERSPACE1
     Table Schema                : FINANCE
     Table Name                  : T2
     Data partition id           : 0
     Mode                        : IS  - Intent Share
...

 Locks Held:    6
 Locks in List: 6
 Locks Displayed: 6

8) Connection Header Event ...
 Appl Handle: 131137
 Appl Id: *N2.finance.081217170053
 Appl Seq number: 00001
 DRDA AS Correlation Token: *N2.finance.081217170053
 Program Name    : db2bp
 Authorization Id: finance
 Execution Id    : finance
 Codepage Id: 1208
 Territory code: 1
 Client Process Id: 7260
 Client Database Alias: A
 Client Product Id: SQL09070
 Client Platform: Unknown
 Client Communication Protocol: Local
 Client Network Name: so2.torolab.ibm.com
 Connect timestamp: 12/17/2008 12:00:43.542242

9) Deadlocked Connection ...
 Deadlock ID:   1
 Deadlock Node: 0
 Participant no.: 1
 Participant no. holding the lock: 2
 Appl Id: *N2.finance.081217170053
 Appl Seq number: 00001
 Appl Id of connection holding the lock: REMOTE_APPLICATION
 Lock wait start time: 12/17/2008 12:00:57.844388
 Lock Name       : 0x020004000100FFFFFF81000000000052
 Lock Attributes : 0x00000000
 Release Flags   : 0x00000000
 Lock Count      : 0
 Hold Count      : 0
 Current Mode    : none
 Deadlock detection time: 12/17/2008 12:01:17.744611
 Table of lock waited on      : T1
 Schema of lock waited on     : FINANCE
```

```
 Data partition id for table  : 0
 Tablespace of lock waited on : USERSPACE1
 Type of lock: Row
 Mode application requested on lock: NS  - Share (CS/RS)
 Node lock occured on: 0
 Lock object name: 8454145
 Application Handle: 131137
 Deadlocked Statement:
   Type     : Dynamic
   Operation: Fetch
   Section  : 201
   Creator  : NULLID
   Package  : SQLC2G17
   Cursor   : SQLCUR201
   Cursor was blocking: FALSE
   Text     : select * from t1
 List of Locks:
...
     Database partition          : 2
     Lock Name                   : 0x020005000400000100000000052
     Lock Attributes             : 0x00000008
     Release Flags               : 0x40000000
     Lock Count                  : 1
     Hold Count                  : 0
     Lock Object Name            : 16777220
     Object Type                 : Row
     Tablespace Name             : USERSPACE1
     Table Schema                : FINANCE
     Table Name                  : T2
     Data partition id           : 0
     Mode                        : X   - Exclusive

     Database partition          : 2
     Lock Name                   : 0x020005000000000000000000054
     Lock Attributes             : 0x00000000
     Release Flags               : 0x40000000
     Lock Count                  : 1
     Hold Count                  : 0
     Lock Object Name            : 5
     Object Type                 : Table
     Tablespace Name             : USERSPACE1
     Table Schema                : FINANCE
     Table Name                  : T2
     Data partition id           : 0
     Mode                        : IX  - Intent Exclusive

     Database partition          : 2
     Lock Name                   : 0x020004000000000000000000054
     Lock Attributes             : 0x00000000
     Release Flags               : 0x00000001
     Lock Count                  : 1
     Hold Count                  : 0
     Lock Object Name            : 4
     Object Type                 : Table
     Tablespace Name             : USERSPACE1
     Table Schema                : FINANCE
     Table Name                  : T1
     Data partition id           : 0
     Mode                        : IS  - Intent Share

 Locks Held:    6
 Locks in List: 6
 Locks Displayed: 6
```

LIST TABLESPACES コマンドと LIST TABLESPACE CONTAINERS コマンド

LIST TABLESPACES コマンドと **LIST TABLESPACE CONTAINERS** コマンドは、どちらも DB2 バージョン 9.7 で非推奨になっています。これらのコマンドは、実行されているメンバー上の情報のみをレポートします。これらのコマンドは、インスタンス内の他のメンバーの情報を取得しません。そのため、表スペース内で使用されているページ数など、一部のデータに関しては正確にレポートできませんでした。代わりに、MON_GET_TABLESPACE 表関数と MON_GET_CONTAINER 表関数を使用してください。

第 15 章 新規モニター・エレメントおよび変更されたモニター・エレメント

cf_wait_time - クラスター・キャッシング・ファシリティー待機時間 : モニター・エレメント

cf_wait_time モニター・エレメントは、クラスター・キャッシング・ファシリティーと通信するために費やした時間を格納します。

この時間には、ロックの認可やページ再利用の実行など、要求された処理または通信の結果として生じる場合がある処理を実行するための時間は含まれません。測定単位はミリ秒です。

表 2170. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT_TIMES_BY _ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

使用法

この値は、DB2 がクラスター・キャッシング・ファシリティーとの通信の際に待機に費やした時間の指標です。

cf_waits - クラスター・キャッシング・ファシリティー待機回数 : モニター・エレメント

DB2 データベース・システムがクラスター・キャッシング・ファシリティーと通信した際に待機した回数。

表 2171. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2172. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

configured_cf_gbp_size - 構成済みクラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント

cf_gbp_sz 構成パラメーターを使用して指定された、割り振り済みの予約されたグループ・バッファー・プール・メモリー (ページ・サイズ 4 KB のページ単位)。

表 2173. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

configured_cf_lock_size - 構成済みクラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント

構成されているグローバル・ロック・メモリー (ページ・サイズ 4 KB のページ単位)。この値は、**cf_lock_sz** 構成パラメーターで指定します。

表 2174. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

configured_cf_mem_size - 構成済みクラスター・キャッシング・ファシリティーのメモリー・サイズ : モニター・エレメント

クラスター・キャッシング・ファシリティーの構成済み合計メモリー・サイズ (ページ・サイズ 4 KB のページ単位)。この値は、**cf_mem_sz** 構成パラメーターで指定します。

表 2175. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

configured_cf_sca_size - 構成済みクラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント

現在割り振られていて予約済みの共用通信域メモリー (ページ・サイズ 4 KB のページ単位)。この値は、**cf_sca_sz** 構成パラメーターで指定します。

表 2176. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

current_cf_gbp_size - 現行クラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント

クラスター・キャッシング・ファシリティーで現在使用されているグループ・バッファー・プール・メモリー (ページ・サイズ 4 KB のページ単位)。

表 2177. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

current_cf_lock_size - 現行クラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント

現在使用されているグローバル・ロック・メモリー (ページ・サイズ 4 KB のページ単位)。

表 2178. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

current_cf_mem_size - 現行クラスター・キャッシング・ファシリティーのメモリー・サイズ :モニター・エレメント

現在使用されている合計メモリー (ページ・サイズ 4 KB のページ単位)。

表 2179. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

current_cf_sca_size - 現行クラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント

現在使用されている共用通信域メモリー (ページ・サイズ 4 KB のページ単位)。

表 2180. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

db_name データベース名 : モニター・エレメント

情報が収集されるデータベースの実名、またはアプリケーションの接続先のデータベースの実名。これはデータベースの作成時に与えられた名前です。

表 2181. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_AUTO_MAINT_QUEUE 表関数 - 自動保守ジョブの情報の取得	ACTIVITY METRICS BASE
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	ACTIVITY METRICS BASE

表 2182. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl_id_info	基本
アプリケーション	appl_remote	基本
表スペース	tablespace_list	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
表	table_list	表
ロック	db_lock_list	基本
動的 SQL	dynsql_list	基本
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl_info	基本

表 2183. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbheader	常に収集される

使用法

このエレメントを使用すると、データが適用される特定のデータベースを識別できます。

ホストへの接続、または System i のデータベース・サーバーへの接続で DB2 Connect を使用しないアプリケーションの場合は、このエレメントと **db_path** モニ

ター・エレメントを組み合わせて使用すると、データベースを個別に識別し、モニターが提供する情報の各レベルに関連付けることができます。

dbpartitionnum - データベース・パーティション番号 : モニター・エレメント

パーティション・データベース環境の場合、これは、データベース・メンバーの数値 ID です。DB2 Enterprise Server Edition および DB2 pureScale 環境の場合には、この値は 0 です。

表 2184. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_MSGS 表関数 - ADMIN_CMD プロシージャーを通して実行するデータ移動ユーティリティーによって生成されたメッセージの検索	ACTIVITY METRICS BASE
ADMIN_GET_STORAGE_PATHS 表関数 - ストレージ・グループのストレージ・パス情報の取得	ACTIVITY METRICS BASE
ADMIN_GET_TAB_COMPRESS_INFO 表関数 - 圧縮節約量の見積もり	ACTIVITY METRICS BASE
ADMIN_GET_TAB_DICTIONARY_INFO 表関数 - 既存の表ディクショナリーのプロパティーの報告	ACTIVITY METRICS BASE
ADMINTABINFO 管理ビューおよび ADMIN_GET_TAB_INFO 表関数 - 表のサイズおよび状態に関する情報の検索	ACTIVITY METRICS BASE
AUDIT_ARCHIVE プロシージャーおよび表関数 - 監査ログ・ファイルのアーカイブ	ACTIVITY METRICS BASE
DBCFG 管理ビューおよび DB_GET_CFG 表関数 - データベース構成パラメーター情報の取得	ACTIVITY METRICS BASE
DBPATHS 管理ビューおよび ADMIN_LIST_DB_PATHS 表関数 - データベース・パスの取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 2184. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE
PDLOGMSGS_LAST24HOURS 管理ビューおよび PD_GET_LOG_MSGS 表関数 - 問題診断メッセージの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

使用法

DB2 pureScale 環境では、複数のメンバーが単一のパーティションで作動します。そのような構成で実行される場合、表スペース内のフリー・ページの番号などのストレージに関する物理属性は、システム内のすべてのメンバーで重複します。それぞれのメンバーが、システムの正確なサイズ合計をレポートします。複数パーティション構成の場合、システム全体の値を把握するために各パーティションからの値がユーザーによって相互に関連付けられる必要があります。

dbpartitionnum モニター・エレメントは **data_partition_id** モニター・エレメントとは異なります。後者のモニター・エレメントは、値に基づいて、表にあるデータを細分化して作成されたデータ・パーティションを識別するのに使用されます。

host_name - ホスト名 : モニター・エレメント

クラスター・キャッシング・ファシリティープロセスがあるホストの名前。

表 2185. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
DB_MEMBERS 表関数	ACTIVITY METRICS BASE

表 2185. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE

id - クラスター・キャッシング・ファシリティー ID モニター・エレメント

db2nodes.cfg ファイルで定義されているクラスター・キャッシング・ファシリティー ID。

表 2186. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
DB2_MEMBER および DB2_CF 管理ビューおよび DB2_GET_INSTANCE_INFO 表関数	ACTIVITY METRICS BASE
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CF_CMD 表関数 - クラスター・キャッシング・ファシリティーのコマンド処理時間の取得	ACTIVITY METRICS BASE
MON_GET_CF_WAIT_TIME 表関数 - クラスター・キャッシング・ファシリティーのコマンド待機時間の取得	ACTIVITY METRICS BASE

lock_escals - ロック・エスカレーション数 : モニター・エレメント

ロックが複数の行ロックから 1 つの表ロックにエスカレートされた回数。

表 2187. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 2187. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2188. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 2189. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
データベース	event_db	常に収集される
接続	event_conn	常に収集される
トランザクション	event_xact	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

アプリケーションが保留するロックの合計数がそのアプリケーションで使用可能なロック・リスト・スペースの最大量に達した場合、またはすべてのアプリケーションが使用するロック・リスト・スペースが合計ロック・リスト・スペースに近くなると、ロックはエスカレートされます。使用可能なロック・リスト・スペースの量は、 **maxlocks** および **locklist** 構成パラメーターによって決まります。

1 つのアプリケーションが使用可能な最大ロック数に達して、エスカレートするロックがほかにない場合は、ほかのアプリケーションに割り振られているロック・リストのスペースが使用されます。ロック・リスト全体が満杯になるとエラーが起こります。

このデータ項目には、排他ロック・エスカレーションおよび DB2 pureScale 環境におけるロック・エスカレーションも含めて、すべてのロック・エスカレーションのカウントが含まれます。DB2 pureScale 環境におけるロック・エスカレーションだけを判別するには、**lock_escals_global** モニター・エレメントを使用してください。

過剰なロック・エスカレーションが起こる場合は、いくつかの原因が考えられます。

- 同時アプリケーションの数に対してロック・リスト・サイズ (**locklist**) が小さい場合。
- 各アプリケーションが使用できるロック・リストのパーセント値 (**maxlocks**) が小さい場合。
- 1 つ以上のアプリケーションが使用しているロックの数が多すぎる場合。
- DB2 pureScale 環境において、グローバル・ロック・リスト・サイズ (**cf_lock_sz**) が小さい場合。

これらの問題を解決するには、次のようにしてください。

- **locklist** 構成パラメーター値を大きくする。
- **maxlocks** 構成パラメーター値を大きくする。
- 以下のいずれかの公式を使用し、その値を **maxlocks** と比較することによって、ロック数の多いアプリケーション、またはロック・リストの大半を保持しているアプリケーションを識別する。
  - 64 ビット・システムでは (((保持されているロック * 64) / (locklist * 4096)) * 100)
  - 32 ビット・システムでは (((保持されているロック * 48) / (locklist * 4096)) * 100)

  これらのアプリケーションがロック・リストの多くを使用すると、ほかのアプリケーションでロック・エスカレーションを起こします。これらのアプリケーションでは行ロックの代わりに表ロックを使用する必要が生じることがありますが、表ロックが原因で **lock_waits** および **lock_wait_time** モニター・エレメント値が増える可能性があります。

lock_escals_global - グローバル・ロック・エスカレーション数 : モニター・エレメント

グローバル・ロック・メモリー使用量が **cf_lock_sz** データベース構成パラメーターで指定した限度に達したために生じた、グローバル・ロックにおけるロック・エスカレーション数。

表 2190. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 2190. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2191. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このモニター・エレメントを、**lock_escals_maxlocks** モニター・エレメントと **lock_escals_locklist** モニター・エレメントと併せて使用して、データベースでエ

スカレーションの原因となっているロック・スペース構成パラメーターを判別します。

lock_escals_locklist - locklist ロック・エスカレーション数 : モニター・エレメント

ローカル・ロック・メモリー使用量が、**locklist** データベース構成パラメーターで指定した限度に達したために生じたロック・エスカレーション数。

表 2192. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2192. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2193. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このモニター・エレメントを、**lock_escals_maxlocks** モニター・エレメントと **lock_escals_global** モニター・エレメントと併せて使用して、データベースでエスカレーションの原因となっているロック・スペース構成パラメーターを判別します。

lock_escals_maxlocks - maxlocks ロック・エスカレーション数 : モニター・エレメント

ローカル・ロック・メモリー使用量が、**maxlocks** データベース構成パラメーターで指定した限度に達したために生じたロック・エスカレーション数。

表 2194. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 2194. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2195. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される

表 2195. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される

使用法

このモニター・エレメントを、**lock_escals_locklist** モニター・エレメントと **lock_escals_global** モニター・エレメントと併せて使用して、データベースでエスカレーションの原因となっているロック・スペース構成パラメーターを判別します。

lock_timeouts_global - グローバル・ロック・タイムアウト : モニター・エレメント

ロックを保持しているアプリケーションがリモート・メンバー上にある場合のロック・タイムアウト数。

表 2196. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2196. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2197. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このエレメントを、**lock_timeouts** モニター・エレメントと併せて使用してください。**lock_timeouts_global** モニター・エレメントは、別のメンバーで保持されているロックを獲得するために待機中に生じたロック・タイムアウト数を示します。同じメンバーで保持されているロックを獲得するための待機中に生じたロック・タイムアウト数を判別するには、以下の数式を使用してください。

```
lock_timeouts - lock_timeouts_global
```

DB2 pureScale 環境以外では、この値は常時ゼロになります。

lock_wait_time_global - グローバル・ロック待機時間 : モニター・エレメント

グローバルなロック待機時間。この時間の測定単位はミリ秒です。

表 2198. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 2198. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2199. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このモニター・エレメントを **lock_wait_time** モニター・エレメント (ロックに費やされた合計待機時間を示します) と併せて使用します。**lock_wait_time_global** モニター・エレメントは、別のメンバー上の競合アプリケーションによって保持されているロックの待機時間を示します。同じメンバー上の競合アプリケーションによって保持されているロックの合計待機時間を判別するには、以下の数式を使用してください。

```
lock_wait_time - lock_wait_time_global
```

DB2 pureScale 環境以外では、この値は常時ゼロになります。

lock_wait_time_global_top - 最長グローバル・ロック待機時間 : モニター・エレメント

別のメンバーで保持されているロックに関して生じた最長ロック待機。この値はミリ秒単位で報告されます。

表 2200. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	常に収集される

lock_waits_global - グローバル・ロック待機 : モニター・エレメント

リモート・メンバー上にあるロックをアプリケーションが保持しているために生じたロック待機数。

表 2201. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	DATA OBJECT METRICS EXTENDED

表 2201. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	DATA OBJECT METRICS EXTENDED
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2202. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

このモニター・エレメントを、**lock_waits** モニター・エレメント (すべてのメンバー上の競合アプリケーションによって保持されているロックが原因で生じたロック待機合計数をレポートします) と一緒に使用してください。**lock_waits_global** モニター・エレメントは、別のメンバー上の競合アプリケーションで保持されていたロック待機回数を示します。待機アプリケーションと同じメンバー上の競合アプリケーションによって保持されているロック待機数を判別するには、以下の数式を使用します。

```
lock_waits - lock_waits_global
```

DB2 pureScale 環境以外では、この値は常時ゼロになります。

member - データベース・メンバー・モニター・エレメント

この結果レコードのデータの取得元となったデータベース・メンバーの数値 ID。

表 2203. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
ADMIN_GET_MEM_USAGE 表関数 - インスタンスの合計メモリー消費量の取得	ACTIVITY METRICS BASE
AUDIT_ARCHIVE プロシージャーおよび表関数 - 監査ログ・ファイルのアーカイブ	ACTIVITY METRICS BASE
DBCFG 管理ビューおよび DB_GET_CFG 表関数 - データベース構成パラメーター情報の取得	ACTIVITY METRICS BASE
ENV_GET_REG_VARIABLES 表関数 - 使用中の DB2 レジストリー設定の取得	ACTIVITY METRICS BASE
ENV_GET_DB2_SYSTEM_RESOURCES 表関数 - DB2(r) システム情報を返す	ACTIVITY METRICS BASE
ENV_GET_SYSTEM_RESOURCES 表関数 - システム情報を戻す	ACTIVITY METRICS BASE
ENV_GET_NETWORK_RESOURCES 表関数 - ネットワーク・アダプター情報を返す	ACTIVITY METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_AUTO_MAINT_QUEUE 表関数 - 自動保守ジョブの情報の取得	ACTIVITY METRICS BASE
MON_GET_AUTO_RUNSTATS_QUEUE 表関数 - 評価用にキューに入れられたオブジェクトの情報の取得	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CF_WAIT_TIME 表関数 - クラスター・キャッシング・ファシリティーのコマンド待機時間の取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	ACTIVITY METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_EXTENDED_LATCH_WAIT 表関数 - ラッチ情報を戻す	ACTIVITY METRICS BASE
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	ACTIVITY METRICS BASE

表 2203. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_FCM - FCM メトリックの取得	ACTIVITY METRICS BASE
MON_GET_FCM_CONNECTION_LIST - すべての FCM 接続に関する詳細の取得	ACTIVITY METRICS BASE
MON_GET_GROUP_BUFFERPOOL 表関数 - グループ・バッファー・プール・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE
MON_GET_INDEX_USAGE_LIST 表関数 - 索引使用リストの情報を返す	ACTIVITY METRICS BASE
MON_GET_LOCKS 表関数 - 現在接続されているデータベース内のすべてのロックのリスト	ACTIVITY METRICS BASE
MON_GET_MEMORY_POOL 表関数 - メモリー・プール情報の取得	ACTIVITY METRICS BASE
MON_GET_MEMORY_SET 表関数 - メモリー・セット情報の取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_REBALANCE_STATUS 表関数 - 表スペースのリバランスの進行の取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_RTS_RQST 表関数 - リアルタイム統計要求の情報の取得	ACTIVITY METRICS BASE
MON_GET_SERVERLIST 表関数 - メンバーの優先順位の詳細を取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	ACTIVITY METRICS BASE

表 2203. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TRANSACTION_LOG 表関数 - ログ情報の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得	ACTIVITY METRICS BASE
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE
PDLOGMSGS_LAST24HOURS 管理ビューおよび PD_GET_LOG_MSGS 表関数 - 問題診断メッセージの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE

表 2203. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 2204. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	-	COLLECT BASE DATA
ロッキング	-	常に収集される
変更履歴	changesummary dbdbmcfg regvar ddlstmtexec txncompletion evmonstart utilstart utillocation utilstop	常に収集される

使用法

DB2 メンバーは、単一ホスト上で DB2 サーバー・ソフトウェアを実行するデータベース・マネージャー・インスタンスであり、DB2 メンバーはそれに接続するアプリケーションからのデータベース要求を受け入れて処理します。

objtype - オブジェクト・タイプ : モニター・エレメント

レポート対象のモニター・データのオブジェクト・タイプ。このモニター・エレメントは、object_type モニター・エレメントの別名です。

表 2205. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_TAB_COMPRESS_INFO 表関数 - 圧縮節約量の見積もり	ACTIVITY METRICS BASE
ADMIN_GET_TAB_DICTIONARY_INFO 表関数 - 既存の表ディクショナリーのプロパティーの報告	ACTIVITY METRICS BASE
MON_GET_AUTO_MAINT_QUEUE 表関数 - 自動保守ジョブの情報の取得	ACTIVITY METRICS BASE
MON_GET_AUTO_RUNSTATS_QUEUE 表関数 - 評価用にキューに入れられたオブジェクトの情報の取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_RTS_RQST 表関数 - リアルタイム統計要求の情報の取得	ACTIVITY METRICS BASE
MON_GET_USAGE_LIST_STATUS 表関数 - 使用リストの状況を返す	ACTIVITY METRICS BASE
PD_GET_DIAG_HIST 表関数 - 指定された機能からレコードを戻す	ACTIVITY METRICS BASE

表 2206. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
変更履歴	utilphase utilstart	常に収集される

使用上の注意

- このエレメントが ADMIN_GET_TAB_COMPRESS_INFO 表関数または ADMIN_GET_TAB_DICTIONARY_INFO 表関数から戻された場合、 object_type モニター・エレメントの戻り値は、「XML」または「DATA」のいずれかです。
- このエレメントが MON_GET_RTS_RQST 表関数から戻された場合、 object_type モニター・エレメントの戻り値は、「TABLE」です。
- このエレメントが MON_GET_AUTO_MAINT_QUEUE 表関数から戻された場合、object_type モニター・エレメントの戻り値は、「DATABASE」、「TABLE」、「NICKNAME」、または「VIEW」です。
- このエレメントが MON_GET_AUTO_RUNSTATS_QUEUE 表関数から戻された場合、 object_type モニター・エレメントの戻り値は、「TABLE」、「NICKNAME」、または「VIEW」です。

- このエレメントが変更履歴イベント・モニターから戻された場合、object_type モニター・エレメントの戻り値は、「DATABASE」、「INDEX」、「PARTITIONGROUP」、「TABLE」、または「TABLESPACE」です。

page_reclaims_initiated_s - 共有アクセスで開始されたページ再利用 : モニター・エレメント

ページが別のメンバーから再利用される原因となった、共有モードでのページ・アクセス回数。内部的に保持されているオブジェクト・スペース・マップに関連するページ再利用は、別個にカウントされます。

表 2207. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

page_reclaims_initiated_x - 排他的アクセスで開始されたページ再利用 : モニター・エレメント

ページが別のメンバーから再利用される原因となった、排他モードでのそのページのアクセス回数。内部的に保持されているオブジェクト・スペース・マップに関連するページ再利用は、別個にカウントされます。

表 2208. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

page_reclaims_s - ページ再利用の共有アクセス : モニター・エレメント

オブジェクトに関連したページが、DB2 pureScale インスタンス内の別のメンバーによって計画的な解放より前に再利用された回数。その場合、そのページを再利用したメンバーは共有アクセスを必要としました。

表 2209. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

内部的に保持されているオブジェクト・スペース・マップに関連するページ再利用は、別個にカウントされます。

page_reclaims_x - ページ再利用の排他的アクセス : モニター・エレメント

オブジェクトに関連したページが、DB2 pureScale インスタンス内の別のメンバーによって計画的な解放より前に再利用された回数。その場合、そのページを再利用したメンバーは排他的アクセスを必要としました。

表 2210. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

内部的に保持されているオブジェクト・スペース・マップに関連するページ再利用は、別個にカウントされます。

pool_async_data_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント

ローカル・バッファー・プール内のページが無効であったために、プリフェッチャーが、グループ・バッファー・プールからデータ・ページを読み取ろうとした回数。

表 2211. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) のデータ・ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads )
 / pool_async_data_gbp_l_reads
```

IBM DB2 pureScale Feature の全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。この数式を使用すると、データベースのスループットでグループ・バッファー・プールがパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_async_data_gbp_l_reads - 非同期のグループ・バッファー・プール・データの論理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属のデータ・ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、プリフェッチャーがグループ・バッファー・プールから読み取ろうとした回数。

表 2212. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) のデータ・ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads )
 / pool_async_data_gbp_l_reads
```

IBM DB2 pureScale Feature の全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。この数式を使用すると、データベースのスループットでグループ・バッファー・プールがパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_async_data_gbp_p_reads - 非同期のグループ・バッファー・プール・データの物理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属のデータ・ページが、GBP 内で検出されなかったために、プリフェッチャーによってディスクからローカル・バッファー・プールに読み込まれた回数。

表 2213. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) のデータ・ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads )
 / pool_async_data_gbp_l_reads
```

IBM DB2 pureScale Feature の全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。この数式を使用すると、データベー

スのスループットでグループ・バッファー・プールがパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_async_data_lbp_pages_found - 非同期のローカル・バッファー・プールの検出データ・ページ : モニター・エレメント

プリフェッチャーがデータ・ページにアクセスしようとして、そのデータ・ページがローカル・バッファー・プールに存在した回数。

表 2214. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) のデータ・ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_data_gbp_l_reads - pool_async_data_gbp_p_reads )
 / pool_async_data_gbp_l_reads
```

IBM DB2 pureScale Feature の全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。この数式を使用すると、データベースのスループットでグループ・バッファー・プールがパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_async_index_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効な索引データ・ページ : モニター・エレメント

ローカル・バッファー・プール内のページが無効であったために、プリフェッチャーが、グループ・バッファー・プールから索引ページを読み取ろうとした回数。

表 2215. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の索引ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads )
 / pool_async_index_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_index_gbp_l_reads - 非同期のグループ・バッファー・プール索引の論理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属の索引ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、プリフェッチャーがグループ・バッファー・プールから読み取ろうとした回数。

表 2216. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の索引ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads )
 / pool_async_index_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_index_gbp_p_reads - 非同期のグループ・バッファー・プール索引の物理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属の索引ページが、GBP 内で検出されなかったために、プリフェッチャーによってディスクからローカル・バッファー・プールに読み込まれた回数。

表 2217. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の索引ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads )
 / pool_async_index_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_index_lbp_pages_found - 非同期のローカル・バッファー・プールの検出索引ページ : モニター・エレメント

プリフェッチャーが索引ページにアクセスしようとして、その索引ページがローカル・バッファー・プールに存在した回数。

表 2218. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の索引ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_index_gbp_l_reads - pool_async_index_gbp_p_reads )
 / pool_async_index_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_xda_gbp_invalid_pages - 非同期のグループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・ページが、ローカル・バッファー・プールで無効のマークが付けられているため、プリフェッチャーによってグループ・バッファー・プールから要求された回数。

表 2219. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 2219. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の XDA ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads )
 / pool_async_xda_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_xda_gbp_l_reads - グループ・バッファー・プール XDA データの非同期論理読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) の GBP 従属のデータ・ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、プリフェッチャーがグループ・バッファー・プールから読み取ろうとした回数。

表 2220. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の XDA ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads )
 / pool_async_xda_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_xda_gbp_p_reads - グループ・バッファー・プール XDA データの非同期物理読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) の GBP 従属のデータ・ページが、GBP で検出されなかったために、プリフェッチャーがディスクからローカル・バッファー・プールに読み込んだ回数。

表 2221. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の XDA ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads )
 / pool_async_xda_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グループ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_async_xda_lbp_pages_found - 非同期のローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・ページが、プリフェッチャーによって要求されてローカル・バッファー・プールで検出された回数。

表 2222. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

プリフェッチャー (または非同期) の XDA ページ・ヒット率は、以下のように計算できます。

```
GBP = ( pool_async_xda_gbp_l_reads - pool_async_xda_gbp_p_reads )
 / pool_async_xda_gbp_l_reads
```

DB2 pureScale インスタンスの全体的なパフォーマンスにおいて、バッファー・プールのヒット・レートは重要な要因となります。前述の数式を使用すると、グルー

プ・バッファー・プールがデータベースのスループットを制限する要因となっているかどうかを判別するのに役立ちます。

pool_data_gbp_invalid_pages - グループ・バッファー・プールの無効なデータ・ページ : モニター・エレメント

データ・ページがローカル・バッファー・プールで無効であったため、代わりにグループ・バッファー・プールから読み取られた回数。 DB2 pureScale 環境以外では、この値は NULL になります。

表 2223. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 2223. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2224. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求されたデータ・ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_DATA_LBP_PAGES_FOUND - POOL_ASYNC_DATA_LBP_PAGES_FOUND) / POOL_DATA_L_READS
```

要求されたデータ・ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_DATA_GBP_L_READS - POOL_DATA_GBP_P_READS) / POOL_DATA_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_data_gbp_l_reads - グループ・バッファー・プール・データの論理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属のデータ・ページが、ローカル・バッファー・プール (LBP) で無効であったか、存在しなかったため、グループ・バッファー・プールから読み取ろうとされた回数。

表 2225. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2225. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2226. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求されたデータ・ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_DATA_LBP_PAGES_FOUND - POOL_ASYNC_DATA_LBP_PAGES_FOUND) / POOL_DATA_L_READS
```

要求されたデータ・ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_DATA_GBP_L_READS - POOL_DATA_GBP_P_READS) / POOL_DATA_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_data_gbp_p_reads - グループ・バッファー・プール・データの物理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属のデータ・ページが、GBP 内で検出されなかったためにディスクからローカル・バッファー・プールに読み込まれた回数。

表 2227. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2227. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2228. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求されたデータ・ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_DATA_LBP_PAGES_FOUND - POOL_ASYNC_DATA_LBP_PAGES_FOUND) / POOL_DATA_L_READS
```

要求されたデータ・ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_DATA_GBP_L_READS - POOL_DATA_GBP_P_READS) / POOL_DATA_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_data_lbp_pages_found - ローカル・バッファー・プールの検出データ・ページ : モニター・エレメント

データ・ページがローカル・バッファー・プールで検出された回数。

表 2229. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2229. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2230. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求されたデータ・ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_DATA_LBP_PAGES_FOUND - POOL_ASYNC_DATA_LBP_PAGES_FOUND) / POOL_DATA_L_READS
```

要求されたデータ・ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_DATA_GBP_L_READS - POOL_DATA_GBP_P_READS) / POOL_DATA_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_index_gbp_invalid_pages - グループ・バッファー・プールの無効な索引ページ : モニター・エレメント

ローカル・バッファー・プール内のページが無効であったために、グループ・バッファー・プールから索引ページを読み取ろうとした回数。

表 2231. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2231. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2232. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求された索引ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_INDEX_LBP_PAGES_FOUND - POOL_ASYNC_INDEX_LBP_PAGES_FOUND) / POOL_INDEX_L_READS
```

要求された索引ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_INDEX_GBP_L_READS - POOL_INDEX_GBP_P_READS) / POOL_INDEX_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_index_gbp_l_reads - グループ・バッファー・プール索引の論理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属の索引ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、グループ・バッファー・プールから読み取ろうとされた回数。

表 2233. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2233. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2234. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求された索引ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_INDEX_LBP_PAGES_FOUND - POOL_ASYNC_INDEX_LBP_PAGES_FOUND) / POOL_INDEX_L_READS
```

要求された索引ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_INDEX_GBP_L_READS - POOL_INDEX_GBP_P_READS) / POOL_INDEX_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_index_gbp_p_reads - グループ・バッファー・プール索引の物理読み取り : モニター・エレメント

グループ・バッファー・プール (GBP) 従属の索引ページが、GBP 内で検出されなかったためにディスクからローカル・バッファー・プールに読み込まれた回数。

表 2235. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2235. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2236. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求された索引ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_INDEX_LBP_PAGES_FOUND - POOL_ASYNC_INDEX_LBP_PAGES_FOUND) / POOL_INDEX_L_READS
```

要求された索引ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_INDEX_GBP_L_READS - POOL_INDEX_GBP_P_READS) / POOL_INDEX_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_index_lbp_pages_found - ローカル・バッファー・プールの検出索引ページ : モニター・エレメント

索引ページがローカル・バッファー・プールに存在していた回数。

表 2237. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 2237. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2238. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
パッケージ・キャッシュ	-	常に収集される
ロッキング	-	常に収集される

使用法

要求された索引ページがローカル・バッファー・プールで検出された頻度を判別するには、以下の数式を使用します。

```
(POOL_INDEX_LBP_PAGES_FOUND - POOL_ASYNC_INDEX_LBP_PAGES_FOUND) / POOL_INDEX_L_READS
```

要求された索引ページがグループ・バッファー・プールで検出された回数を判別するには、以下の数式を使用します。

```
(POOL_INDEX_GBP_L_READS - POOL_INDEX_GBP_P_READS) / POOL_INDEX_GBP_L_READS
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_xda_gbp_invalid_pages - グループ・バッファー・プールの無効な XDA データ・ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・ページが、ローカル・バッファー・プールで無効のマークが付けられているため、グループ・バッファー・プールから要求された回数。

表 2239. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 2239. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2240. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

要求された XDA ページがローカル・バッファー・プールで検出された頻度を判別するには、次の数式を使用します。

```
(pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found) / pool_xda_l_reads
```

要求された XDA ページがグループ・バッファー・プールで検出された回数を判別するには、次の数式を使用します。

```
(pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_xda_gbp_l_reads - グループ・バッファー・プール XDA データの論理読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) の GBP 従属のデータ・ページが、ローカル・バッファー・プールで無効であったか、存在しなかったため、グループ・バッファー・プールから読み取ろうとされた回数。

表 2241. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 2241. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2242. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

要求された XDA ページがローカル・バッファー・プールで検出された頻度を判別するには、次の数式を使用します。

```
(pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found) / pool_xda_l_reads
```

要求された XDA ページがグループ・バッファー・プールで検出された回数を判別するには、次の数式を使用します。

```
(pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_xda_gbp_p_reads - グループ・バッファー・プール XDA データの物理読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) の GBP 従属のデータ・ページが、グループ・バッファー・プールで検出されなかったためにディスクからローカル・バッファー・プールに読み込まれた回数。

表 2243. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 2243. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2244. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

要求された XDA ページがローカル・バッファー・プールで検出された頻度を判別するには、次の数式を使用します。

```
(pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found) / pool_xda_l_reads
```

要求された XDA ページがグループ・バッファー・プールで検出された回数を判別するには、次の数式を使用します。

```
(pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

pool_xda_lbp_pages_found - ローカル・バッファー・プールの検出 XDA データ・ページ : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のデータ・ページが、ローカル・バッファー・プールから要求されて検出された回数。

表 2245. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 2245. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2246. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

要求された XDA ページがローカル・バッファー・プールで検出された頻度を判別するには、次の数式を使用します。

```
(pool_xda_lbp_pages_found - pool_async_xda_lbp_pages_found) / pool_xda_l_reads
```

要求された XDA ページがグループ・バッファー・プールで検出された回数を判別するには、次の数式を使用します。

```
(pool_xda_gbp_l_reads - pool_xda_gbp_p_reads) / pool_xda_gbp_l_reads
```

クラスター・キャッシング・ファシリティーの全体的なパフォーマンスにおいて、ローカル・バッファー・プールとグループ・バッファー・プールのヒット・レートはどちらも重要な要因となります。これらの数式を使用すると、データベースのスループットでローカル・バッファー・プールやグループ・バッファー・プールかパフォーマンスを低下させる要因となっているかどうかを判別するのに役立ちます。

reclaim_wait_time - 再利用待機時間 : モニター・エレメント

DB2 pureScale 環境では、このエレメントは、ページ・ロックの待機にかかった時間を示します。ロック要求によって、ページが再利用されます。この時間の測定単位はミリ秒です。

表 2247. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 2247. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 2248. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される

使用法

スペース・マップ・ページ上で再利用を待機するための時間は別個にカウントされ、**spacemappage_reclaim_wait_time** モニター・エレメントでレポートされます。

spacemappage_page_reclaims_initiated_s - 共有アクセスで開始されたスペース・マップ・ページ再利用 : モニター・エレメント

ページが別のメンバーから再利用される原因となった、スペース・マップ・ページのための共有モードでのページ・アクセス回数。

表 2249. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

この値は、オブジェクト関連の表スペースに関してのみレポートされます。オブジェクト関連の表スペースとは、再利用可能なストレージで使用可能な表スペースで

す。**reclaimable_space_enabled** モニター・エレメントを使用して、再利用可能なストレージに対して表スペースが使用可能かどうかを判別してください。

エクステント・マップ・ページ (EMP) はメタデータであるため、このモニター・エレメントの値には EMP が含まれています。

データ・スペース・マップ・ページにはユーザー・データが含まれているので、**page_reclaims_initiated_s** モニター・エレメントの値に入っていると同時に、**spacemappage_page_reclaims_initiated_s** モニター・エレメントの値にも含まれます。索引スペース・マップ・ページにはユーザー・データが含まれていないので、**spacemappage_page_reclaims_initiated_s** モニター・エレメントの値にしか含まれません。

spacemappage_page_reclaims_initiated_x - 排他的アクセスで開始されたスペース・マップ・ページ再利用 : モニター・エレメント

ページが別のメンバーから再利用される原因となった、スペース・マップ・ページのための排他モードでのページ・アクセス回数。

表 2250. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

この値は、オブジェクト関連の表スペースに関してのみレポートされます。オブジェクト関連の表スペースとは、再利用可能なストレージで使用可能な表スペースです。**reclaimable_space_enabled** モニター・エレメントを使用して、再利用可能なストレージに対して表スペースが使用可能かどうかを判別してください。

エクステント・マップ・ページ (EMP) はメタデータであるため、このモニター・エレメントの値には EMP が含まれています。

データ・スペース・マップ・ページにはユーザー・データが含まれているので、**page_reclaims_initiated_x** モニター・エレメントの値に入っていると同時に、**spacemappage_page_reclaims_initiated_x** モニター・エレメントの値にも含まれます。索引スペース・マップ・ページにはユーザー・データが含まれていないので、**spacemappage_page_reclaims_initiated_x** モニター・エレメントの値にしか含まれません。

spacemappage_page_reclaims_s - スペース・マップ・ページ再利用の共有アクセス : モニター・エレメント

スペース・マップ・ページに関連したページが、DB2 pureScale内の別のメンバーによって計画的な解放より前に再利用された回数。そのページを再利用したメンバーは、スペース・マップ・ページに対する共有アクセスを必要としました。

表 2251. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

この値は、オブジェクト関連の表スペースに関してのみレポートされます。オブジェクト関連の表スペースとは、再利用可能なストレージで使用可能な表スペースです。**reclaimable_space_enabled** モニター・エレメントを使用して、再利用可能なストレージに対して表スペースが使用可能かどうかを判別してください。

エクステント・マップ・ページ (EMP) はメタデータであるため、このモニター・エレメントの値には EMP が含まれています。

データ・スペース・マップ・ページにはユーザー・データが含まれているので、**page_reclaims_s** モニター・エレメントの値に入っていると同時に、**spacemappage_page_reclaims_s** モニター・エレメントの値にも含まれます。索引スペース・マップ・ページにはユーザー・データが含まれていないので、**spacemappage_page_reclaims_s** モニター・エレメントの値にしか含まれません。

spacemappage_page_reclaims_x - スペース・マップ・ページ再利用の排他的アクセス : モニター・エレメント

スペース・マップ・ページに関連したページが、DB2 pureScale内の別のメンバーによって計画的な解放より前に再利用された回数。そのページを再利用したメンバーは、スペース・マップ・ページに対する排他的アクセスを必要としました。

表 2252. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE

使用法

この値は、オブジェクト関連の表スペースに関してのみレポートされます。オブジェクト関連の表スペースとは、再利用可能なストレージで使用可能な表スペースです。**reclaimable_space_enabled** モニター・エレメントを使用して、再利用可能なストレージに対して表スペースが使用可能かどうかを判別してください。

エクステント・マップ・ページ (EMP) はメタデータであるため、このモニター・エレメントの値には EMP が含まれています。

データ・スペース・マップ・ページにはユーザー・データが含まれているので、**page_reclaims_x** モニター・エレメントの値に入っていると同時に、**spacemappage_page_reclaims_x** モニター・エレメントの値にも含まれます。索引スペース・マップ・ページにはユーザー・データが含まれていないので、**spacemappage_page_reclaims_x** モニター・エレメントの値にしか含まれません。

spacemappage_reclaim_wait_time - スペース・マップ・ページ再利用の待機時間 : モニター・エレメント

DB2 pureScale 環境では、このエレメントは、内部的に保守されているオブジェクト・スペース管理に関連したページのページ・ロックの待機時間を示します。その際、ロック要求によって、ページは他のメンバーが再利用します。この時間の測定単位はミリ秒です。

表 2253. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 2253. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、 653 ページの『第 8 章 モニター・エレメントの収集レベル』を参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 2254. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される

table_name - 表名 : モニター・エレメント

表の名前。

表 2255. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE

表 2255. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
ADMIN_GET_TAB_COMPRESS_INFO 表関数 - 圧縮節約量の見積もり	ACTIVITY METRICS BASE
ADMIN_GET_TAB_DICTIONARY_INFO 表関数 - 既存の表ディクショナリーのプロパティーの報告	ACTIVITY METRICS BASE
ADMINTABINFO 管理ビューおよび ADMIN_GET_TAB_INFO 表関数 - 表のサイズおよび状態に関する情報の検索	ACTIVITY METRICS BASE
ADMINTEMPCOLUMNS 管理ビューおよび ADMIN_GET_TEMP_COLUMNS 表関数 - 一時表の列情報を取得する	ACTIVITY METRICS BASE
ADMINTEMPTABLES 管理ビューおよび ADMIN_GET_TEMP_TABLES 表関数 - 一時表の情報を取得する	ACTIVITY METRICS BASE
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

表 2256. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	ロック
ロック	lock_wait	ロック

表 2257. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
表	event_table	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されて

おらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントと **table_schema** を組み合わせて使用すると、リソースの競合の原因を判別できます。

アプリケーション・レベル、アプリケーション・ロック・レベル、およびデッドロック・モニター・レベルでは、現在別のアプリケーションにロックされているためにロック待ちになっているアプリケーションの表を示します。スナップショット・モニターの場合、この項目が有効なのは、「lock」モニター・グループ情報が ON に設定され、さらにアプリケーションが表ロックの取得待ちであることを **lock_object_type** が示している場合に限ります。

オブジェクト・レベルのスナップショット・モニターの場合は、表レベルおよび行レベルのロックについてこの項目が戻されます。このレベルで報告される表は、このアプリケーションがこれらのロックを保留している表です。

表レベルのスナップショット・モニターおよびイベント・モニターの場合は、情報が収集された表を示します。一時表の場合、 **table_name** の形式は 『TEMP (*n*, *m*)』 です。

- *n* は表スペース ID
- *m* は **table_file_id** エレメント

table_schema - 表スキーマ名 : モニター・エレメント

表のスキーマ。

表 2258. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
ADMIN_GET_INDEX_COMPRESS_INFO 表関数 - 圧縮索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_INDEX_INFO 表関数 - 索引情報を戻す	ACTIVITY METRICS BASE
ADMIN_GET_TAB_COMPRESS_INFO 表関数 - 圧縮節約量の見積もり	ACTIVITY METRICS BASE
ADMIN_GET_TAB_DICTIONARY_INFO 表関数 - 既存の表ディクショナリーのプロパティーの報告	ACTIVITY METRICS BASE
ADMINTABINFO 管理ビューおよび ADMIN_GET_TAB_INFO 表関数 - 表のサイズおよび状態に関する情報の検索	ACTIVITY METRICS BASE
ADMINTEMPCOLUMNS 管理ビューおよび ADMIN_GET_TEMP_COLUMNS 表関数 - 一時表の列情報を取得する	ACTIVITY METRICS BASE

表 2258. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
ADMINTEMPTABLES 管理ビューおよび ADMIN_GET_TEMP_TABLES 表関数 - 一時表の情報を取得する	ACTIVITY METRICS BASE
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_INDEX 表関数 - 索引メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PAGE_ACCESS_INFO 表関数 - バッファー・プール・ページの待機情報の取得	ACTIVITY METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	ACTIVITY METRICS BASE
MON_GET_TABLE_USAGE_LIST 表関数 - 表使用リストからの情報を返す	ACTIVITY METRICS BASE

表 2259. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	ロック
ロック	lock_wait	ロック

表 2260. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
表	event_table	常に収集される
デッドロック[1]	lock	常に収集される
デッドロック[1]	event_dlconn	常に収集される
詳細付きデッドロック[1]	event_detailed_dlconn	常に収集される

1 このイベント・モニターは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントと **table_name** を組み合わせて使用すると、リソースの競合の原因を判別できます。

アプリケーション・レベル、アプリケーション・ロック・レベル、およびデッドロック・モニター・レベルでは、現在別のアプリケーションにロックされているため

にロック待ちになっているアプリケーションの表のスキーマを示します。このエレメントは、アプリケーションが表ロックの取得待ちであることを **lock_object_type** が示している場合にのみ設定できます。アプリケーション・レベルおよびアプリケーション・ロック・レベルでのスナップショット・モニターの場合、この項目は「lock」モニター・グループ情報が ON に設定されている場合にのみ有効です。

オブジェクト・レベルのスナップショット・モニターの場合は、表レベルおよび行レベルのロックについてこの項目が戻されます。このレベルで報告される表は、このアプリケーションがこれらのロックを保留している表です。

表レベルのスナップショット・モニターおよびイベント・モニターの場合は、このエレメントは情報が収集された表スキーマを示します。一時表の場合、**table_schema** の形式は 『*<agent_id><auth_id>*』 です。

- *agent_id* は、一時表を作成しているアプリケーションのアプリケーション・ハンドルです。
- *auth_id* は、アプリケーションがデータベースに接続するときに使用する許可 ID です。

tablespace_min_recovery_time - ロールフォワードの最小リカバリー時間 : モニター・エレメント

表スペースをロールフォワードできる最も早い時点を示すタイム・スタンプ。このタイム・スタンプは、ローカル時間を表します。

表 2261. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	ACTIVITY METRICS BASE

表 2262. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

ゼロ以外のときだけ表示されます。

target_cf_gbp_size - ターゲット・クラスター・キャッシング・ファシリティーのグループ・バッファー・プール・サイズ : モニター・エレメント

このモニター・エレメントは、動的サイズ変更の際に、グループ・バッファー・プール・メモリーのターゲット値をページ・サイズ 4 KB のページ単位で表示します。ターゲット値が構成値と一致すると、サイズ変更が完了します。

表 2263. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

target_cf_lock_size - ターゲット・クラスター・キャッシング・ファシリティーのロック・サイズ : モニター・エレメント

このモニター・エレメントは、動的サイズ変更の際に、グローバル・ロック・メモリーのターゲット値をページ・サイズ 4 KB のページ単位で表示します。ターゲット値が構成値と一致すると、サイズ変更が完了します。

表 2264. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

target_cf_sca_size - ターゲット・クラスター・キャッシング・ファシリティーの共用通信域サイズ : モニター・エレメント

このモニター・エレメントは、動的サイズ変更の際に、共用通信域メモリーのターゲット値をページ・サイズ 4 KB のページ単位で表示します。ターゲット値が構成値と一致すると、サイズ変更が完了します。

表 2265. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CF 表関数 - CF メトリックの取得	ACTIVITY METRICS BASE

第 4 部 付録

付録 A. DB2 技術情報の概説

DB2 技術情報は、さまざまな方法でアクセスすることが可能な、各種形式で入手できます。

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、IBM Publications Center (www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) から利用できる DB2 ライブラリーについて説明しています。英語および翻訳された DB2 バージョン 10.1 のマニュアル (PDF 形式) は、www.ibm.com/support/docview.wss?rs=71&uid=swg27009474 からダウンロードできます。

この表には印刷資料が入手可能かどうかが示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: *DB2 インフォメーション・センター*は、PDF やハードコピー資料よりも頻繁に更新されます。

表 2266. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SA88-4671-00	入手可能	2012 年 4 月
管理ルーチンおよびビュー	SA88-4672-01	入手不可	2013 年 1 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻	SA88-4676-01	入手可能	2013 年 1 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻	SA88-4677-01	入手可能	2013 年 1 月
コマンド・リファレンス	SA88-4673-01	入手可能	2013 年 1 月
データベース: 管理の概念および構成リファレンス	SA88-4662-01	入手可能	2013 年 1 月
データ移動ユーティリティー: ガイドおよびリファレンス	SA88-4693-01	入手可能	2013 年 1 月
データベースのモニタリング ガイドおよびリファレンス	SA88-4663-01	入手可能	2013 年 1 月
データ・リカバリーと高可用性 ガイドおよびリファレンス	SA88-4694-01	入手可能	2013 年 1 月
データベース・セキュリティー・ガイド	SA88-4695-01	入手可能	2013 年 1 月

表 2266. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 ワークロード管理ガイドおよびリファレンス	SA88-4685-01	入手可能	2013 年 1 月
ADO.NET および OLE DB アプリケーションの開発	SA88-4665-01	入手可能	2013 年 1 月
組み込み SQL アプリケーションの開発	SA88-4666-01	入手可能	2013 年 1 月
Java アプリケーションの開発	SA88-4669-01	入手可能	2013 年 1 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SA88-4670-00	入手不可	2012 年 4 月
IBM データ・サーバー用の RDF アプリケーション開発	SA88-5083-00	入手可能	2013 年 1 月
SQL および外部ルーチンの開発	SA88-4667-01	入手可能	2013 年 1 月
データベース・アプリケーション開発の基礎	GI88-4279-01	入手可能	2013 年 1 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4280-00	入手可能	2012 年 4 月
グローバリゼーション・ガイド	SA88-4696-00	入手可能	2012 年 4 月
DB2 サーバー機能 インストール	GA88-4679-01	入手可能	2013 年 1 月
IBM データ・サーバー・クライアント機能 インストール	GA88-4680-00	入手不可	2012 年 4 月
メッセージ・リファレンス 第 1 巻	SA88-4688-01	入手不可	2013 年 1 月
メッセージ・リファレンス 第 2 巻	SA88-4689-01	入手不可	2013 年 1 月
Net Search Extender 管理およびユーザーズ・ガイド	SA88-4691-01	入手不可	2013 年 1 月
パーティションおよびクラスタリングのガイド	SA88-4697-01	入手可能	2013 年 1 月
Preparation Guide for DB2 10.1 Fundamentals Exam 610	SC27-4540-00	入手不可	2013 年 1 月

表 2266. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611	SC27-4541-00	入手不可	2013 年 1 月
pureXML ガイド	SA88-4686-01	入手可能	2013 年 1 月
Spatial Extender ユーザーズ・ガイドおよびリファレンス	SA88-4690-00	入手不可	2012 年 4 月
SQL プロシージャー言語: アプリケーションのイネーブルメントおよびサポート	SA88-4668-01	入手可能	2013 年 1 月
SQL リファレンス 第 1 巻	SA88-4674-01	入手可能	2013 年 1 月
SQL リファレンス 第 2 巻	SA88-4675-01	入手可能	2013 年 1 月
Text Search ガイド	SA88-4692-01	入手可能	2013 年 1 月
問題判別およびデータベース・パフォーマンスのチューニング	SA88-4664-01	入手可能	2013 年 1 月
DB2 バージョン 10.1 へのアップグレード	SA88-4678-01	入手可能	2013 年 1 月
DB2 バージョン 10.1 の新機能	SA88-4684-01	入手可能	2013 年 1 月
XQuery リファレンス	SA88-4687-01	入手不可	2013 年 1 月

表 2267. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Connect Personal Edition インストールおよび構成	SA88-4681-00	入手可能	2012 年 4 月
DB2 Connect サーバー機能 インストールおよび構成	SA88-4682-01	入手可能	2013 年 1 月
DB2 Connect ユーザーズ・ガイド	SA88-4683-01	入手可能	2013 年 1 月

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果として生じる可能性がある状態に対応した SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順

SQL 状態ヘルプを開始するには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。
例えば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

他のバージョンの DB2 製品の資料は、ibm.com® のそれぞれのインフォメーション・センターにあります。

このタスクについて

DB2 バージョン 10.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v10r1 です。

DB2 バージョン 9.8 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/ です。

DB2 バージョン 9.7 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/ です。

DB2 バージョン 9.5 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5 です。

DB2 バージョン 9.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9/ です。

DB2 バージョン 8 のトピックについては、*DB2 インフォメーション・センター* の URL (http://publib.boulder.ibm.com/infocenter/db2luw/v8/) を参照してください。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

ローカルにインストールした DB2 インフォメーション・センターは、定期的に更新する必要があります。

始める前に

DB2 バージョン 10.1 インフォメーション・センターが既にインストール済みである必要があります。詳しくは、「*DB2 サーバー機能 インストール*」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

このタスクについて

既存の DB2 インフォメーション・センターは、自動で更新することも手動で更新することもできます。

- 自動更新は、既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、手動更新と比べて、更新中にインフォメーション・センターが使用できなくなる時間が短くなるというメリットがあります。さらに、自動更新は、定期的に実行する他のバッチ・ジョブの一部として実行されるように設定することができます。
- 手動更新は、既存のインフォメーション・センターのフィーチャーと言語の更新に使用できます。自動更新は更新処理中のダウン時間を減らすことができますが、フィーチャーまたは言語を追加する場合は手動処理を使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。自動更新処理では、インフォメーション・センターは、更新を行った後に、インフォメーション・センターを再始動するための停止が発生するだけで済みます。

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

手順

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動更新する手順を以下に示します。

1. Linux オペレーティング・システムの場合、次のようにします。
   a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`/opt/ibm/db2ic/V10.1` ディレクトリーにインストールされています。
   b. インストール・ディレクトリーから `doc/bin` ディレクトリーにナビゲートします。
   c. 次のように `update-ic` スクリプトを実行します。

      ```
      update-ic
      ```
2. Windows オペレーティング・システムの場合、次のようにします。
   a. コマンド・ウィンドウを開きます。
   b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`<Program Files>¥IBM¥DB2 Information Center¥バージョン 10.1` ディレクトリーにインストールされています (`<Program Files>` は「Program Files」ディレクトリーのロケーション)。

c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。

d. 次のように update-ic.bat ファイルを実行します。

```
update-ic.bat
```

タスクの結果

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、doc¥eclipse¥configuration ディレクトリーにあります。ログ・ファイル名はランダムに生成された名前です。例えば、1239053440785.log のようになります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

このタスクについて

ローカルにインストールされた *DB2 インフォメーション・センター* を手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の *DB2 インフォメーション・センター* を停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。DB2 インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

   注: ご使用の環境において、インターネットに接続されていないマシンに *DB2 インフォメーション・センター* の更新をインストールする必要がある場合、インターネットに接続されていて *DB2 インフォメーション・センター* がインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシーを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。
   更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。
3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の *DB2 インフォメーション・センター* を再開します。

注: Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、「**管理者として実行**」を選択します。

手順

コンピューターまたはイントラネット・サーバーにインストール済みの *DB2* インフォメーション・センター を更新するには、以下のようにします。

1. *DB2* インフォメーション・センター を停止します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**停止**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv10 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、*Program_Files*¥IBM¥DB2 Information Center¥バージョン 10.1 ディレクトリーにインストールされています (*Program_Files* は Program Files ディレクトリーのロケーション)。
     c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
     d. 次のように help_start.bat ファイルを実行します。

        ```
        help_start.bat
        ```
   - Linux の場合:
     a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、/opt/ibm/db2ic/V10.1 ディレクトリーにインストールされています。
     b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
     c. 次のように help_start スクリプトを実行します。

        ```
        help_start
        ```

   システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「**更新**」ボタン () をクリックします。(ブラウザーで JavaScript が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「**更新の検索**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。

6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
   - Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

     ```
     help_end.bat
     ```

     注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。
   - Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

     ```
     help_end
     ```

     注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを停止しないでください。
7. *DB2 インフォメーション・センター* を再開します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv10 start
     ```

タスクの結果

更新された *DB2 インフォメーション・センター* に、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 データベース製品のさまざまな機能について学習するための支援となります。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML ガイド* 」の『**pureXML**』
: XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 の資料

トラブルシューティング情報は、「*問題判別およびデータベース・パフォーマンスのチューニング*」または *DB2 インフォメーション・センター* の『データベースの基本』セクションにあります。ここには、以下の情報が記載されています。

- DB2 診断ツールおよびユーティリティーを使用した、問題の切り分け方法および識別方法に関する情報。
- 最も一般的な問題のうち、いくつかの解決方法。
- DB2 データベース製品で発生する可能性のある、その他の問題の解決に役立つアドバイス。

IBM サポート・ポータル

現在問題が発生していて、考えられる原因とソリューションを見つけるには、IBM サポート・ポータルを参照してください。 Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

IBM サポート・ポータル (http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows) にアクセスしてください。

ご利用条件

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

IBM の商標: IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、

利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは、現存するままの状態で提供されるものであり、いかなる種類の保証も提供されません。 IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*お客様の会社名*) (*西暦年*). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _*年を入れる*_. All rights reserved.

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- UNIX は The Open Group の米国およびその他の国における登録商標です。
- インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Celeron、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、 Windows、 Windows NT、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

A

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

[特殊文字]

Printed in Japan

SA88-4663-01

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

データベースのモニタリング ガイドおよびリファレンス