IBM DB2 10.1
for Linux, UNIX, and Windows

DB2 ワークロード管理 ガイドおよびリファレンス

2013 年 1 月更新版

SA88-4685-01
(英文原典：SC27-3891-01)

IBM DB2 10.1
for Linux, UNIX, and Windows

DB2 ワークロード管理 ガイドおよびリファレンス

2013 年 1 月更新版

SA88-4685-01
(英文原典：SC27-3891-01)

ご注意

本書および本書で紹介する製品をご使用になる前に、 579 ページの『付録 E. 特記事項』に記載されている情報をお読みください。

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

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

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

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

原典： SC27-3891-01
IBM DB2 10.1
for Linux, UNIX, and Windows
DB2 Workload Management Guide and Reference
Updated January, 2013

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.12

目次

本書について

本書には、業務目的にかなった、安定した予測可能な実行環境を実現するために役立つ DB2® ワークロード管理のフィーチャーと機能についての情報が記載されています。 DB2 ワークロード管理を使用すると、要求とリソースの両方が管理されます。本書では、データ・サーバー上のワークロードについてのモニター、およびトラブルシューティングの実行に関する情報を提供します。

第 1 章 DB2 ワークロード管理の概念の紹介

効果的なワークロード管理システムは、作業が発生する環境において、目標を効率的に達成する上で役立ちます。効果的なワークロード管理システムの必要性を示す例は、枚挙にいとまがありません。

例えば、食料品店について考えてみましょう。そこでは、顧客へのサービス提供、棚への陳列、在庫管理などのさまざまな活動、つまりアクティビティーを考慮する必要があります。そして、シンプルな目標も設定するでしょう。店主は、店内を歩き回る顧客数と顧客が購入する品数の両方を最大にするという 2 つの目標を達成すると同時に、顧客が満足し、再度来店したいという願いを持って店舗から出ることを願っています。店主は、買い物をする顧客のために十分な量の在庫が必ずあるようにする必要もあります (ただし無駄が発生すると問題となるため、在庫は過剰にならないようにします)。店主はさらに、顧客が購入したものを追跡記録し、この情報を使って、顧客が再度来店するよう促すことを意図した広告を作成します。モニター機構で在庫を追跡し、在庫が少なくなったら通知を送信するようにします。万引きを検出するためにセキュリティー装置を設置します。数品目しか購入しない買い物客が、たくさんの品目を購入する他の顧客の後ろで待たずに購入できるように、特別の優先レジを作ります。これらすべての目標が達成され、これらすべての運用手順が適切に機能するなら、顧客は満足し、別の店舗に行かずにまた来店するでしょう。これらの目標と運用手順はすべて、ワークロード管理の一環として行われます。

データ・サーバー環境では、作業をさらに効果的に管理する必要があります。データ・サーバーがかつてないほど重視されている現在においてはなおさらそのようにいえます。何千というデータ挿入がレジで生成されます。売上目標が達成されているかを調べるためにレポートが常に生成され、収集データをロードするためにバッチ・アプリケーションが実行され、さらにデータを保護してサーバーを最適な状態で実行させるためにバックアップと再編成などの管理タスクが実行されます。こうしたすべての操作はすべて同じデータベース・システムを使用していて、なおかつ同じリソースを獲得するために競合しています。

データ・サーバーを実行するための目標を確実に達成するためには、効率的なワークロード管理システムがどうしても求められます。

ワークロード管理のドメイン

ワークロード管理には、はっきり定義された次の 3 つのドメインがあります。つまり、データ・サーバーに入る作業の識別、実行中の作業の管理、およびデータ・サーバーが効率的に使用されていることを確認するためのモニターというドメインです。

DB2 ワークロード・マネージャーでワークロード管理を適切に行うには、目標を明確にすることから始め、多くの側面を考慮する必要があります。 『第 1 章 DB2 ワークロード管理の概念の紹介』で説明されている食料品店を例に取ると、目標に

は、顧客の購入金額を最大にすること、万引きを最小にすること、再度来店してもらえるように顧客が満足して店を出るようにすることが含まれるかもしれません。

データ・サーバー環境でも目標を定義する必要があります。目標は明確なものである場合もあります。サービス・レベル契約 (SLA) を目的として考え出された目標であれば特にそういえます。例えば、特定のアプリケーションからの照会が消費できる量を、合計プロセッサー・リソースの 10% を超えないようにします。また目標を、特定の時刻に関連付けることができます。例えば、毎日の販売レポートが時間どおりに作成されるように、夜間バッチ・ユーティリティーは、データのロードを午前 8 時までに完了する必要があるかもしれません。他の状況では、目標を定量化することが難しい場合もあります。目標は、データベースのユーザーの満足度を維持し、異常なデータベース・アクティビティーが発生してユーザーの日常の作業を妨げることがないようにする、というものかもしれません。目標を定量化できるものであってもそうでなくても、ワークロード管理についての以下の各ステージを考慮する際に目標を明確にすることは非常に重要です。

識別 何らかの作業の目標を達成しようとする場合、まず作業に関する詳細を識別できなければなりません。食料品店では、買い物客情報はクレジット・カードとデビット・カードから識別できますし、未払いの品目はその品目に付いているアクティブなセキュリティー・タグから識別できます。データ・サーバーの場合には、システムに存在する作業を識別する方法を決定する必要があります。その作業をサブミットするアプリケーションの名前や許可 ID、またはある種の ID を提供する要素を組み合わせて使用することができます。

管理 管理フェーズには、目標に向かって着実に進行していくメカニズムと、目標を達成できなかった場合に取るアクションが含まれます。メカニズムの例には、優先レジでの価格確認管理があります。優先レジを設置するとスループットが早くなり顧客は満足するはずですが、牛乳のカートンに間違った値段が付いていて価格確認が必要になった場合、優先レジの流れは低下してしまいます。その問題の管理方法として、可能ならば別のレジを開き、迅速に価格確認をし、こうしたことが再度起きないように価格の問題を解決するようにします。データ・サーバー上では、記述が悪い SQL ステートメントがいくつか実行されていたり、ピーク時にボリュームが急激に多くなったり、同じリソースに対して異なるアプリケーションによる競合があまりにも多かったりして、全体的なパフォーマンスが悪いことに気付くかもしれません。管理フェーズには、目標を達成するためにリソースを割り当てるメカニズムと、目標を達成できない場合に取るべきアクションが含まれます。 (CPU リソースを間接的および直接的に制御する) これらのワークロード管理メカニズムには、以下のものが含まれます。

1. 並行性しきい値。これは着信する作業の並行性を制御するために、ワークロードで定義されるワーク・アクション・セットで適用されます。
2. あるサービス・クラスから別のサービス・クラスへと作業を移動する機能。現在は、ワークロード管理ディスパッチャーがこれらのサービス・クラスで使用されている場合にのみ有効です。
3. ワークロード管理ディスパッチャー。これはサービス・クラスに割り当てられたワークロードに CPU リソースを特に割り振ります。最初の 2 つのワークロード管理メカニズムでは不十分である場合に CPU リソースをより細かく制御できます。

モニター

モニターが重要である理由は、いくつかあります。第 1 に、目標を達成しているかを調べるには、その目標への進行状況を追跡するメカニズムが必要です。さらに、モニターを行うと、目標の達成を妨げている可能性がある問題を識別する上で役立ちます。店舗においては、店主は顧客の流れを監視したり、万引き、特定の販売品目の在庫が危険なレベルまで不足した状態などの問題の警告を自動的に受け取るようにしたり、店舗での製品の最適な配置方法を判断するために過去の消費行動パターンを分析したりすることができます。データ・サーバーにおいては、データベース・アクティビティーの応答時間には通常明示的な目標があるため、この測定基準について測定するための方法があることと傾向を監視することは重要です。

以下の図は、ワークロード管理のフェーズを示しています。

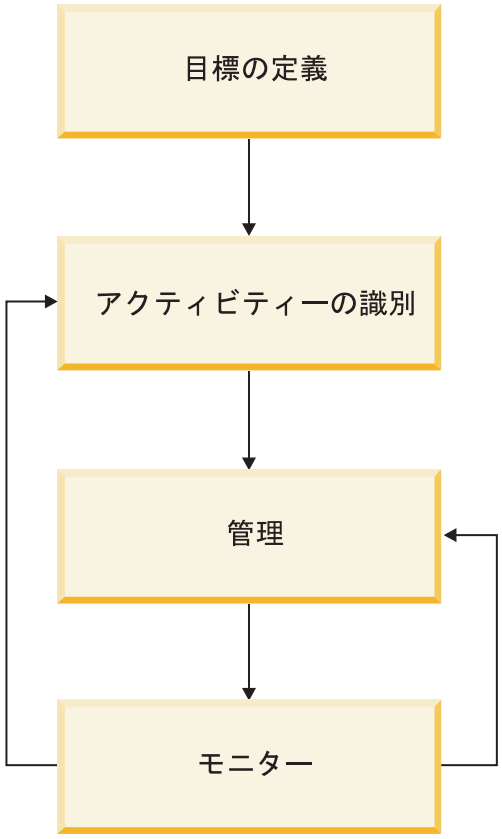


図 1. ワークロード管理のフェーズ

ワークロード管理管理者権限 (WLMADM)

特定のデータベースのワークロード・オブジェクトを管理するには、ワークロード管理管理者 (WLMADM) 権限が必要です。この権限があれば、DB2 ワークロード管理オブジェクトに関して、作成、変更、ドロップ、コメント作成、およびアクセス権限の付与と取り消しを行えます。

ワークロード管理オブジェクトは、バッファー・プールおよび表スペースと同様のシステム・オブジェクトです。このため、それらに関連付けられている所有者はいません。

セキュリティー管理者 (SECADM 権限を保持するユーザー) または ACCESSCTRL 権限を持つユーザーは、WLMADM 権限をユーザー、グループ、またはロールに付与できます。

WLMADM 権限により、以下のタスクを実行できます。

- 次の DB2 ワークロード管理オブジェクトに関して CREATE、 ALTER、COMMENT ON、および DROP ステートメントを発行する。
  - ヒストグラム・テンプレート
  - サービス・クラス
  - しきい値
  - ワーク・アクション・セット
  - ワーク・クラス・セット
  - ワークロード
- ワークロード特権に関して GRANT および REVOKE ステートメントを発行する。

データベース管理者 (DBADM) 権限を保持している場合、暗黙的に WLMADM 権限を保持していることになります。

DB2 ワークロード管理に関してよくある質問

この FAQ では、DB2 ワークロード管理について、よくある質問に回答します。

一般

- DB2 ワークロード管理は、どの DB2 プラットフォームで使用できますか。
- 現在、AIX® を使用していません。この場合、プロセッサー・リソースや I/O アクティビティーをまったく制御できないことになりますか。
- Query Patroller が廃止され、DB2 ガバナーが非推奨になりましたが、DB2 ワークロード・マネージャーにどのようにマイグレーションすればよいですか。
- DB2 ワークロードが使用するクライアント情報フィールドを WebSphere® Application Server が受け渡す方法はありますか。
- 作業が正しいワークロードに割り当てられていないのはなぜですか。
- DB2 ワークロード・マネージャーが REORGCHK、IMPORT、EXPORT、その他の CLP コマンドに影響するのはなぜですか。
- アクティビティーの実行中にその割り当て先のサービス・クラスを変更する方法はありますか。
- バッチ作業の大部分が、同一の ID で CLP スクリプトを使用して行われていますが、それらを一意的に識別して個別に管理するにはどのようにすればよいですか。

- COLLECT AGGREGATE ACTIVITY DATA 節は、COLLECT ACTIVITY DATA 節とどのように使い分ければよいのですか。
- DB2 ワークロード管理は新規の AIX WPAR (ワークロード・パーティション) フィーチャーとどのように連動しますか。
- DB2_OPT_MAX_TEMP_SIZE レジストリー変数と、SQLTEMPSPACE に基づく DB2 しきい値との関係はどのようなものですか。

ライセンス交付

- DB2 ワークロード・マネージャーに関するどのようなライセンス交付要件がありますか。

モニター

- ワークロード管理に関連した種々のイベント・モニターからはどのような情報を取得できますか。

OS ワークロード管理 (AIX WLM および Linux WLM)

- AIX WLM または Linux WLM を使用すべき理由は何ですか。
- 現在、AIX を使用していません。この場合、プロセッサー・リソースや I/O アクティビティーをまったく制御できないことになりますか。
- I/O アクティビティーの管理に AIX WLM を使用できますか。
- メモリー使用の管理に AIX WLM を使用できますか。
- DB2 WLM は新規の AIX WPAR (ワークロード・パーティション) フィーチャーとどのように連動しますか。

プラットフォーム

- DB2 ワークロード・マネージャーは、どの DB2 プラットフォームで使用できますか。
- 現在、AIX を使用していません。この場合、プロセッサー・リソースや I/O アクティビティーをまったく制御できないことになりますか。
- DB2 ワークロード管理は新規の AIX WPAR (ワークロード・パーティション) フィーチャーとどのように連動しますか。
- AIX WLM または Linux WLM を使用すべき理由は何ですか。

Query Patroller およびガバナー

- この新機能は Query Patroller と DB2 ガバナーにどのような影響を与えますか。
- Query Patroller と DB2 ガバナーが非推奨になりましたが、DB2 ワークロード・マネージャーにどのようにマイグレーションすればよいですか。

しきい値

- 同一の作業セットに対して複数の CONCURRENTDBCOORDACTIVITIES 並行性しきい値を作成することはできますか。
- ワークロード管理しきい値によってキューに入れられるアクティビティーと、キューにおけるそれらのアクティビティーの順序はどのようにして判別できますか。

ワークロード管理ディスパッチャー

- ワークロード管理ディスパッチャーを使用する必要がありますか。

- ワークロード管理ディスパッチャーを有効にすると、どのように動作が変化する可能性がありますか。
- ワークロード管理ディスパッチャーの導入により、CONCURRENTDBCOORDACTIVITIES などの並行性しきい値は不要になりましたか。

DB2 ワークロード管理は、どの DB2 プラットフォームで使用できますか。

DB2 ワークロード管理は、DB2 9.5 for Linux, UNIX, and Windows 以降でサポートされるすべてのプラットフォームで使用可能です。オプションの緊密な統合（サービス・クラス・レベルでの DB2 ワークロード管理と、オペレーティング・システムのワークロード管理の機能との間で提供される）は、AIX プラットフォーム、および 2.6.26 以降のカーネルに基づくすべての Linux プラットフォームで使用可能です。

ワークロード管理ディスパッチャーを使用する必要がありますか。

大半のワークロード管理の構成は、並行性しきい値から始まります。これは、同時に実行を開始できる作業の量を制御することによって、すべてのリソースの消費に影響します。ただし、並行性しきい値が、消費される処理リソースの合計を効率よく制限することができず、優先順位の高い作業が影響を受ける場合があります。例えば、複雑な作業が 1 つの照会の実行のみに制限されているが、これが優先順位の高い作業を阻害するほどのリソースを消費するシナリオの場合などです。このような場合、ワークロード管理ディスパッチャーを使用して、明示的に CPU 消費を制御し、優先順位の高い作業を保護します。

以下のいずれか (またはすべて) の状況で、ワークロード管理ディスパッチャーを使用することができます。

- 複数のユーザー間またはアプリケーション間の CPU リソース共有を管理する必要があるが、使用中のオペレーティング・システムには、各サービス・クラスの `outbound_correlator` フィールドを介して DB2 ワークロード管理と統合するオペレーティング・システム (OS) ワークロード・マネージャーが存在しない場合。
- 複数のユーザー間またはアプリケーション間の CPU リソース共有を管理する必要があるが、オペレーティング・システムのルート特権を持っていない場合。
- 複数のシステムにわたるマルチ・メンバー環境で複数のユーザー間またはアプリケーション間の CPU リソース共有を管理する必要があるが、各システムの OS WLM を使ってこれを管理すると多大な管理操作が必要になる場合。
- CPU がそれほど使用されていないときでも、特定のサービス・クラスを制限するためにハード共有を使って複数のユーザー間またはアプリケーション間の CPU リソース共有を管理する必要があるが、OS WLM ではそれができない場合、または期待される結果が出ない場合。

この新機能は Query Patroller と DB2 ガバナーにどのような影響を与えますか。

DB2 ワークロード・マネージャーでは、ワークロード管理に独立したアプローチを導入しており、Query Patroller や DB2 ガバナーにはまったく依存も相互作用もし

ません。バージョン 10.1 のリリース以降、Query Patroller は廃止されました。DB2 ガバナーは DB2 バージョン 9.7 リリースで非推奨になりました。引き続き機能しますが、DB2 ワークロード管理ストラテジーの中心的な機能ではなくなりました。将来のリリースにおいて DB2 ガバナーに対するこれ以上の投資は予定されていません。

DB2 9.5 以降が初めてインストールされると、デフォルトのユーザー・サービス・クラスが自動的に定義されて、すべての作業はそこで実行されます。 DB2 ガバナーは、どのサービス・クラスのエージェントでも監視できますが、エージェント優先順位を調整できるのはデフォルトのユーザー・サービス・クラスのエージェントについてのみです。

現在、AIX を使用していません。この場合、プロセッサー・リソースや I/O アクティビティーをまったく制御できないことになりますか。

すべてのプラットフォームのユーザーは、SQL (例えば CREATE ステートメントや ALTER SERVICE CLASS ステートメント) を使用して、サービス・クラス間のプロセッサー・リソースおよび I/O アクティビティーを同じように制御できます。

ワークロード管理ディスパッチャーが有効になっているときに CPU 使用量を制御するには、DB2 サービス・クラスの CPU リミット属性を使用して、サービス・クラスで消費できる CPU リソースの数量を制限します。さらに、ワークロード管理ディスパッチャー CPU シェア (**wlm_disp_cpu_shares**) のデータベース・マネージャー構成パラメーターも有効になっている場合、DB2 サービス・クラスの CPU シェア属性を使用することにより、他のサービス・クラスによる CPU 消費と比較して、そのサービス・クラスで消費できる CPU リソースの割り当て分を指定することができます。 AIX および一部の Linux プラットフォームでは、これらのオペレーティング・システムによって提供されるワークロード管理機能を活用して CPU 消費を制御することにより、これらのアプローチを補足 (あるいは置換) することができます。

I/O アクティビティーの場合、すべてのプラットフォームのユーザーが DB2 サービス・クラスのバッファー・プールまたはプリフェッチャー優先順位属性を高、中、低の値に設定できます。すべてのサービス・クラスは、デフォルトでは優先順位が中で実行します。

I/O アクティビティーを管理するために AIX または Linux の WLM、または DB2 ワークロード管理ディスパッチャーを使用できますか。

現在、AIX WLM と Linux WLM は、どちらもスレッド・レベルの I/O アクティビティー制御をサポートしていません。ただし、並行性しきい値を使用して I/O アクティビティーを間接制御したり、CPU リソースを操作するために DB2 ワークロード管理ディスパッチャー、AIX WLM、あるいは Linux WLM を使用したりすることは可能です。実行中のスレッドで使用可能な CPU リソースが増えれば増えるほど、そのスレッドが I/O リソースを要求する頻度は少なくなります。

任意の DB2 サービス・クラスの BUFFERPOOL PRIORITY 属性を使用することによって、バッファー・プールの振る舞いに影響を与えることができます。また、いずれかの DB2 サービス・クラスの PREFETCH PRIORITY 属性を使用して DB2

プリフェッチャー I/O アクティビティーを制御することもできます。

メモリー使用の管理に AIX WLM または Linux WLM を使用できますか。

DB2 データ・サーバーは、さまざまなサービス・クラスからの複数のエージェントがアクセスする共用メモリーを主に使用します。このため、AIX WLM と Linux WLM のどちらを使用しても、異なるサービス・クラス間でメモリーの割り振りを分割することはできません。

ステートメントの実行が許可されるまで消費は開始されないため、SQL ステートメントの実行中に消費されるメモリー (ソート・ヒープなど) は、並行性しきい値の使用によって間接的に影響を受ける場合があります。ただし、I/O アクティビティーとは異なり、CPU 使用量の制限は、消費されるメモリーの量には影響しません。実際、CPU 使用量を制限することにより、メモリーの状態が悪化する可能性があります。照会の実行速度が遅くなり、割り振られたメモリーが保持される時間が長くなるためです。

DB2 ワークロードが使用するクライアント情報フィールドを WebSphere Application Server が受け渡す方法はありますか。

WebSphere Application Server バージョン 6.0 およびバージョン 6.1 は、CLIENT INFO フィールドを DB2 データ・サーバーに設定または受け渡すことができます。これには、ご使用のアプリケーションで明示的に行う方法 (クライアント情報のデータベースへの引き渡しを参照) と、WebSphere Application Server によって暗黙的に行う方法 (暗黙的に設定されるクライアント情報を参照) があります。

同一の作業セットに対して複数の CONCURRENTDBCOORDACTIVITIES 並行性しきい値を作成することはできますか。

データベースのレベル、作業が実行されるサービス・クラス、またはデータベース・レベルあるいはワークロード・レベルで適用されるワーク・アクション・セットの中でこのしきい値を定義することによって、同一セットのアクティビティーに適用される 1 つ以上の CONCURRENTDBCOORDACTIVITIES 並行性しきい値を作成できます。アクティビティーに新しい並行性しきい値が適用されるたびに、この並行性しきい値を強制するためのオーバーヘッドが増えることは認識しておいてください。複数の並行性しきい値レベルが本当に必要かどうかを検証してください。

作業が正しいワークロードに割り当てられていないのはなぜですか。

接続が所定のワークロードにマップされない理由は多数あります。最もよく見られるのは、ワークロードに対する USAGE 特権に関して問題がある、接続属性の大/小文字の区別でスペルが正しくない、または評価順序の前の段階でワークロード定義が一致するものがあったという理由です。

ワークロードに接続を割り当てられるようにするには、その前に接続属性が、ワークロード定義の接続属性と一致しており、セッション許可 ID がそのワークロードに対する USAGE 特権を持っている必要があります。よく見られる失敗に、ワークロードを作成したものの、そのワークロードに対する USAGE 特権をユーザーに付与していなかったというものがあります (「*SQL リファレンス*」の『GRANT (ワー

クロード特権) ステートメント』を参照)。ワークロードの使用特権を他のユーザーに付与できるのは、ACCESSCTRL 権限、 SECADM 権限、または WLMADM 権限を持つユーザーのみです。 ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM 権限を持つユーザーには、すべてのワークロードに対して暗黙的な使用特権があります。

ワークロードの接続属性には、大/小文字の区別があります。例: システム・ユーザー ID が大文字の場合、指定する SYSTEM_USER 接続属性も大文字でなければなりません。

接続が、期待したワークロードにマップされない理由を特定するには、いくつかの情報を収集する必要があります。作業のマップ先のワークロードはどれか。そのワークロードは、ワークロード定義を評価順序で調べる際に使用されると思われるワークロードの前なのか、後になるのか (ヒント: SYSCAT.WORKLOADS の EVALUATIONORDER 列の値を使用して昇順に並べ、ワークロード定義を選択してみます)。

ターゲット接続に使用する接続属性がどれかわからない場合は、以下のようにさまざまな方法で接続に使用する値を見つけることができます。

- 接続がアクティブ状態のときに、WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を使ってシステムに対する照会を発行します。
- 接続上でカーソルを開き、そのカーソルに対して WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを使用して、アクティビティー・イベント・モニターにアクティビティー情報をキャプチャーします (ヒント: アクティビティー情報イベント・モニターを必ず作成し、活動化してください)。
- 接続で使用されているワークロードに関する詳細なアクティビティー情報の収集をオンにし、アクティビティー情報をキャプチャーするためのステートメントを 1 つ発行し、収集をオフにします。

DB2 ワークロード・マネージャーが REORGCHK、IMPORT、EXPORT、その他の CLP コマンドに影響するのはなぜですか。

これらの CLP コマンドは、DB2 ワークロード管理のしきい値の影響を受けます。データベース・エンジンは、これらのユーティリティーを使ったシステム要求と、ユーザーが直接 CLP 内で対話的に行う要求とを区別できないためです。

アクティビティーの実行中にその割り当て先のサービス・クラスを変更する方法はありますか。

はい。元のサービス・サブクラスに対する REMAP ACTIVITY アクションで CPUTIMEINSC、DATATAGINSC または SQLROWSINSC しきい値を定義することにより、アクティビティーが実行されているサービス・サブクラスを、同じ親サービス・スーパークラス内の別のサービス・サブクラスに変更できます。 DB2 ワークロード管理は、最初に接続の関連ワークロード定義に基づいてサービス・クラスにアクティビティーをマップし、そのサービス・クラスにワーク・アクション・セットが存在する場合は要求に応じてアクティビティーを変更します。次に DB2 エ

ージェントを、割り当てられたサービス・クラスで実行されるようセットアップします。 REMAP ACTIVITY アクションが定義されているしきい値にアクティビティーが違反した場合、しきい値違反が検出されるとエージェントは指定されたターゲット・サービス・サブクラス (同一スーパークラス下) に自身を再マップするので、アクティビティーは新しいサービス・サブクラスで実行を続けます。

バッチ作業の大部分が、同一の ID で CLP スクリプトを使用して行われていますが、それらを一意的に識別して個別に管理するにはどのようにすればよいですか。

次の 2 つの選択肢があります。

クライアント・アプリケーション名が自動的に CLP スクリプト・ファイル名に設定されるように、CLP に機能拡張が追加されました。このファイル名の先頭には **CLP** という接頭部が付けられます (サーバーでは、このフィールドの値を CURRENT CLIENT_APPLNAME 特殊レジスターで調べることができます)。例えば、CLP スクリプト・ファイル名が **batch.db2** の場合、このスクリプトの実行時に、CLP によって CURRENT CLIENT_APPLNAME 特殊レジスターの値が **CLP batch.db2** に設定されます。このフィーチャーを使用すると、クライアント・アプリケーション名に基づいて、異なる CLP スクリプトを異なるワークロードに関連付けることができます。

例えば、CLP ファイル batch1.db2 にワークロードを作成するには、次のような DDL ステートメントを発行できます。

```
CREATE WORKLOAD batch1 CURRENT CLIENT_APPLNAME ('CLP batch1.db2')
SERVICE CLASS class1
```

CLP ファイル **batch2.db2** にワークロードを作成するには、次のような DDL ステートメントを発行できます。

```
CREATE WORKLOAD batch2 CURRENT CLIENT_APPLNAME ('CLP batch2.db2')
SERVICE CLASS class2
```

これら 2 つのバッチ・ファイルは、異なるワークロードに関連付けられるため、異なるサービス・クラスに割り当てて、異なる方法で管理できます。

もう 1 つの方法として、新規のストアード・プロシージャー WLM_SET_CLIENT_INFO を使用する方法があります。このプロシージャーは、簡単な CALL SQL ステートメントを使用して、サーバー側で任意のクライアント情報フィールドの値を設定できます。既存の任意の CLP スクリプトに CALL ステートメントを挿入することにより、これらのフィールドを使用してこれらスクリプトを一意的に識別でき、異なるワークロード定義にマップすることができます。

詳しくは、「*管理ルーチンおよびビュー*」の『WLM_SET_CLIENT_INFO プロシージャー』を参照してください。

COLLECT ACTIVITY DATA 節と比較して、COLLECT AGGREGATE ACTIVITY DATA 節を使用する必要があるのはどのような場合ですか。

この質問への回答は、モニターが必要となる理由と、その情報を使用して何をするかによって変わってきます。

集約アクティビティー情報は、節の有効範囲の中で実行した作業のセット全体に及ぶものであり、このセットのサマリー特性をキャプチャーするものです。個々のアクティビティーに関する特定の詳細をキャプチャーすることはしません。COLLECT AGGREGATE ACTIVITY DATA 節は、DB2 ワークロード、DB2 サービス・クラス、DB2 ワーク・アクション・セットに指定できます。通常の操作のモニターの場合、COLLECT AGGREGATE ACTIVITY DATA 節を使用してください。その理由は、これが非常に軽量で、履歴レコード用に統計イベント・モニターで自動的に収集でき、全体的な応答時間のパターンに関する重要な情報を提供するためです。特定の作業タイプに関するより深い理解が必要な場合は、DB2 ワーク・アクション・セットの中で COUNT ACTIVITY アクションまたは COLLECT AGGREGATE ACTIVITY DATA アクションを使用して、ワークロード、サービス・クラス、またはデータベースの中で実行しているさまざまなタイプの作業に関するより詳細な情報を (最小のオーバーヘッドで) 収集してください。

これとは対照的に、アクティビティー情報には、COLLECT ACTIVITY DATA 節の有効範囲の中で実行するすべてのアクティビティーそれぞれに関する詳細情報が含まれます。この節は、DB2 ワークロード、DB2 サービス・クラス、DB2 ワーク・アクション・セット、および DB2 しきい値に指定できます。これは、キャプチャーされる個々のアクティビティーについてのより深い分析を可能にします。例えば、新規アプリケーションがサブミットする SQL ステートメントの流れとタイプを理解したり、Explain 機能や設計アドバイザーなどのツールを使用してパフォーマンス調整の機会を調べたりする目的で使用できます。これは影響を受ける各アクティビティーについての情報をはるかに多くキャプチャーするため、この節を使用した場合の影響は他のモニター方法よりもこれらアクティビティーへの影響が大きいため、注意深く制御する必要があります。

DB2 ワークロード管理は新規の AIX WPAR (ワークロード・パーティション) フィーチャーとどのように連動しますか。

DB2 ワークロード管理のさまざまな側面はすべて、AIX WPAR の内部で作用しますが、AIX WPAR は AIX WLM フィーチャーの使用をサポートしていないため、DB2 サービス・クラスを AIX WLM サービス・クラスと緊密に統合するという選択肢はこの環境にとって有用ではありません。

DB2_OPT_MAX_TEMP_SIZE レジストリー変数と、SQLTEMPSPACE に基づく DB2 しきい値との関係はどのようなものですか。

これら 2 つに直接の関係はありません。 **DB2_OPT_MAX_TEMP_SIZE** レジストリー変数は、照会で使用できる TEMPORARY 表スペースの容量を制限するための照会コンパイラーに対するディレクティブです。これによってオプティマイザーは、コストはかかります (効率が悪い可能性はある) が、SYSTEM TEMPORARY 表スペースでは使用するスペースが少なくて済むプランを選択します。 SQLTEMPSPACE に基づく DB2 しきい値は、オプティマイザーが選択したプランのタイプには影響しません。このしきい値は、単に DB2 データ・サーバーに各メンバーでの SYSTEM TEMPORARY 表スペースの使用を照会によってモニターさせるもので、通常の処理中に所定の制限を超えると、しきい値違反を生成します。

Query Patroller が廃止され、DB2 ガバナーが非推奨になりましたが、DB2 ワークロード・マネージャーにどのようにマイグレーションすればよいですか。

DB2 バージョン 9.5 で戦略的ワークロード管理ソリューションとして DB2 ワークロード・マネージャーが導入された後、バージョン 10.1 のリリースで Query Patroller が廃止され、DB2 バージョン 9.7 リリース以降は DB2 ガバナーが非推奨になりました (将来のリリースでは除去される可能性があります)。

このリリースでは DB2 ガバナーが引き続きサポートされますが、このリリースで導入されているものも含め、DB2 ワークロード・マネージャーの新しいフィーチャーと機能を採用し始めてください。 DB2 ワークロード・マネージャーには、多数の追加オプションがあるので、それらを検討する必要があることに注意してください。DB2 データ・サーバー上の作業を制御するためのアプローチについて、現在のワークロード管理の見方で再考する必要が生じる可能性があります。DB2 ベスト・プラクティスの記事『Implementing DB2 workload management in a data warehouse』には、特に Query Patroller からマイグレーションするユーザー向けの補足情報が含まれています。関連タスクのセクションに記されている該当するタスク・トピックも使用できます。

DB2 Query Patroller から DB2 ワークロード・マネージャーへのマイグレーションを可能にするために、DB2 V9.7 フィックスパック 1 以降、サンプル・スクリプト (`qpwlmmig.pl`) が含まれています。追加情報については、Query Patroller から DB2 ワークロード・マネージャーへのマイグレーション方法を詳しく説明している以下のいずれかのタスクを参照してください。

- サンプル・スクリプトを使用した Query Patroller から DB2 ワークロード・マネージャーへのマイグレーション
- Query Patroller から DB2 ワークロード・マネージャーへのマイグレーション

DB2 ワークロード・マネージャーに関するどのようなライセンス交付要件がありますか。

DB2 データ・サーバーのワークロード管理機能のサブセットは、ライセンスによってその使用が制限されます。ライセンス交付を受けたこのサブセットは、DB2 ワークロード・マネージャーと呼ばれ、サービス・クラス、ワークロード、しきい値、またはワーク・アクション・セットの作成を制御します。ワークロード管理機能のこのサブセットにアクセスするには、以下のライセンス交付を受けた製品のいずれかが必要です。

- DB2 Enterprise Server Edition for Linux, UNIX, and Windows
- DB2 Advanced Enterprise Server Edition for Linux, UNIX, and Windows
- Database Enterprise Developer Edition for Linux, UNIX, and Windows
- IBM® InfoSphere® Warehouse (すべてのエディション)
- IBM Smart Analytics System

以下のワークロード管理機能は、ライセンスによる制限を受けません。

- デフォルトのサービス・クラスとワークロードの使用および変更。これには、すべてのモニター機能が含まれます。
- ヒストグラム・テンプレートの作成、変更、またはドロップ

- DB2 ワークロード管理の表関数またはストアード・プロシージャーの使用。
- ワークロード管理イベント・モニターの作成、活動化、停止、またはドロップ。
- ワークロード特権の付与、変更、または取り消し

ワークロード管理に関連した種々のイベント・モニターからはどのような情報を取得できますか。

しきい値違反、統計、およびアクティビティーに関する各イベント・モニターは、しきい値違反、運用統計、集約アクティビティー・データ、および個別のアクティビティー・データについての情報をそれぞれ収集します (270 ページの『WLM イベント・モニターを使用した履歴モニター』を参照してください)。

各イベント・モニターは 1 つ以上の論理データ・グループを収集し (「データベースのモニタリング *ガイドおよびリファレンス*」の『イベント・タイプの論理データ・グループへのマッピング』を参照)、それぞれの論理データ・グループには 1 つ以上のモニター・エレメントが入っています (「データベースのモニタリング *ガイドおよびリファレンス*」の『イベント・モニターの論理データ・グループおよびモニター・エレメント』を参照)。

例えば、しきい値違反イベント・モニターによって収集される情報を確認するには、『イベント・タイプの論理データ・グループへのマッピング』トピックにある 表 3 の中でまず探してください。この表では、event_thresholdviolations という単一の論理データ・グループにしきい値違反イベント・モニターが情報を収集することが示されています (アクティビティー・イベント・モニターのように、複数の論理データ・グループに情報を収集するイベント・モニターもあります)。次に、『イベント・モニターの論理データ・グループおよびモニター・エレメント』トピックで event_thresholdviolations 論理データ・グループを探します。このトピックには、event_thresholdviolations 論理データ・グループで報告されるモニター・エレメントが示されています。それには、以下が含まれています。

- **activate_timestamp** - タイム・スタンプのアクティブ化
- **activity_collected** - 収集されたアクティビティー
- **activity_id** - アクティビティー ID
- **agent_id** - アプリケーション・ハンドル (エージェント ID)
- **appl_id** - アプリケーション ID
- **coord_partition_num** - コーディネーター・パーティション番号
- **destination_service_class_id** - 宛先サービス・クラス ID
- **source_service_class_id** - ソース・サービス・クラス ID
- **threshold_action** - しきい値アクション
- **threshold_maxvalue** - しきい値最大値
- **threshold_predicate** - しきい値述部
- **threshold_queuesize** - しきい値キュー・サイズ
- **thresholdid** - しきい値 ID
- **time_of_violation** - 違反時刻
- **uow_id** - 作業単位 ID

この例で概要が示されている方法を使用すると、それぞれのイベント・モニターで収集されるデータを判別できます。

ワークロード管理しきい値によってキューに入れられるアクティビティーと、キューにおけるそれらのアクティビティーの順序はどのようにして判別できますか。

これを行うには、まず WLM_GET_SERVICE_CLASS_AGENTS 表関数を使ってビューを作成した後、キューに入ったアクティビティーをキューのエントリー時間順にリストするステートメントを実行できます。この実行方法については、268 ページの『例: WLM しきい値によってキューに入れられるアクティビティーと、そのキュー順序の判別』を参照してください。

ワークロード管理ディスパッチャーを有効にすると、どのように動作が変化する可能性がありますか。

wlm_dispatcher データベース・マネージャー構成パラメーターを介してワークロード管理ディスパッチャーを ON に設定すると、複数のサービス・クラスの間で作業の優先順位付けを行うためにこれまでエージェント優先順位を使用していた場合には、ワークロード管理ディスパッチャーが有効になっている間、このエージェント優先順位を使用できません。その結果、すべてのサービス・クラスはデフォルトのエージェント優先順位を持っているかのように扱われます。

wlm_disp_cpu_shares データベース・マネージャー構成パラメーターを介して CPU シェアを使用可能にした場合、サービス・クラスの CPU シェアや CPU リミットを指定しないと、すべてのサービス・クラスはシステム上の CPU リソースに関する等しいソフト共有を受け取ります。すべてのサービス・クラスが CPU リソースに関する等しいソフト共有を受け取る結果として、サービス・クラスへの CPU リソース割り振りが、以前の DB2 リリースとは異なるものになる可能性があります。そのため、ワークロードに適した CPU シェアまたは CPU リミット値を設定することを考慮する必要があります。 CPU シェアおよび CPU リミットの値を決定する方法について、詳しくは、190 ページの『ワークロード管理ディスパッチャー』を参照してください。

ワークロード管理ディスパッチャーの導入により、CONCURRENTDBCOORDACTIVITIES などの並行性しきい値は不要になりましたか。

DB2 ワークロード管理ディスパッチャーと並行性しきい値を一緒に使用することができます。実行される作業量を制御するうえで、並行性しきい値は引き続き非常に役立ちます。実行を開始する各アクティビティーに対して、DB2 データベース・マネージャーは CPU リソースに加えて他のリソースをそのアクティビティーに提供します (通常は、アクティビティーの実行期間にわたってそれらが保持されます)。このような CPU 以外のリソースには、DB2 エージェント、ソート・メモリー、TEMPORARY 表スペース、ロック、I/O などが含まれます。アクティビティーの実行開始を防ぐことで、これらの CPU 以外の追加リソースは消費されず、他のアクティビティーでこれらを使用できます。

さらに、実行中の作業の発生元を判別するために、DB2 データベース・マネージャー内のさまざまなポイントで並行性しきい値を適用できます。例えば特定の 1 つの

ワークロードから送られる大きな照会に対して並行性しきい値を設定すると、同じサービス・クラスに寄与する他のワークロードと比較した場合、サービス・クラスでその特定のワークロードに使用できるリソースの消費や共有が制限されます。

要約すると、アクティビティーがいつ実行を開始してシステム上の CPU リソースと CPU 以外のリソースを消費し始めるかを制御するために、並行性しきい値を使用できます。そのようなアクティビティーがいったん実行を開始した後、どれほどの CPU リソースを消費できるかを制御するには、ワークロード管理ディスパッチャーを使用できます。

AIX WLM または Linux WLM を使用すべき理由は何ですか。

DB2 ワークロードの CPU 消費を制御するために DB2 ワークロード管理ディスパッチャーを使用する場合でも、以下のような理由で、AIX WLM または Linux WLM を併用することができます。

- オペレーティング・システム (OS) のワークロード・マネージャーは、オペレーティング・システム・レベルでのリソース消費のモニター機能を提供します。
- OS のワークロード・マネージャーは、DB2 データベース・マネージャー・スレッドだけでなく、ホストまたは LPAR 全体にわたるすべてのプロセスやスレッドの制御を提供できます。これは、リソースに関して DB2 データベース・マネージャーと競合するプロセスを制御する必要がある場合に役立つことがあります。

第 2 章 作業識別

DB2 ワークロード管理ソリューションを正常にインプリメントする上で重要なことは、作業を識別することです。

作業を識別するには、ワークロード、ワーク・クラス、およびデータ・タグという 3 つの方法があります。

- ワークロードは作業のソースを識別する場合に使用できます。例えば、作業をサブミットしたアプリケーション名またはシステム許可 ID などの、主要なセッション属性を使用することにより、接続レベルまたはトランザクション・レベルでソースを識別することができます。
- ワーク・クラスを使用した場合、作業のある決まった特性を特定することにより、作業を識別できます。ワーク・クラスを定義して関心のある作業を識別できます。例えば、データ・サーバー上のデータを変更するのみのステートメント (例: INSERT、UPDATE、または DELETE ステートメント) などの識別が可能です。
- データ・タグは、表スペースおよびストレージ・グループのデータ・タグとして使用できます。これにより、実行時にアクセスされるデータのタイプにより、作業を間接的に識別できます。該当するデータをタグ付けすることにより、そのデータにアクセス中の作業を識別できます。

アクティビティー

ワークロードのモニターと制御を行う方法の 1 つに、個々のアクティビティー・ベースで行う方法があります。DB2 データ・サーバーが SQL または XQuery ステートメントに関するアクセス・プランを実行するか、またはロード・ユーティリティーを実行するたびに、対応するアクティビティーが作成されます。

ワークロード・モニターの場合、一般的に使用されるモニター・エレメントはアクティビティー単位に関する情報を提供します。例えば、アクティビティー実行時間モニター・エレメント (coord_act_exec_time) や並行アクティビティーの高水準点 (concurrent_act_top) などのモニター・エレメントから、ワークロードのボリュームや実行時間に関する情報を入手できます。

ワークロード制御の場合、ほとんどのワークロード制御としきい値は各アクティビティーに適用されます。例えば、ACTIVITYTOTALTIME しきい値は、データ・サーバーがアクティビティーの処理に費やす最大時間を制御します。

データ・サーバー上でアクティビティーを起動するステートメントまたはコマンド

以下のステートメントまたはコマンドは、データ・サーバー上でアクティビティーを起動します。

- すべての DML ステートメント
- すべての DDL ステートメント
- CALL ステートメント

- ロード・ユーティリティー

アクティビティーのライフ・サイクル

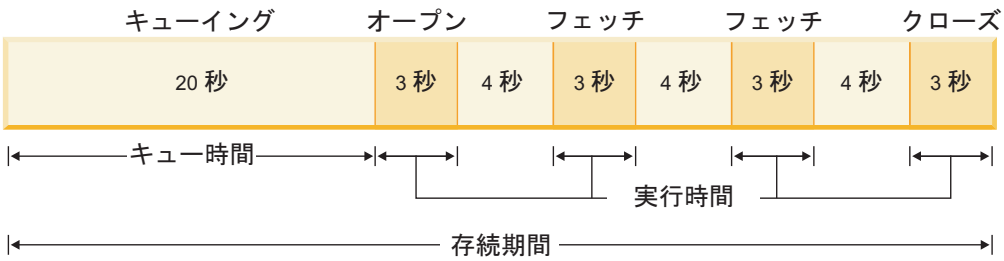
DML ステートメントに関するアクティビティーのライフ・サイクルには、アクセス・プランの実行前または実行の外部で発生する処理は含まれません。したがって、データベースへの接続やアクセス・プランへの SQL のコンパイルなどの操作は、アクティビティー・ベースのモニターの対象外になります。

アクティビティーはそのライフ・サイクル内で、さまざまな状態で時間を費やすことがあり、それらの状態は activity_state イベント・モニター・エレメントによって報告されます。アクティビティーがなる可能性のある状態の一部を以下に示します。

- EXECUTING - この状態は、コーディネーター・エージェントがアクティビティーに対して作業中であることを示します。ロック待機状態になるアクティビティーは、実行中として報告されます。
- IDLE - この状態は、コーディネーター・エージェントがクライアントからの次の要求を待機していることを示します。
- QUEUED - 一部のしきい値に組み込みキューが含まれています。この状態は、アクティビティーが実行開始の順番をキュー中で待機していることを示します。

アクティビティーに関するモニター・データは、アクティビティーの存続時間の終わりに集計されます。

以下の図は、長期実行照会の存続期間を、キュー時間と実行時間に分類する方法を示しています。



SQL ステートメントのタイプとアプリケーション開発

このセクションでは、さまざまな SQL ステートメントに対して作成されるアクティビティーについて説明し、これらのアクティビティーの存続時間の開始点と終点を識別します。この情報を使用すると、アクティビティーを通じて SQL ステートメントがモニターおよび制御される方法を理解できます。

SELECT ステートメント: SELECT ステートメントは 1 つのアクティビティーによって表されます。FETCH 操作や、副選択または副照会などのカーソル要求が含まれます。アクティビティーは、DB2 データ・サーバーがカーソルの OPEN ステートメントまたは要求の処理を始める際に開始し、データ・サーバーがカーソルの CLOSE ステートメントまたは要求の処理を完了する際に終了します。

WITH HOLD カーソルを使用する SELECT ステートメント: WITH HOLD カーソルを使用する場合、アプリケーションは、ある作業単位内でカーソルをオープン

し、そのカーソルを後続の作業単位内でクローズすることができます。カーソルは複数の作業単位でオープンしたままになります。対応するアクティビティーのライフ・サイクルが終了するのはカーソルのクローズ後に限られるので、このアクティビティーはカーソルがオープンしている限り存在します。

CALL ステートメントとストアード・プロシージャー: CALL ステートメント自体は 1 つのアクティビティーによって表されますが、ストアード・プロシージャーのペイロードはネストされたアクティビティーに以下のようにまたがることができます。

表 1. ストアード・プロシージャーの内容と、ストアード・プロシージャーが作成するアクティビティー

ストアード・プロシージャーの内容	作成される追加のアクティビティー
単一の SQL ステートメント	1 つ
ストアード・プロシージャーに SQL ステートメントがない	0
SQL プロシージャー、複数の SQL ステートメント、およびループ論理	複数のアクティビティー。各ステートメントの呼び出しごとに 1 つのアクティビティーが対応
別のストアード・プロシージャーに対する呼び出し	そのストアード・プロシージャーに関するアクティビティー

CALL ステートメントに関連付けられたアクティビティーは、DB2 データ・サーバーがステートメントまたは要求の処理を始める際に開始し、ストアード・プロシージャー処理の完了後に終了します。

トリガーおよび UDF: SQL ステートメントがトリガーまたは UDF を呼び出す際に、追加のアクティビティーは作成されません。トリガーまたは UDF によって行われる作業は、そのトリガーまたは UDF を呼び出した SQL ステートメントに関するアクティビティーのものになります。トリガーまたは UDF が追加の SQL ステートメントを実行する場合は、ステートメントの実行時と同様に処理されます。つまり、ステートメントごとにアクティビティーが作成されます。

PREPARE ステートメント: アクティビティーはアクセス・プランが実行されるまで作成されないので、アクティビティーは作成されません。

ネストされたアクティビティー

ネストされたアクティビティーは、ワークロードのアクティビティー・ベースのモニターや制御に大きな影響を及ぼすことはありませんが、いくつかの追加情報が適用されます。

自身の中にネストされたアクティビティーを持つことができるアクティビティーは、以下のとおりです。

- ストアード・プロシージャー
- 無名ブロック
- 自律型ルーチン
- UDF を実行する DML アクティビティー

- カーソルからのロード (内部にカーソル・アクティビティーがネストされているロード・アクティビティー)
- トリガー定義の一部として前の部分でリストされたいずれかのアクティビティーを含むトリガーの対象の DML アクティビティー

ネストされたアクティビティーは、モニター情報内で以下のように報告されます。

- ネストされたアクティビティーは、ゼロ以外の親 UOW ID とゼロ以外の親アクティビティー ID によって示されます。
- ネストされたアクティビティーは、ヒストグラムまたはヒストグラムから派生する統計に対してはカウントされません。
- ネストされたアクティビティーのデータは、親アクティビティーのメトリックの一部としては報告されません。例えば、CALL ステートメントによって実行されるプロシージャーが、プロセッサー時間 10 秒を費やす挿入を実行する場合、このプロセッサー時間は挿入アクティビティーに関するプロセッサー時間メトリックに対してカウントされるだけで、親の CALL アクティビティーに関するプロセッサー時間メトリックに対してはカウントされません。

ワークロード制御は、ネストされたアクティビティーを次のように考慮します。

- UDF またはトリガー内にネストされたアクティビティーは、CONCURRENTDBCOORDACTIVITIES しきい値を考慮する際には含まれません。
- ロード・アクティビティー内にネストされたカーソル・アクティビティーは、CONCURRENTDBCOORDACTIVITIES しきい値を考慮する際には含まれません。

アクティビティーとロード・ユーティリティー

ロード・ユーティリティーを実行すると、複数のアクティビティーが生成され、その 1 つはロード・アクティビティーになり、その他の複数のアクティビティーのタイプは READ、WRITE または OTHER になります。カーソルからのロードの場合は、ロード・アクティビティーのロード元のカーソルに関する追加アクティビティーが作成されます。このカーソル・アクティビティーは、ロード・アクティビティーの、ネストされたアクティビティーです。

アクティビティー・イベント・モニター

マルチメンバー・データベース環境でアクティビティー・イベント・モニターを使用してアクティビティーをモニターするときは、アクティビティー・イベント情報がどのようにキャプチャーされるかを知っておく必要があります。全メンバーでのアクティビティー・イベントを収集する場合、コーディネーター以外のメンバーにおいて、特定のアクティビティーの複数のイベントが表示されることがあります。場合によって 1 つのアクティビティーの複数のレコードが記録される理由は、セクション (SQL ステートメントの実行可能形式) 内のイベント順序付けに応じて、アクティビティーがリモート・メンバーとの間を往来することがあるためです。その結果、コーディネーター以外のメンバーでアクティビティーの複数のレコードが収集される可能性があります。リモート・メンバー上でアクティビティーによって実行される処理を理解するには、アクティビティーのすべてのレコードを考慮する必要があります。例えば、リモート・メンバー上のアクティビティーのすべてのレコードからメトリックを集約することができます。

対照的に、コーディネーター・メンバーで収集されるイベント・データでは、アクティビティーが一度だけ記録されます。

ワークロード管理 DDL ステートメント

ワークロード管理 DDL ステートメントは CREATE、ALTER、および DROP の各ステートメントから構成されます。これらは、サービス・クラス、ワークロード、ワーク・クラス・セット、ワーク・アクション・セット、しきい値、およびヒストグラムを操作するときに使用します。

DB2 ワークロード管理 DDL ステートメントを以下に示します。

- CREATE SERVICE CLASS、ALTER SERVICE CLASS、および DROP SERVICE CLASS
- CREATE WORKLOAD、ALTER WORKLOAD、および DROP WORKLOAD
- GRANT USAGE ON WORKLOAD および REVOKE USAGE ON WORKLOAD
- CREATE THRESHOLD、ALTER THRESHOLD、および DROP THRESHOLD
- CREATE WORK CLASS SET、ALTER WORK CLASS SET、および DROP WORK CLASS SET
- CREATE WORK ACTION SET、ALTER WORK ACTION SET、および DROP WORK ACTION SET
- CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、および DROP HISTOGRAM TEMPLATE

ワークロード管理 DDL ステートメントは、以下のように他の DB2 DDL ステートメントとは異なります。

- 全データベース・メンバーにおいて、非コミット・ワークロード管理 DDL ステートメントは、一度に 1 つしか許可されません。非コミット・ワークロード管理 DDL ステートメントが存在する場合、それ以降のワークロード管理 DDL ステートメントは非コミット・ワークロード管理 DDL ステートメントがコミットされるかロールバックされるまで待機します。ワークロード管理 DDL ステートメントは、発行された順序に処理されます。
- すべてのワークロード管理 DDL ステートメントの後は、COMMIT または ROLLBACK ステートメントでなければなりません。
- ワークロード管理 DDL ステートメントは、XA トランザクションでは発行できません。ある接続でワークロード管理 DDL ステートメントを発行した後、同じ接続でそのワークロード管理 DDL ステートメントの直後に COMMIT または ROLLBACK ステートメントを発行する必要があります。 XA トランザクションでは、複数の接続がトランザクションに参加することが可能で、いずれの接続もトランザクションをコミットまたはロールバックできます。この状況では、ワークロード管理環境を正常にインプリメントすることは不可能です。
- DB2 for z/OS® は、DB2 Database for Linux, UNIX, and Windows のワークロード管理 DDL ステートメントを認識しません。

ワークロードでの発生元による作業識別

ワークロードは、受け取った作業をそのソースに基づいて識別し、他のすべての作業から独立して、DB2 ワークロード管理で後からモニターまたは管理できるようにします。そのソースは、作業がサブミットされたデータベース接続の属性を使用して決定されます。

接続の属性は、接続が確立され、ワークロード定義と一致したときに最初に評価されます。接続と特定のワークロード定義との間のこの関係は、*ワークロード・オカレンス* と呼ばれます。属性のいずれかがその接続の存続期間中に変更される場合、そのワークロードの割り当ては変更後の次の作業単位が開始する時に再評価されます。新規のワークロード定義がその時点で接続により一致していることがわかった場合、(以前に割り当てられたワークロードの) 古いワークロード・オカレンスは終了し、新たに割り当てられたワークロード定義の新規のオカレンスが開始します。それぞれの接続は、一度に 1 つのワークロードだけに割り当てられますが、複数の接続 (ワークロード・オカレンス) が同じワークロード定義に同時に割り当てられる可能性もあります。詳しくは、27 ページの『ワークロードの割り当て』を参照してください。

例えば、アプリケーション `Accounts` によって作成されたすべての接続をワークロード REPORTING (それらの接続下のアクティビティーを `Marketing` サービス・クラスで実行するようマップする) に割り当てるには、次のように CREATE WORKLOAD ステートメントを発行します。

```
CREATE WORKLOAD REPORTING APPLNAME('Accounts') SERVICE CLASS Marketing
```

これによって次のワークロードが作成されます。

図 2. REPORTING ワークロード

次に、REPORTING ワークロードに対する USAGE 特権を PUBLIC に付与します。

```
GRANT USAGE ON WORKLOAD REPORTING TO PUBLIC
```

セッション・ユーザー・グループ `Deptmgr` に属する接続下でアプリケーション `Accounts` によって作成されたすべてのアクティビティーを SUMMARY ワークロード (アクティビティーを `HumanResources` サービス・クラスにマップする) に割り当てるには、次のようなステートメントを発行します。

```
CREATE WORKLOAD SUMMARY SESSION_USER_GROUP('Deptmgr') APPLNAME('Accounts')
SERVICE CLASS HumanResources
```

これによって次のワークロードが作成されます。

図 3. SUMMARY ワークロード

次に、SUMMARY ワークロードに対する USAGE 特権を PUBLIC に付与します。

```
GRANT USAGE ON WORKLOAD SUMMARY TO PUBLIC
```

SYSCAT.WORKLOADS ビューを照会することにより、ワークロード定義を表示できます。また、SYSCAT.WORKLOADCONNATTR ビューを照会することにより、各ワークロードに対して指定した接続属性を表示できます。SYSCAT.WORKLOADAUTH ビューを照会すると、ワークロードを使用する権限がある人物を表示できます。任意の時点でシステムに存在するワークロード・オカレンスを調べるには、WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数または MON_GET_WORKLOAD 表関数を使用します。

SYSDEFAULTUSERWORKLOAD はデフォルトのワークロードです。ワークロード評価時にカスタム定義ワークロードに割り当てられていない接続は、このデフォルトのワークロードに割り当てられます。これによって、すべてのデータベース接続がワークロードに関連付けられるようになります。デフォルトのワークロード SYSDEFAULTUSERWORKLOAD に割り当てられた作業は、デフォルトで SYSDEFAULTUSERCLASS サービス・クラスで実行されます。

サポートされるデータベース接続属性

ワークロード定義において少なくとも 1 つのデータベース接続属性を指定する必要があります。各接続属性は 1 つ以上の値を持つことができます。ワークロード定義内の特定の接続属性の値が指定されていない場合、データ・サーバーはワークロードの評価時にその属性を調べません。

表 2. ワークロード定義の接続属性

接続属性	説明
アドレス	クライアントがデータベース・サーバーと通信するために使用する実際の通信アドレス。サポートされるプロトコルは TCP/IP のみです。アドレスは、IPv4 アドレス、IPv6 アドレス、またはセキュア・ドメイン・ネームでなければなりません。
アプリケーション名	クライアントで実行中のアプリケーションの名前で、データ・サーバーに認識されます。アプリケーション名は、システム・モニター出力の「**アプリケーション名**」フィールドに表示される値と等しくなります。詳しくは、**appl_name** モニター・エレメントを参照してください。

表 2. ワークロード定義の接続属性 (続き)

接続属性	説明
システム許可 ID	SYSTEM_USER 特殊レジスターに設定されている、データベースに接続したユーザーの許可 ID。異なる許可 ID を持つユーザーとして接続することにより、SYSTEM_USER の値を変更できます。
セッション許可 ID	SESSION_USER 特殊レジスターに設定されている、アプリケーションの現行セッションで使用される許可 ID。トラステッド・コンテキストまたは SET SESSION AUTHORIZATION ステートメントを使用することにより、SESSION_USER の値を変更できます。
セッション許可 ID のグループ	現行セッション・ユーザーが属するグループ。
セッション許可 ID のロール	現行セッション・ユーザーに付与されるロール。
クライアント・ユーザー ID	CURRENT CLIENT_USERID (または CLIENT USERID) 特殊レジスターで設定されている、クライアント情報からのクライアント・ユーザー ID。特定の DB2 クライアント、sqleseti (クライアント情報の設定) API、または WLM_SET_CLIENT_INFO プロシージャーによって提供される定義済みメカニズムのいずれかを使用することにより、クライアント・ユーザー ID の値を変更できます。
クライアント・アプリケーション名	CURRENT CLIENT_APPLNAME (または CLIENT APPLNAME) 特殊レジスターで設定されている、クライアント情報からのアプリケーション名。特定の DB2 クライアント、sqleseti API、または WLM_SET_CLIENT_INFO プロシージャーによって提供される定義済みメカニズムのいずれかを使用することにより、クライアント・アプリケーション名の値を変更できます。
クライアント・ワークステーション名	CURRENT CLIENT_WRKSTNNAME (または CLIENT WRKSTNNAME) 特殊レジスターで設定されている、クライアント情報からのワークステーション名。特定の DB2 クライアント、sqleseti API、または WLM_SET_CLIENT_INFO プロシージャーによって提供される定義済みメカニズムのいずれかを使用することにより、クライアント・ワークステーション名の値を変更できます。

表 2. ワークロード定義の接続属性 (続き)

接続属性	説明
クライアント・アカウンティング・ストリング	CURRENT CLIENT_ACCTNG (または CLIENT ACCTNG) 特殊レジスターで設定されている、クライアント情報からの会計情報ストリング。 sqleseti API または WLM_SET_CLIENT_INFO プロシージャーを使用することにより、クライアント・アカウンティング・ストリングの値を変更できます。

接続属性でのワイルドカードの使用

接続属性の中には、CREATE WORKLOAD および ALTER WORKLOAD ステートメントにアスタリスク (*) をワイルドカードとして指定することをサポートしているものもあります。ワイルドカードを使用できるのは、接続属性がいくつかの類似した値を持てる場合です。ワイルドカードを使用した正規表現でそれらの値をマッチングさせることができ、考えられる値ごとに接続属性を定義する必要はありません。

ワイルドカードのアスタリスク (*) はゼロ個以上の文字とマッチングします。アスタリスクとマッチングさせる必要がある場合は、二重アスタリスク (**) を使用してアスタリスクをリテラル文字として指定します。

例: いくつもの売掛金アプリケーション (*accrec01*、*accrec02 ... accrec15*) を DB2 ワークロード・マネージャーで同等に扱われるようにすべて同じワークロードに属するようにする場合は、*CURRENT CLIENT_APPLNAME('accrec*')* 接続属性を定義することにより、ワークロードを作成または変更するときにこれらのアプリケーションのすべてとマッチングするようにします。同様に、*acc*rec* 売掛金アプリケーション (アスタリスク文字を含んだ名前) は、*CURRENT CLIENT_APPLNAME('acc**rec')* 接続属性がマッチングします。

以下のワークロード接続属性がワイルドカードの使用をサポートしています。

- APPLNAME
- CURRENT CLIENT_ACCTNG
- CURRENT CLIENT_APPLNAME
- CURRENT CLIENT_USERID
- CURRENT CLIENT_WRKSTNNAME

要求を識別するためのクライアント情報の設定

デフォルトで、多くのアプリケーション・サーバーは、処理対象のすべてのクライアント要求に対し、同じ情報を使って接続をセットアップし、同じクライアント情報を渡します (存在する場合)。WebSphere および Cognos® などの一部の製品は、クライアント情報フィールドを介して各要求に関する固有の情報をプッシュダウンする機能を提供しています。この情報は、DB2 内でエンド・ユーザー要求を一意的に識別します。他のほとんどの製品は、エンド・ユーザー要求の処理を開始する前

に固有のクライアント情報を DB2 に送信できるように、アプリケーション・サーバーをカスタマイズするための方法を提供しています。

固有のクライアント属性をアプリケーション・サーバーから指定することにより、DB2 内の要求の特殊な扱いや、さまざまなクライアントからの要求をさまざまなワークロード (および、さまざまなサービス・クラス) に割り当てることが可能になります。

接続属性の評価順序

環境の使用特性を分析する際に、CREATE WORKLOAD ステートメントを使用して独自のワークロードを作成し、それらを特定のサービス・クラスにマップすることができます。ワークロードを作成する時には、ワークロードの割り当て中に接続属性を評価するために使用する値、およびワークロードを評価する順番 (他のワークロードに対する相対的な順番) の両方を定義します。複数のワークロードが接続属性と一致する場合があるため、評価順序が変更可能であることにより、一致するどのワークロードを選択するかを決めることが可能になります。セッション・ユーザーがワークロードに対する USAGE 特権があるかどうかによっても、一致するどのワークロードを選択するかが決まります。詳しくは、27 ページの『ワークロードの割り当て』を参照してください。

以下の図は、複数の要求を A、B、C、D の順で各ワークロードに照らして評価し、次いで特定のワークロードに割り当てて、当てはまるサービス・クラスで実行する様子を示します。既存のワークロードと一致しない要求は、SYSDEFAULTUSERWORKLOAD ワークロードとマッチングされます。デフォルトの保守クラスおよびデフォルトのシステム・クラスで実行するアクティビティーのタイプについては、81 ページの『デフォルトのサービス・スーパークラスおよびサブクラス』を参照してください。

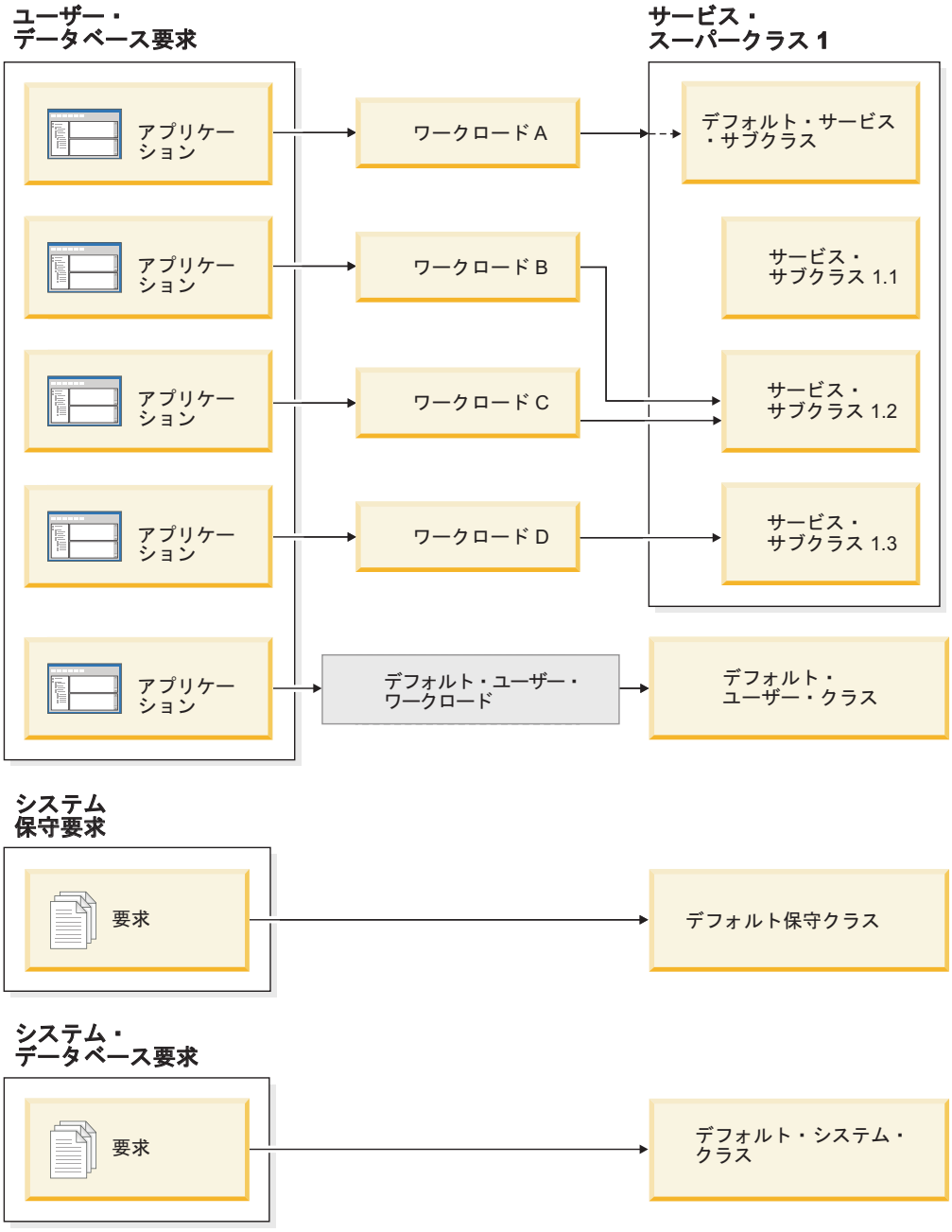


図 4. サービス・クラスとワークロード

ワークロードの割り当て

データベース接続が確立された後の最初の作業単位の初めに、使用可能になっている各ワークロードの接続属性を評価することによって、データ・サーバーは接続をワークロードに割り当てます。

ワークロードが評価される順序は、SYSCAT.WORKLOADS 表にある各ワークロードの EVALUATIONORDER 列値によって決定されます。一致する接続属性を持つワークロードが見つかると、データ・サーバーは、現行のセッション・ユーザーに

そのワークロードに対する USAGE 特権があるかどうかを確認します。一致するワークロードに対する USAGE 特権をユーザーが持っている場合、ワークロードの割り当ては完了し、接続はそのワークロードに割り当てられます。一致するワークロードに対する USAGE 特権をユーザーが持っていない場合、データ・サーバーは、セッション・ユーザーが USAGE 特権を持つ、一致するワークロードが見つかるまで、ワークロードの評価を続行します。一致するワークロードが見つからない場合、データ・サーバーは SYSDEFAULTUSERWORKLOAD ワークロードを使用しようとします。現行セッション・ユーザーにそのワークロードに対する USAGE 特権がない場合、SQL4707N が戻され、作業単位は拒否されます。それ以外の場合、接続は SYSDEFAULTUSERWORKLOAD ワークロードに割り当てられます。

CREATE WORKLOAD または ALTER WORKLOAD ステートメントの POSITION キーワードを使用して、評価順序を次のように設定することができます。

- 評価順序内でのワークロードの絶対位置を指定します。次に例を示します。

  ```
  CREATE WORKLOAD...POSITION AT 2
  ```

  POSITION AT 2 は、ワークロードが評価順序の 2 番目に置かれることを意味します。一致するワークロードのうち、評価順序の位置がより高いものが最初に評価されます。つまり、位置 2 と位置 3 にあるワークロードが一致する場合、位置 2 にあるワークロードが位置 3 にあるワークロードの前に評価されます。

  CREATE WORKLOAD または ALTER WORKLOAD ステートメントで指定する位置が既存のワークロードの合計数より大きい場合、ワークロードは、評価順序の最後から 2 番目 (SYSDEFAULTUSERWORKLOAD ワークロードの前) に置かれます。これは、CREATE WORKLOAD または ALTER WORKLOAD ステートメントで POSITION LAST を指定した場合と効果は同じです。
- POSITION BEFORE *workload-name* または POSITION AFTER *workload-name* キーワードを使用します。*workload-name* は既存のワークロードです。このキーワードは、評価順序における新規または変更されたワークロードの位置を、別のワークロードから見た相対的な位置で指定します。次に例を示します。

  ```
  ALTER WORKLOAD...POSITION BEFORE workload2
  ```

POSITION キーワードを指定しない場合は、デフォルトで、新規ワークロードは、評価順序内の他の定義済みのワークロードの後、および常に最後であると見なされる SYSDEFAULTUSERWORKLOAD ワークロードの前に置かれます。

ワークロードの再割り当て

接続属性の 1 つ、またはワークロード定義のセットが変更されると、接続の割り当て先のワークロードが作業単位の境界ごとに変わる可能性があります。作業単位の境界とは、接続とその現行トランザクションとの関連付けが解除されるポイントのことです。作業単位の境界の原因となるイベントには、コミット、ロールバック、XA 終了 (成功)、XA コミット、および XA ロールバックがあります。

以下のイベントのいずれかが発生したことをデータ・サーバーが検出すると、新しい作業単位の初めにワークロード割り当てが再評価されます。

- 関連した接続属性が変更されている。ワークロード定義で指定可能な接続属性のリストについて詳しくは、22 ページの『ワークロードでの発生元による作業識別』の表を参照してください。ワークロードの再評価は、現行セッション許可 ID

が変更されたときにも行われます。これは、トラステッド・コンテキストが原因でデータベース接続が切り替えられるためです。詳しくは、トラステッド・コンテキストおよびトラステッド接続を参照してください。

- ワークロードを作成または変更した。
- ユーザー、グループ、またはロールにワークロードに対する USAGE 特権を付与したか、あるいはワークロードに対する USAGE 特権をユーザー、グループ、またはロールから取り消した。

作業単位の境界をまたぐアクティビティーがまだアクティブである間は、別のワークロードに接続を再割り当てできません。アクティビティーになるのは、ロード操作、ストアード・プロシージャーまたは表関数、あるいは WITH HOLD カーソルなどの、複数の UOW 間でリソースを維持する操作です。現在のワークロード・オカレンスは、すべてのアクティビティーが完了するまで実行されます。ワークロードの再割り当ては、次の作業単位の開始時に実行されます。

以下のいずれかの場合、ワークロードの割り当てまたは再割り当てが試行されると、SQL4707N エラーが生じます。

- データ・サーバーが、データベースへのアクセスが許可されていないワークロードに対して接続を割り当てようとした場合。詳しくは、 42 ページの『ワークロード・オカレンスがデータベースにアクセスしないようにする』を参照してください。
- データ・サーバーが SYSDEFAULTUSERWORKLOAD ワークロードに接続を割り当てようとするものの、現行セッション・ユーザーがこのワークロードに対する USAGE 特権を持っていない場合。

ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM 権限がある場合、データベース接続をデフォルトの管理者ワークロードである SYSDEFAULTADMWORKLOAD ワークロードに割り当てることができます。詳しくは、 32 ページの『デフォルトの管理ワークロードを使用した修正アクションの実行』を参照してください。

XA トランザクションおよびワークロード再割り当て

XA_END (成功)、XA コミット、および XA ロールバックなどの XA 呼び出しは、作業単位の終わりを示す DB2 COMMIT または ROLLBACK を発行します。ワークロード再評価は作業単位の初めに行うことができるため、これらの XA 呼び出しによってワークロード再評価が開始することがありますが、ワークロード再評価の理由は XA トランザクション自体とは直接関連していません。

デフォルトのワークロード

デフォルトのユーザー・ワークロード SYSDEFAULTUSERWORKLOAD は、すべての接続の初期割り当て先にするデータ・サーバーのワークロードを提供します。デフォルトの管理ワークロード SYSDEFAULTADMWORKLOAD は、これ以外では行えない修正管理アクションを可能にします。これらのワークロードは両方ともデータベース作成時に作成され、ドロップすることはできません。

デフォルトのユーザー・ワークロード (SYSDEFAULTUSERWORKLOAD)

デフォルトのユーザー・ワークロードに割り当てられる接続は、デフォルトのユーザー・サービス・スーパークラス SYSDEFAULTUSERCLASS にマップされます。このスーパークラスはデフォルトの実行環境を提供します。ユーザー定義ワークロードを作成すると、接続をユーザー定義サービス・クラスにマップできます。さらに、SYSDEFAULTUSERWORKLOAD を変更して、SYSDEFAULTUSERCLASS とは異なるサービス・クラスに接続をマップすることもできます。

SYSDEFAULTUSERWORKLOAD ワークロードは、SYSCAT.WORKLOADS 表を照会することによって表示できます。

以下の表では、SYSCAT.WORKLOADS ビューに返される、SYSDEFAULTUSERWORKLOAD ワークロードに関する列とその値、およびそれらの値を変更できるかどうかを示します。 SYSDEFAULTUSERWORKLOAD ワークロードへの接続の割り当て方法については、27 ページの『ワークロードの割り当て』を参照してください。

表 3. SYSCAT.WORKLOADS 内の SYSDEFAULTUSERWORKLOAD 項目

列	値	DBADM または WLMADM 権限 (および COLLECT 節のための SQLADM) がある場合に ALTER WORKLOAD ステートメントを使用して変更可能かどうか
WORKLOADID	1	不可
WORKLOADNAME	SYSDEFAULTUSERWORKLOAD	不可
EVALUATIONORDER	最後から 2 番目	不可
CREATE_TIME	データベース作成のタイム・スタンプ	不可
ALTER_TIME	最後に行われたワークロード定義の更新のタイム・スタンプ	不可 (ただし、ワークロード定義を更新したときにデータ・サーバーはこの列を変更します)
ENABLED	Y	不可
ALLOWACCESS	Y	可
SERVICECLASSNAME	SYSDEFAULTSUBCLASS	可
PARENTSERVICECLASSNAME	SYSDEFAULTUSERCLASS	可
COLLECTAGGACTDATA	N	可
COLLECTACTDATA	N	可
COLLECTACTPARTITION	C	可
COLLECTDEADLOCK	W	可
COLLECTLOCKTIMEOUT	W	可
COLLECTLOCKWAIT	N	可
LOCKWAITVALUE	0	可
COLLECTACTMETRICS	N	可
COLLECTUOWDATA	N	可
EXTERNALNAME	NULL	不可

表 3. SYSCAT.WORKLOADS 内の SYSDEFAULTUSERWORKLOAD 項目 (続き)

列	値	DBADM または WLMADM 権限 (および COLLECT 節のための SQLADM) がある場合に ALTER WORKLOAD ステートメントを使用して変更可能かどうか
REMARKS	BLANK	可

詳しくは、SYSCAT.WORKLOADS を参照してください。

デフォルトの管理ワークロード (SYSDEFAULTADMWORKLOAD)

ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM ユーザーは、いつでもこのワークロードを使用して、データベースを照会したり、管理タスクやモニター・タスクを実行したりできます。しかし、通常は以下の状況でこのワークロードが使用されます。

- 管理者が割り当てられているワークロードが、データベースへのアクセスを許可されていない (つまり、ワークロードに対して CREATE WORKLOAD または ALTER WORKLOAD ステートメントの DISALLOW DB ACCESS キーワードが指定されている)。
- しきい値に違反したために管理者がデータベースでの作業を実行できない。

SYSDEFAULTADMWORKLOAD ワークロードは、以下の点で他のワークロードとは異なります。

- ドロップしたり、使用不可にしたりすることはできません。
- DISALLOW DB ACCESS を指定することはできません。
- このワークロードのオカレンスおよびその中のアクティビティーに適用されるしきい値はありません。
- このワークロードは、SYSDEFAULTUSERCLASS サービス・スーパークラスでのみ実行できます。詳しくは、81 ページの『デフォルトのサービス・スーパークラスおよびサブクラス』を参照してください。
- 接続をこのワークロードに割り当てることができます。そのためには、コマンド行プロセッサー (CLP) から **SET WORKLOAD** コマンドを使用するか、WLM_SET_CLIENT_INFO ストアード・プロシージャーを呼び出します (そして **client_workload** パラメーターに SYSDEFAULTADMWORKLOAD を指定します)。詳しくは、32 ページの『デフォルトの管理ワークロードを使用した修正アクションの実行』を参照してください。

SYSDEFAULTADMWORKLOAD ワークロードは、SYSCAT.WORKLOADS 表を照会することによって表示できます。以下の表では、SYSCAT.WORKLOADS カタログ・ビューに返される、SYSDEFAULTADMWORKLOAD ワークロードに関する列とその値、およびそれらの値を変更できるかどうかを示します。

表 4. SYSCAT.WORKLOADS 内の SYSDEFAULTADMWORKLOAD 項目

列	値	DBADM または WLMADM 権限 (および COLLECT 節のための SQLADM) がある場合に ALTER WORKLOAD ステートメントを使用して変更可能かどうか
WORKLOADID	2	不可
WORKLOADNAME	SYSDEFAULTADMWORKLOAD	不可
EVALUATIONORDER	最後	不可
CREATE_TIME	データベース作成のタイム・スタンプ	不可
ALTER_TIME	最後に行われたワークロード定義の更新のタイム・スタンプ	不可 (ただし、ワークロード定義を更新したときにデータ・サーバーはこの列を変更します)
ENABLED	Y	不可
ALLOWACCESS	Y	不可
SERVICECLASSNAME	SYSDEFAULTSUBCLASS	不可
PARENTSERVICECLASSNAME	SYSDEFAULTUSERCLASS	不可
COLLECTAGGACTDATA	N	可
COLLECTACTDATA	N	可
COLLECTACTPARTITION	C	可
COLLECTDEADLOCK	W	可
COLLECTLOCKTIMEOUT	W	可
COLLECTLOCKWAIT	N	可
LOCKWAITVALUE	0	可
COLLECTACTMETRICS	N	可
COLLECTUOWDATA	N	可
EXTERNALNAME	NULL	不可
REMARKS	BLANK	可

詳しくは、SYSCAT.WORKLOADS を参照してください。

デフォルトの管理ワークロードを使用した修正アクションの実行

デフォルトの管理ワークロード SYSDEFAULTADMWORKLOAD は、DB2 が供給する特殊なワークロード定義で、どの DB2 のしきい値の影響も受けません。このワークロードを使用して、他の方法では実行できない修正アクションを実行します。例えば、ワークロード内のすべてのアクティビティーが実行されないようにする禁止しきい値定義の変更などです。

始める前に

SET WORKLOAD (または WLM_SET_CLIENT_INFO プロシージャー) コマンドを使用して、デフォルト管理ワークロード SYSDEFAULTADMWORKLOAD に接続を割り当てます。

SET WORKLOAD コマンドの使用には特別な権限は必要ありませんが、デフォルト管理ワークロードに接続を割り当てるためには

ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM 権限が必要です。権限がない場合は、ワークロードの割り当て時に SQL0552N が戻されます。

このタスクについて

このワークロードはしきい値の影響を受けないため、ワークロード管理制御が制限されており、定期的な日常の業務に使用することはお勧めできません。

手順

デフォルト管理ワークロードに接続を割り当てるには、次のように **SET WORKLOAD** コマンドを発行します。

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD
```

コマンドがいつ有効になるかは、いつコマンドを発行したかによって異なります。

- データベースへ接続する前に SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドを発行した場合は、接続が確立された後、最初の作業単位が開始されるときに、接続が SYSDEFAULTADMWORKLOAD に割り当てられます。
- 作業単位の開始時に SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドを発行した場合は、データベースへの接続が確立された後、sqleseti (クライアント情報設定) 要求以外の最初の要求がサブミットされるときに、接続が SYSDEFAULTADMWORKLOAD に割り当てられます。
- 作業単位の中間で SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドが発行された場合は、接続が確立された後、次の作業単位の開始時に、接続が SYSDEFAULTADMWORKLOAD に割り当てられます。

接続が SYSDEFAULTADMWORKLOAD に関連付けられていると、次のいずれかの状況が発生した場合に、次の作業単位の開始時にワークロードの再割り当てが行われます。

- セッション・ユーザーから SYSADM または DBADM 権限を取り消した。この場合は、SQL0552N が戻されます。
- SET WORKLOAD TO AUTOMATIC コマンドを発行した。このコマンドは、次の作業単位を SYSDEFAULTADMWORKLOAD ワークロードに割り当てず、次の作業単位の開始時には通常のワークロード評価を行うことを示します。詳しくは、27 ページの『ワークロードの割り当て』を参照してください。

例

次の例は、SYSDEFAULTADMWORKLOAD ワークロードを使用して、その他の修正アクションが不可能な場合の修正アクションの実行方法を示しています。

どのアクティビティーも実行できないほどに極めて禁止度の高い並行性しきい値を作成してしまった場合は (しきい値を常に超えてしまう)、そのしきい値が問題の修正の妨げになることがあります。設定したしきい値を変更するためには、まず、デフォルトの管理ワークロードで作業を実行するようにワークロードを設定する必要があります。このワークロードで実行されているアクティビティーはしきい値の制御下にないので、問題を修正してから、(ご使用の ID の) ワークロードを再びデフォルト動作に戻すことができます。

問題の原因になっているしきい値は、次のステートメントで誤って作成されています。並行性を 100 に設定すべきでしたが、0 に設定されています。このしきい値のために、事実上どのアクティビティーも実行を妨げられています。

```
CREATE THRESHOLD PROHIBITIVE FOR DATABASE ACTIVITIES
   ENFORCEMENT DATABASE WHEN CONCURRENTDBCOORDACTIVITIES > 0
   STOP EXECUTION
```

注: このステートメントは、どのような場合に極めて禁止度の高いしきい値が作成されてしまうかを例示することのみを目的としています。このステートメントを発行しないでください。

次のようなまったく単純な SELECT ステートメントでさえ、実行しようとするとエラーが返されます。並行性が 0 に設定されているためです。

```
SELECT * FROM SYSCAT.TABLES

SQL4712N  しきい値 "PROHIBITIVE" を超えました。  理由コード = "6"。
SQLSTATE=5U026
```

修正アクションを実行するには、次のようにワークロードをデフォルトの管理ワークロードに設定する必要があります。

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD
```

このステートメントは ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM 権限を持つユーザーのみが発行でき、どの接続も、アクティビティーが禁止しきい値に制約されない SYSDEFAULTADMWORKLOAD ワークロードに割り当てられるようになります。

これで、アクティビティーが実行されるよう次のようにしきい値を変更することによって、問題を修正できるようになりました。

```
ALTER THRESHOLD PROHIBITIVE WHEN CONCURRENTDBCOORDACTIVITIES > 100 STOP EXECUTION
```

修正したら、接続が SYSDEFAULTADMWORKLOAD ではなく前の割り当て先のワークロードに割り当てられるように、次のようにワークロードを元の設定に変更します。

```
SET WORKLOAD TO AUTOMATIC
```

これまでと同じ SELECT ステートメントが、今度は次のように正常に完了します。

```
SELECT * FROM SYSCAT.TABLES

...

DB20000I  SQL コマンドが正常に完了しました。
```

ワークロードの作成

CREATE WORKLOAD ステートメントを使用して、カタログにワークロードを追加します。

始める前に

ワークロードを作成するためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』
- 559 ページの『付録 A. 一般的な命名規則』

手順

ワークロードを作成するには、次のようにします。

1. CREATE WORKLOAD ステートメントを使用して、以下に挙げるワークロードのプロパティーを 1 つ以上指定します。
   - ワークロードの名前。
   - 接続属性。一致が生じるためには、ワークロードに指定した接続属性と一致するものを着信接続が提供しなければなりません。詳しくは、 22 ページの『ワークロードでの発生元による作業識別』を参照してください。接続属性を指定する際は、値が OR 演算され、属性が AND 演算されることに注意してください。例えば、UserID (bob OR sue OR frank) AND Application (SAS) となります。
   - このワークロードのオカレンスがデータベースへのアクセスを許可されるかどうかを示す値。デフォルトでは、このワークロードのオカレンスはデータベースへのアクセスを許可されます。
   - ワークロードが使用可能か使用不可かを示す値。デフォルトでは、ワークロードは使用可能になっています。
   - このワークロードのオカレンスによってサブミットされた作業が実行されるサービス・クラス。デフォルトは SYSDEFAULTUSERCLASS サービス・スーパークラスです。ユーザー定義のサービス・スーパークラスを指定する場合、ワークロード・オカレンスによってサブミットされた作業は、そのサービス・スーパークラスの SYSDEFAULTSUBCLASS サービス・サブクラスで実行されます。

     注: SYSDEFAULTUSERCLASS サービス・スーパークラスを含め、サービス・スーパークラスの下に SYSDEFAULTSUBCLASS サービス・サブクラスを指定することはできません。
     サービス・スーパークラスを指すようにワークロードを定義済みであるものの、それによってサブミットされた作業をデフォルトの SYSDEFAULTSUBCLASS サービス・サブクラスで実行させたくない場合は、ユーザー定義のサービス・サブクラスを直接指すようにワークロード定義を変更するか、あるいはサービス・スーパークラスで定義されたワーク・アクション・セットを使用して個別の作業を別のサービス・サブクラスにマップすることができます。詳しくは、 111 ページの『ワーク・アクションとワーク・アクション・セット』を参照してください。
   - メモリーにキャッシュされる際の、他のワークロードに対するワークロードの相対的な位置。新しいワークロードの位置は、ワークロード割り当て時にワークロードが評価される順序を決定します。デフォルトでは、新しいワークロードは一番後ろに配置されます。これは、そのワークロードが最後 (デフォルトのユーザー・ワークロードが考慮される直前) に評価されることを意味します。詳しくは、 27 ページの『ワークロードの割り当て』を参照してください。
   - このワークロードに関連付けられた接続によってサブミットされたアクティビティーに関する、モニター・アクティビティーのメトリック収集レベル。ワー

クロードに対するデフォルトのアクティビティー・メトリック収集設定は、NONE です。アクティビティーに対する効率的なアクティビティー収集設定値は、ワークロード・アクティビティー・メトリック収集レベルと **mon_act_metrics** データベース構成パラメーターの両方を組み合わせたものであることに注意してください。

- 収集するアクティビティー情報のタイプ。デフォルトでは、ワークロードに関連するアクティビティーの情報は、まったくアクティビティー・イベント・モニターに送信されません。
- 収集する集約アクティビティー情報。ワークロードに使用される集約アクティビティー情報は、CREATE WORKLOAD 操作がコミットされた後にのみ変更されます。
- 収集するロック・タイムアウト・イベント情報。デフォルトでは、ロック・イベントが発生したときにロック・イベントに関するデータがロッキング・イベント・モニター (アクティブな場合) に送信されますが、前のロック・タイムアウト・イベントは送信されません (WITHOUT HISTORY)。
- 収集するデッドロック情報。デフォルトでは、デッドロック・イベントが発生したときにデッドロック・イベントに関するデータがロッキング・イベント・モニター (アクティブな場合) に送信されますが、前のデッドロック・イベントは送信されません (WITHOUT HISTORY)。
- 収集するロック待機イベント情報。デフォルトでは、設定された待機時間内にロックが獲得されないと、ロック待機情報は収集されません。
- 作業単位の終了時に、作業単位イベント・モニター (アクティブな場合) に送信される、このワークロードに関連したトランザクションごとの作業単位の情報。デフォルトでは、作業単位の情報は送信されません。
- ワークロードがそのヒストグラムのテンプレートとして使用するヒストグラム・テンプレート。指定されるヒストグラム・テンプレートは、SYSCAT.HISTOGRAMTEMPLATEUSE ビューに反映されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、292 ページの『ワークロード管理のヒストグラム』を参照してください。

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューに追加されます。変更をコミットすると、各アプリケーションの次の作業単位の開始時にワークロードの再評価が行われるようになります。どのワークロードを選択するかによっては、アプリケーションが別のワークロードに再割り当てされることがあります。

次のタスク

ワークロードを作成した後、1 人以上のセッション・ユーザーにそのワークロードに対する USAGE 特権を付与することが必要になる場合もあります。(WLMADM または DBADM 権限を持つセッション・ユーザーには、すべてのワークロードを使用する暗黙特権があります。) 接続の接続属性がワークロードの接続属性と完全に一致する場合でも、セッション・ユーザーがそのワークロードに対する USAGE 特権を持っていない場合は、データ・サーバーは、ワークロードの評価を実行する際に、そのワークロードを考慮しません。詳しくは、42 ページの『ワークロードに対する USAGE 特権の付与』を参照してください。

ワークロードの変更

ALTER WORKLOAD ステートメントは、カタログ内のワークロードを変更します。

始める前に

ワークロードを変更するためには、SQLADM、WLMADM、または DBADM 権限が必要です。COLLECT 節以外の節を指定するには、許可 ID に WLMADM または DBADM 権限が組み込まれている必要があります。

前提条件について詳しくは、21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

手順

ワークロードを変更するには、次のようにします。

1. ALTER WORKLOAD ステートメントを使用して、以下に挙げるワークロードのプロパティーを 1 つ以上指定します。
   - 接続属性。ワークロードが SYSDEFAULTUSERWORKLOAD または SYSDEFAULTADMWORKLOAD ワークロードである場合以外は、ワークロード定義に接続属性の追加やドロップができます。一致が生じるためには、ワークロードに指定した接続属性と一致するものを着信接続が提供しなければなりません。詳しくは、22 ページの『ワークロードでの発生元による作業識別』を参照してください。ワークロードの接続属性を確認するには、SYSCAT.WORKLOADCONNATTR ビューを照会します。
   - このワークロードのオカレンスがデータベースへのアクセスを許可されるかどうかを示す値。デフォルトでは、このワークロードのオカレンスはデータベースへのアクセスを許可されます。 SYSDEFAULTADMWORKLOAD ワークロードからデータベースへのアクセスを除去することはできません。
   - ワークロードが使用可能か使用不可かを示す値。デフォルトでは、ワークロードは使用可能になっています。SYSDEFAULTUSERWORKLOAD ワークロードや SYSDEFAULTADMWORKLOAD ワークロードは使用不可にできません。
   - このワークロードのオカレンスが実行されるサービス・クラス。デフォルトは SYSDEFAULTUSERCLASS サービス・スーパークラスです。ユーザー定義のサービス・スーパークラスを指定する場合は、そのサービス・スーパークラスの下にサービス・サブクラスを指定することができます。SYSDEFAULTUSERCLASS サービス・スーパークラスを含め、サービス・スーパークラスの下に SYSDEFAULTSUBCLASS サブクラスを指定することはできません。加えて、SYSDEFAULTSYSTEMCLASS または SYSDEFAULTMAINTENANCECLASS サービス・スーパークラスは指定できません。
   - ワークロード割り当て時にワークロードが評価される順序を決定する、他のワークロードに対するワークロードの相対的な位置。SYSDEFAULTUSERWORKLOAD または SYSDEFAULTADMWORKLOAD ワークロードの位置は指定できません。詳しくは、27 ページの『ワークロードの割り当て』を参照してください。

- 収集するアクティビティー情報のタイプ。デフォルトでは、ワークロードに関連するアクティビティーの情報は、まったくアクティビティー・イベント・モニターに送信されません。
- このワークロードに関連付けられた接続によってサブミットされたアクティビティーに関する、モニター・アクティビティーのメトリック収集レベル。アクティビティーに対する効率的なアクティビティー収集設定値は、ワークロード・アクティビティー・メトリック収集レベルと **mon_act_metrics** データベース構成パラメーターの両方を組み合わせたものであることに注意してください。
- 収集する集約アクティビティー情報。ワークロードに使用される集約アクティビティー情報は、ALTER WORKLOAD 操作がコミットされた後にのみ変更されます。
- ロック・イベントの発生時に、ロッキング・イベント・モニター (アクティブな場合) に送信する、ロック・タイムアウト・イベント情報。
- デッドロック・イベントの発生時に、ロッキング・イベント・モニター (アクティブな場合) に送信する、デッドロック情報。
- 収集するロック待機イベント情報。
- 作業単位の終了時に、作業単位イベント・モニター (アクティブな場合) に送信される、このワークロードに関連したトランザクションごとの作業単位の情報。
- ワークロードがそのヒストグラムのテンプレートとして使用するヒストグラム・テンプレート。指定されるヒストグラム・テンプレートは、SYSCAT.HISTOGRAMTEMPLATEUSE ビューに反映されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、292 ページの『ワークロード管理のヒストグラム』を参照してください。

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。変更をコミットすると、各アプリケーションの次の作業単位の開始時にワークロードの再評価が行われるようになります。どのワークロードを選択するかによっては、アプリケーションが別のワークロードに再割り当てされることがあります。

次のタスク

1 人以上のセッション・ユーザーにそのワークロードに対する USAGE 特権を付与することが必要になる場合もあります。 (DBADM 権限を持つセッション・ユーザーには、すべてのワークロードを使用する暗黙特権があります。) 接続の接続属性がワークロードの接続属性と完全に一致する場合でも、セッション・ユーザーがそのワークロードに対する USAGE 特権を持っていない場合は、データ・サーバーは、ワークロードのオカレンスを作成するためにワークロードに接続を関連付けません。詳しくは、42 ページの『ワークロードに対する USAGE 特権の付与』を参照してください。

ワークロードを使用可能にする

DB2 データ・サーバーは、ワークロードに指定された接続属性を現行セッションの接続属性と突き合わせて検査します。データ・サーバーは、一致するワークロードを探す際、使用不可にされたワークロードを考慮しません。

始める前に

ワークロードを変更するためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

このタスクについて

ワークロードは、作成時はデフォルトで使用可能になっています。ワークロードを使用不可として作成した場合、ワークロード評価の実行時にデータ・サーバーにそのワークロードを考慮させるには、これを使用可能にする必要があります。

手順

ワークロードを使用可能にするには、次のようにします。

1. 使用可能にするワークロードを識別します。 次の例に示すように、SYSCAT.WORKLOADS ビューを照会することによって、使用不可になっているワークロードのセットを表示できます。

   ```
   SELECT * FROM SYSCAT.WORKLOADS WHERE ENABLED='N'
   ```

2. ALTER WORKLOAD ステートメントを使用して、使用不可になっているワークロードを使用可能にします。

   ```
   ALTER WORKLOAD...ENABLE
   ```

   ALTER WORKLOAD ステートメントが成功すると、ワークロードの定義がデータベース・カタログに書き込まれます。

3. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。

タスクの結果

実際にワークロードが使用可能になるのは次の作業単位が開始されるときです。その時点でワークロードの再評価が行われ、データ・サーバーはワークロードの再評価を実行する際に新しく使用可能になったワークロードを考慮するようになります。

ワークロードを使用不可にする

このタスクを使用して、ワークロードの割り当て中に、特定のワークロードが考慮の対象にならないようにします。ワークロードを使用不可にすると、データ・サーバーは、一致するワークロードを検索する際にそのワークロードを考慮しなくなります。代わりに、データ・サーバーは次の一致するワークロードに作業単位を割り当てます。一致するカスタム定義のワークロードがない場合、その作業はデフォルトのワークロードに割り当てられます。

始める前に

ワークロードを作成したり変更したりするためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

手順

ワークロードを使用不可にするには、次のようにします。

1. ALTER WORKLOAD ステートメントの DISABLE オプションを使用して、ワークロードを使用不可にします。

   ```
   ALTER WORKLOAD...DISABLE
   ```

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。

タスクの結果

実際にワークロードが使用不可になるのは次の作業単位が開始されるときです。その時点でワークロードの再評価が行われ、接続属性が一致し、必要な権限がある、次に使用可能なワークロードに接続が割り当てられます。

ワークロードのドロップ

ワークロードをドロップすると、ワークロードがデータベース・カタログから除去されます。

始める前に

ワークロードをドロップするためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

手順

ワークロードをドロップするには、次のようにします。

1. ALTER WORKLOAD ステートメントを指定して、ワークロードを使用不可にします。 詳しくは、 39 ページの『ワークロードを使用不可にする』を参照してください。ワークロードを使用不可にすると、そのワークロードの新しいオカレンスはデータベースで実行できなくなります。
2. WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を使用して、このワークロードのオカレンスが実行中でないことを確認します。 詳しくは、WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を参照してください。

   WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数は、アクティブ状態のワークロード・オカレンスに対応するアプリケーション・ハンドルを戻します。 **FORCE APPLICATION** コマンドを使用し、アプリケーション・ハンドルを使用してアプリケーションを終了させることができます。
3. DROP WORKLOAD ステートメントを指定して、ワークロードをドロップします。 例えば、ACCTNG ワークロードをドロップする場合は以下のステートメントを指定します。

```
DROP WORKLOAD ACCTNG
```

4. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューから除去されます。さらに、ワークロードの許可情報が SYSCAT.WORKLOADAUTH ビューから除去されます。

ワークロード・オカレンスにデータベースへのアクセスを許可する

データベースへのアクセスを許可していなかったワークロードで、これからオカレンスを実行できるようにする場合は、ワークロードに変更を加えてデータベースへのアクセスが許可されるようにします。デフォルトでは、ワークロードは作成されたときに、データベースへのアクセスを許可されます。

始める前に

ワークロードを変更してデータベースにアクセスできるようにするためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

このタスクについて

ワークロードがデータベースにアクセスしないようにするときも、データ・サーバーは引き続き、ワークロード割り当てを実行する際にそのワークロードを審査します。しかし、そのワークロードのオカレンスはすべて拒否されてエラーになります。

手順

ワークロードにデータベースへのアクセスを許可するには、次のようにします。

1. ALTER WORKLOAD ステートメントの ALLOW DB ACCESS オプションを使用して、ワークロードにデータベースへのアクセスを許可します。 例えば、WL1 というワークロードにデータベースへのアクセスを許可する場合は、次のステートメントを指定します。

   ```
   ALTER WORKLOAD WL1 ALLOW DB ACCESS
   ```

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。

タスクの結果

ワークロードを変更して、ワークロード・オカレンスがデータベースへのアクセスを許可されても、それが有効になるのはデータ・サーバーがそのワークロードの次の作業単位を分析するときです。例えば、DISALLOW DB ACCESS を指定していたワークロード A に変更を加えて ALLOW DB ACCESS を指定すると、ワークロード A の新しいオカレンスは実行を許可されます。変更前は、ワークロード A のすべてのオカレンスが拒否され、エラーになっていました。

ワークロード・オカレンスがデータベースにアクセスしないようにする

このタスクを使用して、どのワークロードがデータベースにアクセスできるかを制御します。ワークロード・オカレンスが実行できるようになる前に、データ・サーバーは、ワークロードがデータベースへのアクセスを許可されるかどうかを検査します。ワークロード・オカレンスにデータベースへのアクセスを許可しない場合は、ワークロード・オカレンスが拒否されたことを示すエラーが戻されます。

始める前に

ワークロードがデータベースにアクセスしないようにするためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

このタスクについて

ワークロード・オカレンスを防止することとワークロードを使用不可にすることは異なります。ワークロードを使用不可にすると、ワークロード定義はメモリーにキャッシュされないため、ワークロードの割り当ての際に考慮されません。

手順

ワークロードがデータベースにアクセスしないようにするには、次のようにします。

1. 次の例に示すように、ALTER WORKLOAD ステートメントの DISALLOW DB ACCESS オプションを使用します。

   ```
   ALTER WORKLOAD workload-name DISALLOW DB ACCESS ...
   ```

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。

タスクの結果

ワークロード・オカレンスがデータベースにアクセスしないようにするためのワークロードの変更は、既に実行されているワークロード・オカレンスの次の作業単位が開始されるときに有効になります。例えば、 ALLOW DB ACCESS と指定されているワークロード A に対し、DISALLOW DB ACCESS を指定して変更した場合、既に実行されているワークロード A のオカレンスは、次の作業単位が開始されるときに SQL エラーを受け取ります。ワークロード A の新規オカレンスは拒否されます。

ワークロードに対する USAGE 特権の付与

ワークロードを接続に関連付けるには、セッション・ユーザーがそのワークロードに対する USAGE 特権を持っている必要があります。
ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM 権限を持つユーザーは、暗黙的にすべてのワークロードに対する USAGE 特権を持ちます。

始める前に

GRANT USAGE ON WORKLOAD ステートメントを使用するためには、ACCESSCTRL、SECADM、または WLMADM 権限が必要です。

前提条件について詳しくは、21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

このタスクについて

データ・サーバーが着信接続の属性と一致するワークロードを見つけると、データ・サーバーはセッション・ユーザーがそのワークロードに対する USAGE 特権を持っているかどうかを検査します。セッション・ユーザーがそのワークロードに対する USAGE 特権を持っていない場合、データ・サーバーは次の一致するワークロードを探します。(つまり、セッション・ユーザーが USAGE 特権を持っていないワークロードは、あたかも存在していないかのように扱われます。) したがって、ワークロードの USAGE 特権を使用することにより、一致する複数のワークロードの中からどのワークロードにユーザー、グループ、またはロールを割り当てるかを一層制御できるようになります。例えば、同じ接続属性を持つワークロードを複数定義しておいて、それぞれのワークロードに対する USAGE 特権を特定のユーザー、グループ、またはロールだけに付与することができます。詳しくは、27 ページの『ワークロードの割り当て』を参照してください。

クライアントは、クライアント・ユーザー ID、クライアント・アプリケーション名、クライアント・ワークステーション名、およびクライアント・アカウンティング・ストリング (これらはワークロードへの接続の割り当てに使用される接続属性の一部です) を許可なしで設定することができます。それで、ワークロードの USAGE 特権は、どのセッション・ユーザーにワークロードを使用する権限を持たせるかを制御するためにも使用することができます。

USAGE 特権の情報は、SYSCAT.WORKLOADAUTH ビューを照会することによって表示できます。

RESTRICTIVE オプションを使用してデータベースを作成する場合は、データベース作成時に、SYSDEFAULTUSERWORKLOAD ワークロードに対する USAGE 特権が PUBLIC に付与されません。このワークロードに対する USAGE 特権を明示的に WLMADM および DBADM ユーザー以外に付与する必要があります。セッション・ユーザーが SYSDEFAULTUSERWORKLOAD を含むどのワークロードに対しても USAGE 特権を持っていない場合は、データ・サーバーがワークロードにデータベース接続を関連付けようとする際に SQL4707N が戻されます。

手順

ワークロードに対する USAGE 特権を付与するには、次のようにします。

1. GRANT USAGE ON WORKLOAD ステートメントを使用します。 特定のユーザー、グループ、ロール、または PUBLIC に対して USAGE 特権を付与することができます。例えば、ACCOUNTS ワークロードの USAGE 特権を CPA グループに付与するには、次のステートメントを発行します。

   ```
   GRANT USAGE ON WORKLOAD ACCOUNTS TO GROUP CPA
   ```

SYSDEFAULTADMWORKLOAD ワークロードに対する USAGE 特権は付与できません。SYSDEFAULTADMWORKLOAD ワークロードは、SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドを発行する ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM ユーザーだけが使用できます。

2. 変更をコミットします。 変更をコミットすると、SYSCAT.WORKLOADAUTH ビューが更新されます。GRANT ステートメントがコミットされるまで、データ・サーバーは、新しく許可されたユーザー、グループ、またはロールに対してワークロード割り当てを実行する際にそのワークロードを考慮できません。

ワークロードに対する USAGE 特権の取り消し

REVOKE USAGE ON WORKLOAD ステートメントを使用して、ワークロードに対する USAGE 特権を取り消すことができます。

始める前に

REVOKE USAGE ON WORKLOAD ステートメントを使用するためには、ACCESSCTRL、SECADM、または WLMADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

このタスクについて

SYSDEFAULTADMWORKLOAD ワークロードに対する USAGE 特権は、明示的に取り消すことはできません。このワークロードは、SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドを発行する ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM ユーザーだけが使用できます。したがって、REVOKE USAGE ON WORKLOAD ステートメントは、SYSDEFAULTADMWORKLOAD に対しては機能しません。

手順

ワークロードに対する USAGE 特権を取り消すには、次のようにします。

1. REVOKE USAGE ON WORKLOAD ステートメントを使用します。 特定のユーザー、グループ、ロール、または PUBLIC から USAGE 特権を取り消すことができます。例えば、ACCOUNTS ワークロードの USAGE 特権を PUBLIC から取り消すには、次のステートメントを指定します。

   ```
   REVOKE USAGE ON WORKLOAD ACCOUNTS FROM PUBLIC
   ```

2. 変更をコミットします。 変更をコミットすると、SYSCAT.WORKLOADAUTH ビューが更新されます。REVOKE ステートメントがコミットされるまで、データ・サーバーは、ワークロード割り当てを実行する際にこのワークロードを考慮します。

例: ワークロードの割り当て

データベース接続が確立された後の最初の作業単位の初めに、使用可能になっている各ワークロードの接続属性を評価することによって、データ・サーバーは接続をワークロードに割り当てます。

接続属性の値またはワークロード定義自体が作業単位の間に変更される場合は、各作業単位の初めにワークロードの再評価が行われます。

以下の図は、ワークロードの割り当てを示しています。AppA を介して照会をサブミットする Marketing グループのユーザーは、APPAQUERIES ワークロードに割り当てられます。PAYROLL が APPAQUERIES の前に置かれているにも関わらず、PAYROLL ワークロードには割り当てられません。これは、ワークロード PAYROLL の定義が SESSION_USER GROUP キーワードを Finance として指定しているためです。AppA を介して照会をサブミットする Finance グループのユーザーは、FINANCE ワークロードに割り当てられます。PAYROLL ワークロードの方がより特定的で、かつその定義に AppA と Finance の両方が指定されているにも関わらず、PAYROLL ワークロードには割り当てられません。これは、FINANCE ワークロードが PAYROLL ワークロードの前に置かれているためです。AppB を介して照会をサブミットする Marketing グループのユーザーは、SYSDEFAULTUSERWORKLOAD ワークロードに割り当てられます。これは、FINANCE、PAYROLL、または APPAQUERIES ワークロード定義に指定されている接続属性がどれも AppB アプリケーションまたは Marketing グループと一致しないためです。

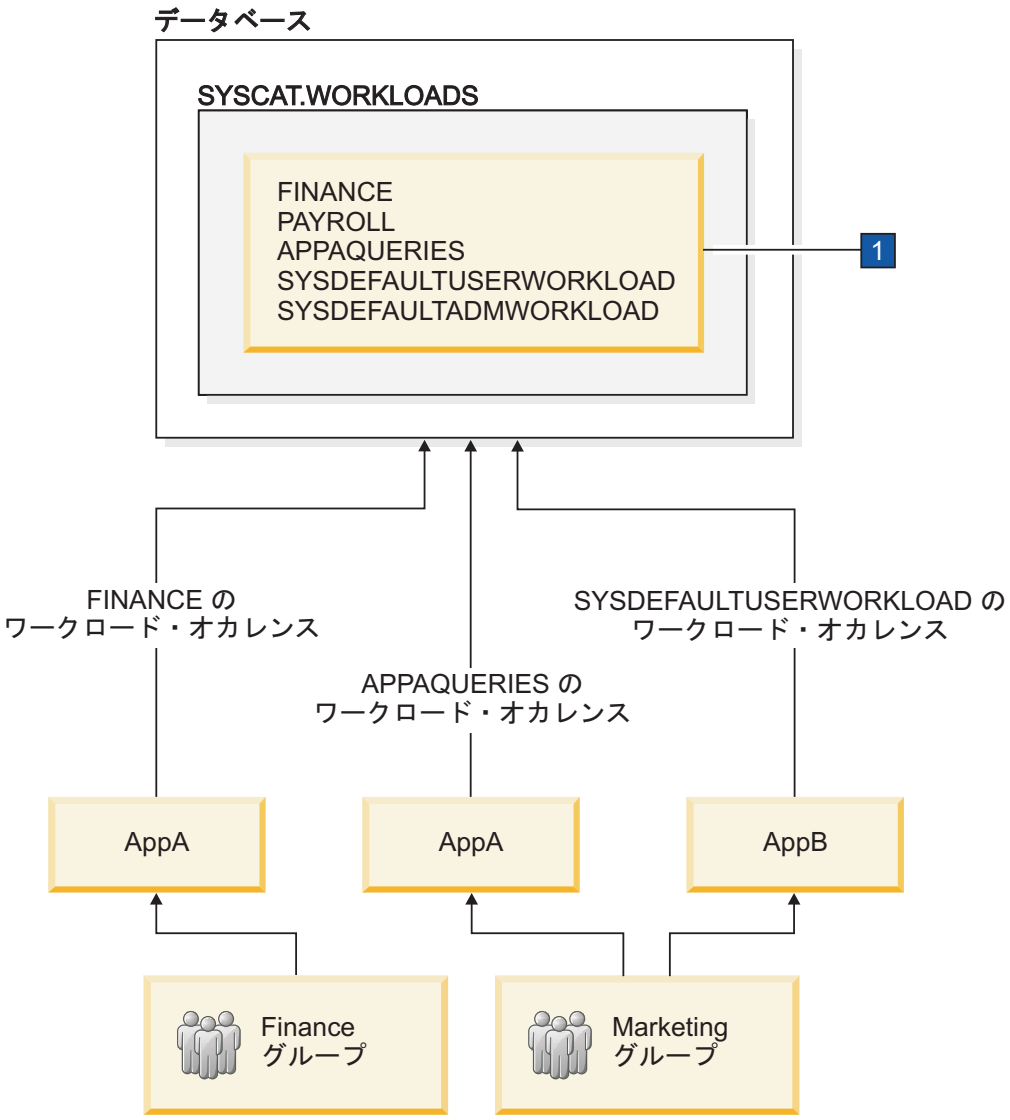


図 5. ワークロードの割り当ての例

1 前の図では、CREATE WORKLOAD ステートメントは次のとおりです。

```
CREATE WORKLOAD PAYROLL APPLNAME ('AppA') SESSION_USER GROUP ('FINANCE')
SERVICE CLASS SC1

CREATE WORKLOAD APPAQUERIES APPLNAME('AppA') POSITION LAST
SERVICE CLASS SC2

CREATE WORKLOAD FINANCE SESSION_USER GROUP ('FINANCE') SERVICE CLASS SC1
POSITION BEFORE PAYROLL
```

デフォルトで、多くのアプリケーション・サーバーは、処理対象のすべてのクライアント要求に対し、同じ情報を使って接続をセットアップし、同じクライアント情報を渡します (存在する場合)。WebSphere および Cognos などの一部の製品は、クライアント情報フィールドを介して各要求に関する固有の情報をプッシュダウンする機能を提供しています。この情報は、DB2 内でエンド・ユーザー要求を一意的に識別します。他のほとんどの製品は、エンド・ユーザー要求の処理を開始する前に固有のクライアント情報を DB2 に送信できるように、アプリケーション・サーバーをカスタマイズするための方法を提供しています。

固有のクライアント属性をアプリケーション・サーバーから指定することにより、DB2 内の要求の特殊な扱いや、さまざまなクライアントからの要求をさまざまなワークロード (および、さまざまなサービス・クラス) に割り当てることが可能になります。

以下の図は、セッション・ユーザー APPUSER を使用してデータベースへの接続を確立するアプリケーション・サーバーを介して、さまざまなユーザー・アプリケーション (marketing.exe、auditing.exe、および reporting.exe) が照会をサブミットする、3 層の環境の例を示しています。次の 3 つのワークロードが定義されます。marketing.exe によってサブミットされる照会用のワークロード、reporting.exe によってサブミットされる照会用のワークロード、および残りの照会用のワークロードです。図に示されているとおり、marketing.exe によってサブミットされる照会を MARKETING ワークロードに割り当てるには、アプリケーション・サーバーは sqleseti API を呼び出して、CURRENT CLIENT_APPLNAME 特殊レジスターの値を marketing.exe に設定します。同様に、reporting.exe によってサブミットされる照会を REPORTING ワークロードに割り当てるには、サーバーは sqleseti を呼び出して、CURRENT CLIENT_APPLNAME 特殊レジスターの値を reporting.exe に設定します。図でサーバーが sqleseti を呼び出して CURRENT CLIENT_USERID 特殊レジスターを Lidia に設定する (他には何も変更されない。つまりクライアント・アプリケーション名が引き続き reporting.exe に設定される) 場合、ワークロードの再割り当ては生じないということに注意してください。これは、特に CURRENT CLIENT_USERID が Lidia に設定されて定義されているワークロードがないためです。

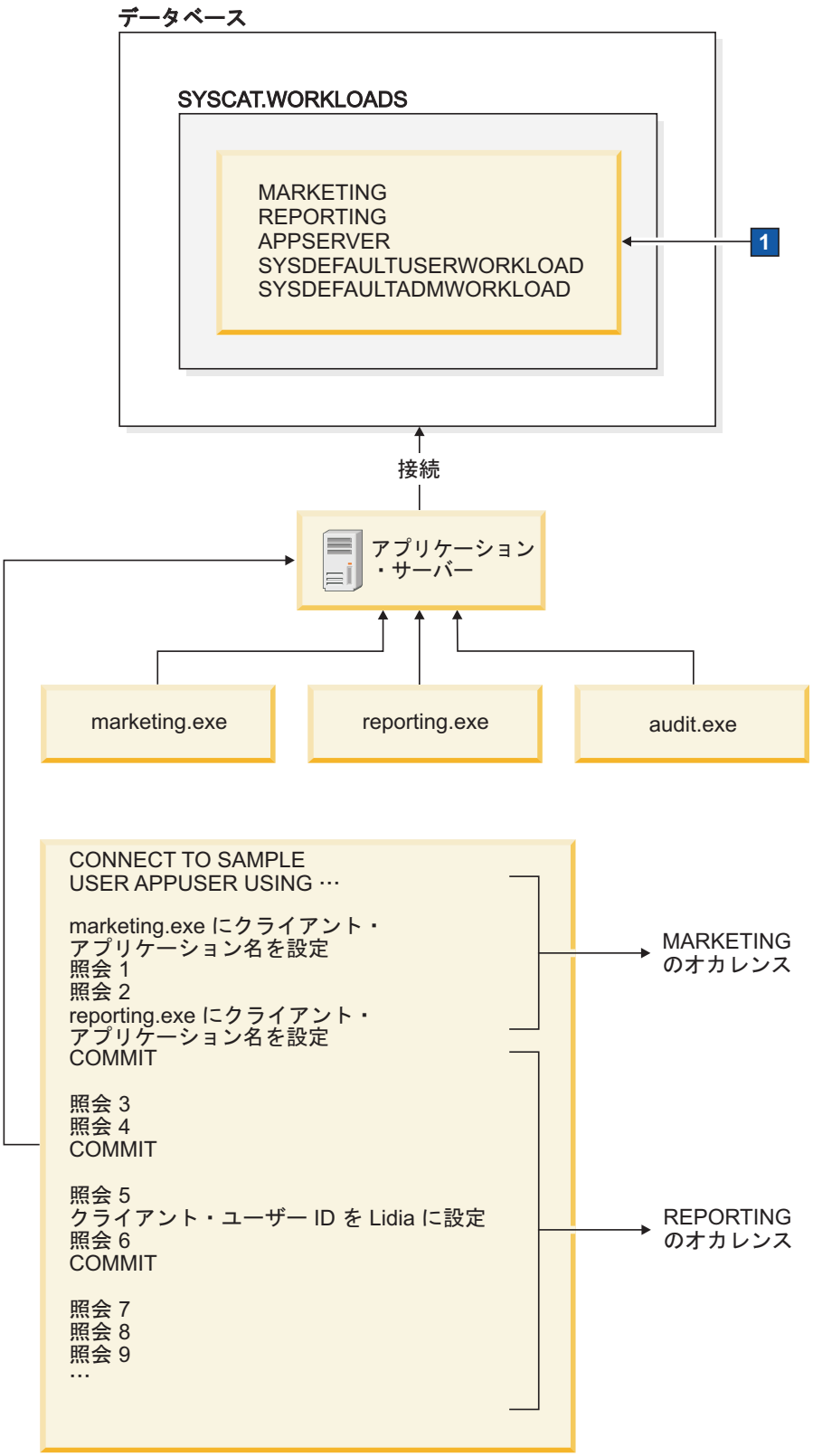


図 6. 3 層環境におけるワークロードの割り当ての例

以下のステートメントは、前の図の枠 1 で指定されたワークロードを定義するために使用されます。

```
CREATE WORKLOAD MARKETING SESSION_USER ('APPUSER')
CURRENT CLIENT_APPLNAME ('marketing.exe') SERVICE CLASS SC2
POSITION AT 1

CREATE WORKLOAD REPORTING SESSION USER ('APPUSER')
CURRENT CLIENT_APPLNAME ('reporting.exe') SERVICE CLASS SC4
POSITION AFTER MARKETING

CREATE WORKLOAD APPSERV SESSION_USER ('APPUSER')
SERVICE CLASS SC1
```

例: ワークロード属性に単一値が含まれる場合のワークロードの割り当て

このトピックの例では、データ・サーバーがワークロードの割り当てを実行する方法について示します。この例では、各ワークロード接続属性ごとに 1 つの値のみが指定されます。

以下のワークロードがカタログ内に存在すると想定します。

表 5. カタログ内のワークロード

評価順序	ワークロード名	ADDRESS	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	REPORTS		AppA								
2	INVENTORY REPORT		AppB	LYNN		ACCOUNTING	TELEMKTR				
3	SALES REPORT		AppC	KATE	KATE		SALESREP				
4	AUDIT REPORT		AppB			ACCOUNTING	FINANALYST				
5	EXPENSE REPORT		AppA	TIM			EXPENSE APPROVER				
6	AUDIT RESULT				LYNN			LYNN			Audit Group

以下の属性を持つデータベース接続が確立されると想定します。

表 6. データベース接続属性

ADDRESS	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
9.26.53.111	AppA	TIM	TIM	FINANCE	FINANALYST、EXPENSE APPROVER	NULL	NULL	NULL	Business account

最初の作業単位がサブミットされるときに、データ・サーバーは、リストにある最初のワークロードから始めて、カタログ内の各ワークロードを検査します。さらに、一致する属性を含むワークロードが見つかるまで、ワークロードを昇順で処理します。一致するワークロードが見つかると、作業単位はそのワークロードのオカレンスの下で実行されます。接続を割り当てるワークロードを決定する場合、データ・サーバーは接続属性を決定論的順序で比較します。

データ・サーバーは最初に REPORTS ワークロードを検査して一致がないかどうか調べます。REPORTS ワークロードはリストの最初にあります。

表 7. カタログ内の REPORTS ワークロード

評価順序	ワークロード名	ADDRESS	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	REPORTS		AppA								

データ・サーバーは、以下の決定論的順序で接続属性を検査します。

1. APPLNAME。データベース接続の APPLNAME の値 AppA は REPORTS ワークロード用の APPLNAME の値と一致します。
2. SYSTEM_USER。これはワークロード定義では設定されていません。すべての値 (NULL 値を含む) が一致であると見なされます。
3. SESSION_USER。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
4. SESSION_USER GROUP。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
5. SESSION_USER ROLE。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
6. CURRENT CLIENT_USERID。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
7. CURRENT CLIENT_APPLNAME。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
8. CURRENT CLIENT_WRKSTNNAME。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
9. CURRENT CLIENT_ACCTNG。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。

この場合、REPORTS ワークロードの接続属性と接続で渡される情報との間に明示的および暗黙的な一致が存在するために、データ・サーバーは REPORTS ワークロードを潜在的な一致として選択します。ワークロードを選択した後、データ・サーバーは、セッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを検査します。セッション・ユーザー TIM に REPORTS ワークロードに対する USAGE 特権がある場合、そのワークロードは接続に使用されます。ただし、TIM が REPORTS ワークロードに対する USAGE 特権を所有しない場合、データ・サーバーは、INVENTORYREPORT ワークロードを検査して一致がないかどうかを調べることによって処理を続行します。

EXPENSEREPORT ワークロードで追加の接続属性が指定されているため、TIM をそのワークロードに割り当てたいと仮定します。この場合、ワークロードの評価順序を変更して、次のようにワークロード・リストの REPORTS の前に EXPENSEREPORT を配置します。

```
ALTER WORKLOAD EXPENSEREPORT POSITION AT 1
```

さらに、以下の SQL ステートメントを使用して同じ結果を得ることもできます。

```
ALTER WORKLOAD EXPENSEREPORT BEFORE REPORTS
```

ALTER WORKLOAD ステートメントを有効にするには、COMMIT ステートメントを ALTER WORKLOAD ステートメントの直後に発行する必要があります。カタログでの ALTER WORKLOAD ステートメントの効果は次のとおりです。

表 8. EXPENSEREPORT ワークロードを位置変更した後のカタログ内のワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	EXPENSE REPORT	AppA	TIM			EXPENSE APPROVER				

表 8. *EXPENSEREPORT* ワークロードを位置変更した後のカタログ内のワークロード *(続き)*

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
2	REPORTS	AppA								
3	INVENTORY REPORT	AppB	LYNN		ACCOUNTING	TELEMKTR				
4	SALES REPORT	AppC	KATE	KATE		SALESREP				
5	AUDIT REPORT	AppB			ACCOUNTING	FINANALYST				
6	AUDIT RESULT			LYNN			LYNN			Audit Group

TIM がまだ EXPENSEREPORT ワークロードに対する USAGE 特権を持っていない場合は、以下のステートメントを発行する必要があります (COMMIT ステートメントによって GRANT ステートメントを有効にします)。

```
GRANT USAGE ON WORKLOAD EXPENSEREPORT TO USER TIM
COMMIT
```

次の作業単位の開始時に、ワークロードの再割り当てが実行され、データ・サーバーは TIM からの接続を EXPENSEREPORT ワークロードに割り当てます。さらに、同じ属性を持つ他の接続によってサブミットされた新しい作業単位も、EXPENSEREPORT ワークロードと関連付けられます。

例: 複数のワークロードが存在する場合の作業単位に対するワークロードの割り当て

このトピックの例では、データ・サーバーが既存のワークロードへの接続を割り当てるために、ワークロード評価を実行する方法を示します。

以下のワークロードがカタログ内で定義されていると想定します。

表 9. *カタログ内のワークロード*

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	EXPENSE REPORT	AppB	TIM			EXPENSE APPROVER				
2	REPORTS	AppB								
3	INVENTORYREPORT	AppA	LYNN		ACCOUNTING	TELEMKTR				
4	SALES REPORT	AppC	KATE	KATE		SALESREP				
5	AUDIT REPORT	AppA			ACCOUNTING	FINANALYST				
6	AUDIT RESULT			LYNN			LYNN			Audit Group

以下の属性を持つデータベース接続が確立されると想定します。

表 10. *データベース接続属性*

APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
AppA	LYNN	LYNN	ACCOUNTING	FINANALYST, SALESREP	LYNN	NULL	wrkstn2	Audit group

最初の作業単位がサブミットされると、データ・サーバーはカタログ内の各ワークロードを昇順の評価順序で検査し、接続が提供するものと接続属性が一致するワークロードを見つけると停止します。ワークロードを確認する際、データ・サーバーは接続属性を決定論的順序で比較します。

最初に、データ・サーバーは EXPENSEREPORT ワークロードを検査します。

表 11. カタログ内の EXPENSEREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	EXPENSEREPORT	AppB	TIM			EXPENSE APPROVER				

ワークロード定義にある APPLNAME 属性は AppB ですが、接続から渡された APPLNAME 属性は AppA なので、マッチングは不可能です。データ・サーバーは、リストの 2 番目の REPORTS ワークロードへ進みます。

表 12. カタログ内の REPORTS ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
2	REPORTS	AppB								

この場合もやはり、ワークロード定義にある APPLNAME 属性は AppB なので AppA と一致しません。データ・サーバーはリスト内の 3 番目のワークロード、INVENTORYREPORT へ進みます。

表 13. カタログ内の INVENTORYREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
3	INVENTORYREPORT	AppA	LYNN		ACCOUNTING	TELEMKTR				

データ・サーバーはサブミットされた接続属性と INVENTORYREPORT ワークロードの間の一致を検査します。属性が以下の順で検査されます

1. APPLNAME。ワークロード定義および接続の両方が AppA の値を持つため、一致します。
2. SYSTEM_USER。ワークロード定義および接続の両方が LYNN の値を持つため、一致します。
3. SESSION_USER。接続は LYNN の値を渡しました。ワークロードに SESSION_USER 属性は設定されていないため、NULL 値を含め接続から渡されたどんな値でも一致します。
4. SESSION_USER GROUP。ワークロード定義および接続の両方が ACCOUNTING の値を持つため、一致します。
5. SESSION_USER ROLE。ワークロード定義は TELEMKTR の値を指定していますが、接続は FINANALYST および SALESREP の値を提供しました。この属性には一致が発生しません。

データ・サーバーは INVENTORYREPORT ワークロードと接続の属性を突き合わせようとするのを停止し、リストにある 4 番目のワークロードである SALESREPORT へ進みます。

表 14. カタログ内の SALESREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
4	SALESREPORT	AppC	KATE	KATE		SALESREP				

SALESREPORT ワークロード定義の APPLNAME が AppC なので、接続 (APPLNAME に対して AppA の値を渡した) との一致は発生しません。そこで、データ・サーバーはリスト内の 5 番目のワークロード、AUDITREPORT へ進みます。

表 15. カタログ内の AUDITREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
5	AUDITREPORT	AppA			ACCOUNTING	FINANALYST				

データ・サーバーは、AUDITREPORT ワークロードと接続の属性を決定論的順序で比較します。

1. APPLNAME。ワークロード定義および接続の両方が AppA の値を持つため、一致します。
2. SYSTEM_USER。接続は LYNN の値を渡しました。ワークロードに SYSTEM_USER 属性は設定されていないため、接続から渡されたどんな値でも一致します。
3. SESSION_USER。接続は LYNN の値を渡しました。ワークロードに SESSION_USER 属性は設定されていないため、接続から渡されたどんな値でも一致します。
4. SESSION_USER GROUP。ワークロードおよび接続の両方がこの属性に対して ACCOUNTING の値を持つため、一致します。
5. SESSION_USER ROLE。ワークロードおよび接続の両方がこの属性に対して FINANALYST の値を持つため、一致します。
6. CURRENT CLIENT_USERID。ワークロードに CURRENT CLIENT_USERID 属性は設定されていないため、接続から渡されたどんな値でも一致します。
7. CURRENT CLIENT_APPLNAME。ワークロードに CURRENT CLIENT_APPLNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
8. CURRENT CLIENT_WRKSTNNAME。ワークロードに CURRENT CLIENT_WRKSTNNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
9. CURRENT CLIENT_ACCTNG。ワークロードに CURRENT CLIENT_ACCTNG 属性は設定されていないため、接続から渡されたどんな値でも一致します。

すべての接続属性を処理し、一致するワークロードを見つけた後、データ・サーバーはセッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを検査します。LYNN は AUDITREPORT ワークロードに対する USAGE 特権を持っていないと想定します。この状態では、たとえすべての接続属性が一致しても、このワークロードは接続と関連付けられません。データ・サーバーは評価リスト内の 6 番目のワークロード、AUDITRESULT へ進みます。

表 16. カタログ内の AUDITRESULT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
6	AUDITRESULT			LYNN			LYNN			Audit Group

データ・サーバーは、AUDITRESULT ワークロードと接続の属性を決定論的順序で比較します。

1. APPLNAME。ワークロードに APPLNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
2. SYSTEM_USER。ワークロードに SYSTEM_USER 属性は設定されていないため、接続から渡されたどんな値でも一致します。
3. SESSION_USER。ワークロードおよび接続の両方がこの属性に対して LYNN の値を持つため、一致します。
4. SESSION_USER GROUP。ワークロードに SESSION_USER GROUP 属性は設定されていないため、接続から渡されたどんな値でも一致します。
5. SESSION_USER ROLE。ワークロードに SESSION_USER ROLE 属性は設定されていないため、接続から渡されたどんな値でも一致します。
6. CURRENT CLIENT_USERID。ワークロードおよび接続の両方がこの属性に対して LYNN の値を持つため、一致します。
7. CURRENT CLIENT_APPLNAME。ワークロードに CURRENT CLIENT_APPLNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
8. CURRENT CLIENT_WRKSTNNAME。ワークロードに CURRENT CLIENT_WRKSTNNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
9. CURRENT CLIENT_ACCTNG。ワークロードおよび接続の両方がこの属性に対して Audit Group の値を持つため、一致します。

すべての接続属性を処理し、一致するワークロードを見つけた後、データ・サーバーはセッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを検査します。この状態で、セッション・ユーザー LYNN は AUDITRESULT ワークロードに対する USAGE 特権を持つと想定します。接続属性がすべて一致し、セッション・ユーザーが USAGE 特権を持っているので、接続は AUDITRESULT ワークロードに割り当てられます。

例: ワークロード属性に複数の値が含まれる場合のワークロードの割り当て

このトピックの例では、データ・サーバーがワークロードの割り当てを実行する方法について示します。この例では、いくつかのワークロード定義は 1 つの接続属性に対して複数の値を許可します。

以下のワークロードがカタログ内で定義されていると想定します。

表 17. カタログ内のワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURREN CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	ITEMINQ		KYLE, GEORGE		RETAIL, SALES					
2	DAILY TRANS REPORT	AppC		KYLE, CAROL	SALES, ACCOUNTING					
3	SALES SUMMARY	AppA, AppB				ACCOUNTANT, FINANALYST				

以下の属性を持つデータベース接続が確立されると想定します。

表 18. データベース接続属性

APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURREN CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
AppC	LINDA	KYLE	SALES	ACCOUNTANT	LINDA	NULL	NULL	Business Account

最初の作業単位がサブミットされると、データ・サーバーはカタログ内の各ワークロードを昇順の評価順序で検査し、接続が提供するものと接続属性が一致するワークロードを見つけると停止します。ワークロードを確認する際、データ・サーバーは接続属性を決定論的順序で比較します。

最初に、データ・サーバーは ITEMINQ ワークロードを検査します。

表 19. カタログ内の ITEMINQ ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURREN CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	ITEMINQ		KYLE, GEORGE		RETAIL, SALES					

データ・サーバーはサブミットされた接続属性と ITEMINQ ワークロードの間の一致を検査します。属性が以下の順で検査されます

1. APPLNAME。ワークロードに APPLNAME 属性は設定されていないため、NULL 値を含め、接続から渡されたどんな値でも一致します。
2. SYSTEM_USER。接続は LINDA の値を渡しました。しかし、ITEMNO ワークロードの値は KYLE および GEORGE です。この属性には一致が発生しません。

データ・サーバーは ITEMNO ワークロードと接続を突き合わせようとするのを停止し、リストにある 2 番目のワークロードである DAILYTRANSREPORT へ進みます。

表 20. カタログ内の DAILYTRANSREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURREN CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
2	DAILYTRANSREPORT	AppC		KYLE, CAROL	SALES, ACCOUNTING					

データ・サーバーは、DAILYTRANSREPORT ワークロードと接続の属性を決定論的順序で比較します。

1. APPLNAME。ワークロード定義および接続の両方が AppC の値を持つため、一致します。
2. SYSTEM_USER。ワークロードに SYSTEM_USER 属性は設定されていないため、NULL 値を含め、接続から渡されたどんな値でも一致します。
3. SESSION_USER。接続によって渡された SESSION_USER の値は KYLE で、ワークロード SESSION_USER の値の 1 つと一致します。接続から渡されたのが CAROL だったとしても、KYLE と CAROL の両方が DAILYTRANSREPORT ワークロード定義の一部として指定されているので、一致することになります。
4. SESSION_USER GROUP。接続によって渡された SESSION_USER GROUP の値は SALES で、ワークロード SESSION_USER GROUP の属性に対して指定された SALES の値に一致します。接続から渡されたのが ACCOUNTING だったとしても、SALES と ACCOUNTING の両方がワークロード定義に指定されているので、一致することになります。

5. SESSION_USER ROLE。ワークロードに SESSION_USER ROLE 属性は設定されていないため、接続から渡されたどんな値でも一致します。
6. CURRENT CLIENT_USERID。ワークロードに CURRENT CLIENT_USERID 属性は設定されていないため、接続から渡されたどんな値でも一致します。
7. CURRENT CLIENT_APPLNAME。ワークロードに CURRENT CLIENT_APPLNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
8. CURRENT CLIENT_WRKSTNNAME。ワークロードに CURRENT CLIENT_WRKSTNNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
9. CURRENT CLIENT_ACCTNG。ワークロードに CURRENT CLIENT_WRKSTNNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。

すべての接続属性を処理し、接続に一致するワークロードを見つけた後、データ・サーバーはセッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを検査します。この状態で、セッション・ユーザー KYLE は DAILYTRANSREPORT ワークロードに対する USAGE 特権を持つと想定します。接続属性がすべて一致し、セッション・ユーザーが USAGE 特権を持っているので、接続は DAILYTRANSREPORT ワークロードに割り当てられます。

タイプ、コスト、またはワーク・クラスでアクセスされるデータによる作業の識別

ワークロードに付随するアクティビティーの起点に注目する接続属性を使用する方法に加えて、ワーク・クラスを含むワーク・クラス・セットを作成することで、タイプ、コスト、またはアクセス対象として予測されるデータに基づいてアクティビティーを識別できます。

ワーク・クラスとは、アクティビティーの属性に基づいて個々のデータベース・アクティビティーを分類する方法のことです。ワーク・クラスのワーク・アクションが定義されていれば、そのワーク・アクションがワーク・クラスに適用され、ワーク・クラス内のアクティビティーの管理方法もそのワーク・アクションで決まります。詳しくは、108 ページの『ワーク・アクション・セットによるアクティビティーのタイプへの制御の適用』を参照してください。

以下の表は、ワーク・クラスに使用可能なタイプ・キーワードと、さまざまなキーワードに対応する SQL ステートメントを示しています。ロード・ユーティリティーを除いて、以下の表内のすべてのステートメントは、EXECUTE、EXECUTE IMMEDIATE、または OPEN 要求の処理の直前に代行受信されます。ロード・ユーティリティーは、クライアントから実行した場合、データ・サーバー上で実際のロード操作を開始する前に要求を発行することがあります。

表 21. 作業タイプ

作業タイプ・キーワード	該当する SQL ステートメント
READ (SET ステートメントを含んだ組み込み READ SQL)	• すべての SELECT ステートメント (SELECT INTO、VALUES INTO、全選択) **例外:** DELETE、INSERT、または UPDATE を含む SELECT ステートメントは除外されます。 • すべての XQuery ステートメント
WRITE (SET ステートメントを含んだ組み込み WRITE SQL)	• すべての UPDATE ステートメント (検索条件付き、位置指定) • すべての DELETE ステートメント (検索条件付き、位置指定) • すべての INSERT ステートメント (values、副選択) • すべての MERGE ステートメント • DELETE、INSERT、または UPDATE ステートメントを含むすべての SELECT ステートメント
CALL	CALL ステートメント CALL ステートメントは、CALL および ALL ワーク・クラス・タイプにのみ分類されます。 **注:** 無名ブロックと自律型ルーチンは、CALL ステートメントに分類されます。
DML (SET ステートメントを含んだ組み込み READ または WRITE SQL)	READ および WRITE ワーク・クラス・タイプに分類される全ステートメント。
DDL	• すべての ALTER ステートメント • すべての CREATE ステートメント • COMMENT ステートメント • DECLARE GLOBAL TEMPORARY TABLE ステートメント • DROP ステートメント • FLUSH PACKAGE CACHE ステートメント • すべての GRANT ステートメント • REFRESH TABLE • すべての RENAME ステートメント • すべての REVOKE ステートメント • SET INTEGRITY ステートメント
LOAD	ロード・ユーティリティー ロード・ユーティリティーは、LOAD および ALL ワーク・クラス・タイプにのみ分類されます。

表 21. 作業タイプ (続き)

作業タイプ・キーワード	該当する SQL ステートメント
ALL	先行するキーワードすべてによって表される作業タイプ。 **注:** アクションがしきい値の場合、しきい値が適用されるデータベース・アクティビティーは、しきい値のタイプに応じて異なります。例えば、しきい値のタイプが ESTIMATEDSQLCOST の場合、(timeron 単位の) 見積コストがある DML アクティビティーだけがこのしきい値により影響を受けます。 詳しくは、 73 ページの『例: ALL キーワードを使用して定義されたワーク・クラスの処理』を参照してください。

以下の図は、作業タイプ・キーワードの階層図を示しています。

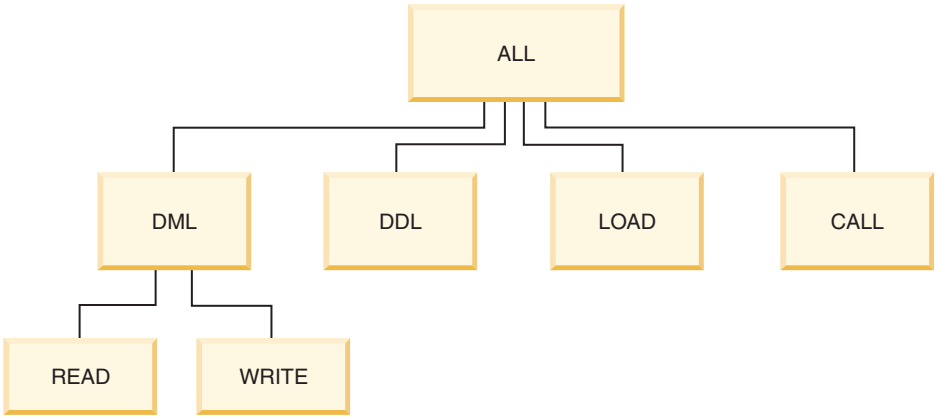


図 7. 作業タイプ・キーワード

使用可能などのキーワードにも属さない SQL ステートメントは分類されず、ワーク・クラスやワーク・クラス・セットが存在しないかのように振る舞います。例えば、ステートメントが SET SCHEMA であり、ワーク・クラス・セット内の唯一のワーク・クラスが作業タイプ DML である場合、そのステートメントは分類されず、適用できるワーク・アクションはありません。それで、アクションが MAP の場合、SET SCHEMA アクティビティーはデフォルトのサービス・サブクラス (SYSDEFAULTSUBCLASS) で実行します。アクションがしきい値の場合、アクティビティーに適用されるしきい値はありません。

追加識別

ワーク・クラスでは、DML 作業 (あるいは READ および WRITE ステートメント) の ID に予測エレメントを使用することもできます。予測エレメントは役に立ちます。なぜなら、データベース・アクティビティーがデータ・サーバーのリソースの消費を開始する前にアクションを取るために使用できる、それらのアクティビティ

ーに関する情報を得られるからです。以下の表には、ワーク・クラスがサポートする予測エレメントに関する情報が示されています。

表 22. 予測識別の特性

予測エレメント	説明
見積コスト	特定の timeron の範囲内の DML を含めるため、DB2 コンパイラーから取得できる見積コストを使用します (例えば、1 000 000 timeron を超える見積コストを持つすべての大規模な照会に対してワーク・クラスを作成します)
見積カーディナリティー	特定の戻り行数の範囲内の DML を含めるため、DB2 コンパイラーから返される見積行数 (カーディナリティー) を使用します (例えば、500 000 行を超える行を返すと見積もられる大規模な照会に対してワーク・クラスを作成します)
アクセスされるデータの見積もり	特定のデータ・タグが割り当てられたデータにアクセスする DML を含めるために、DB2 コンパイラーからアクセスされるデータ見積もりを使用します (例えば、データ・タグ値 3 でタグ付けされた表スペースのデータにアクセスすると見積もられる照会に関するワーク・クラスを作成します) **注:** アクセス対象として DB2 コンパイラーが予測する表スペースの集合は、最適化された SQL ステートメントに基づきます (これはユーザー指定の SQL ステートメントと異なる可能性があります)。場合によっては、例えば範囲パーティション表の中に挿入するとき、アクセス対象として予測される表スペースの数は、期待される表スペースより多くなります。

CALL ステートメントが呼び出すプロシージャーのスキーマ名を使用することによりアクティビティーを識別することもできます。

ワークロード属性とワーク・クラス・タイプに基づいて作業を識別して、次のステージである作業の管理のために準備できます。

ワーク・クラスおよびワーク・クラス・セットの扱いについて詳しくは、以下のトピックを参照してください。

ワーク・クラスとワーク・クラス・セット

ワーク・クラスとは、アクティビティーの属性に基づいて個々のデータベース・アクティビティーを分類する方法のことです。ワーク・クラスは、さまざまなワーク・アクション・セットが共有できるワーク・クラス・セットにグループ分けされます。

アクティビティーに関連付けるワーク・クラスを決めるデータベース・アクティビティー属性の例としては、アクティビティー・タイプ (DDL、DML、LOAD)、見積コスト (使用可能な場合)、見積カーディナリティー (使用可能な場合)、見積データ・タグ、およびスキーマ (使用可能な場合) があります。

ワーク・クラス

ワーク・クラスには以下の属性があります。

- ワーク・クラス名。これはワーク・クラス・セット内で固有でなければなりません。
- データベース・アクティビティー属性。これは以下の情報で構成されます。
  - このワーク・クラスに分類されるデータベース・アクティビティーのタイプ。定義済みキーワード (例えば、CALL、READ、WRITE、DML、DDL、LOAD、および ALL) を使用して、データベース要求をさまざまなカテゴリーに分類することができます。異なるタイプのデータベース・アクティビティーを、その作業タイプに基づいて、1 つのワーク・クラスに関連付けることができます。例えば、WRITE キーワードには、UPDATE、DELETE、INSERT、MERGE と、DELETE、INSERT、または UPDATE を含む SELECT が含まれます。詳しくは、 56 ページの『タイプ、コスト、またはワーク・クラスでアクセスされるデータによる作業の識別』を参照してください。
  - データベース・アクティビティーの DML または XQuery タイプを詳細に分類する範囲情報。
    - 指定する範囲のタイプ (timeron コストまたはカーディナリティーのいずれか)。値の範囲の指定はオプションです。例えば、ワーク・クラスの範囲を指定する場合、見積コストが 100 timeron 未満のすべての照会を、他の照会とは異なる仕方で処理するように指定できます。
    - 範囲の下限。
    - 範囲の上限。
  - 呼び出されるルーチンのスキーマ。スキーマの指定はオプションです。ワーク・クラスの定義時に、呼び出されるプロシージャーのスキーマに応じて CALL ステートメントをさらに詳細に分類するためにスキーマ属性を使用できます。例えば、ワーク・クラスのスキーマに SCHEMA1 を指定し、作業タイプが CALL の場合、SCHEMA1 プロシージャーを呼び出すすべての CALL ステートメントはそのワーク・クラスに分類されます。 CALL または ALL 以外のワーク・クラス・タイプにスキーマを指定すると、エラー SQL0628N が返されます。
  - アクティビティーで変更が加えられる可能性のあるデータに与えられる識別タグ。例えばワーク・クラスのデータ・タグ 3 を指定した場合、データ・タグ 3 を持つ表スペースまたはストレージ・グループ内のデータを処理するアクティビティーを分離して、異なる方法で扱うことができます。
- ワーク・クラスの評価順序 (またはワーク・クラス・セット内のワーク・クラスの位置)。詳しくは、 68 ページの『ワーク・クラス・セット内のワーク・クラスの評価順序』を参照してください。
- ワーク・クラスを一意的に識別する自動生成クラス ID。

ワーク・クラスは以下の 2 とおりの方法で作成できます。

- 新規ワーク・クラスを含む新規ワーク・クラス・セットを作成するには、CREATE WORK CLASS SET ステートメントの WORK CLASS キーワードを使用します。
- 新規ワーク・クラスを既存のワーク・クラス・セットに追加するには、ALTER WORK CLASS SET ステートメントの ADD キーワードを使用します。

ワーク・クラスを変更するには、ALTER WORK CLASS SET ステートメントの ALTER WORK CLASS キーワードを使用することができます。

ワーク・クラス・セットからワーク・クラスをドロップするには、ALTER WORK CLASS SET ステートメントの DROP WORK CLASS キーワードを使用し、ワーク・クラス・セットをドロップするには、DROP WORK CLASS SET ステートメントを使用することができます。

ワーク・クラスを表示するには、SYSCAT.WORKCLASSES ビューを照会します。

ワーク・クラス・セット

1 つ以上のワーク・クラスをグループ化するには、ワーク・クラス・セットを使用します。ワーク・クラス・セットは以下の属性で構成されます。

- ワーク・クラス・セットの固有の記述名。
- ワーク・クラス・セットに提供するコメント。
- ゼロ以上のワーク・クラス (ワーク・クラスはワーク・クラス・セット内でしか存在できませんが、ワーク・クラス・セットにはワーク・クラスが含まれていなくても構いません)。
- ワーク・クラス・セットを一意的に識別する自動生成 ID。

新規ワーク・クラス・セットを作成するには、CREATE WORK CLASS SET ステートメントを使用します。空のワーク・クラス・セットを作成して後からワーク・クラスを追加することもできますし、1 つ以上のワーク・クラスを含むワーク・クラス・セットを作成することもできます。

既存のワーク・クラス・セットを変更するには、ALTER WORK CLASS SET ステートメントを使用して以下の方法で実行できます。

- ワーク・クラス・セットにワーク・クラスを追加します。
- ワーク・クラス・セット内のワーク・クラスのワーク・クラス属性を変更します。
- ワーク・クラス・セットからワーク・クラスをドロップします。

ワーク・クラス・セット属性は変更できません。

ワーク・クラス・セットをドロップするには、DROP WORK CLASS SET ステートメントを使用します。

ワーク・クラス・セットを表示するには、SYSCAT.WORKCLASSSETS カタログ・ビューを照会します。

以下の図は、ワーク・クラス・セット内のワーク・クラスの例を示しています。

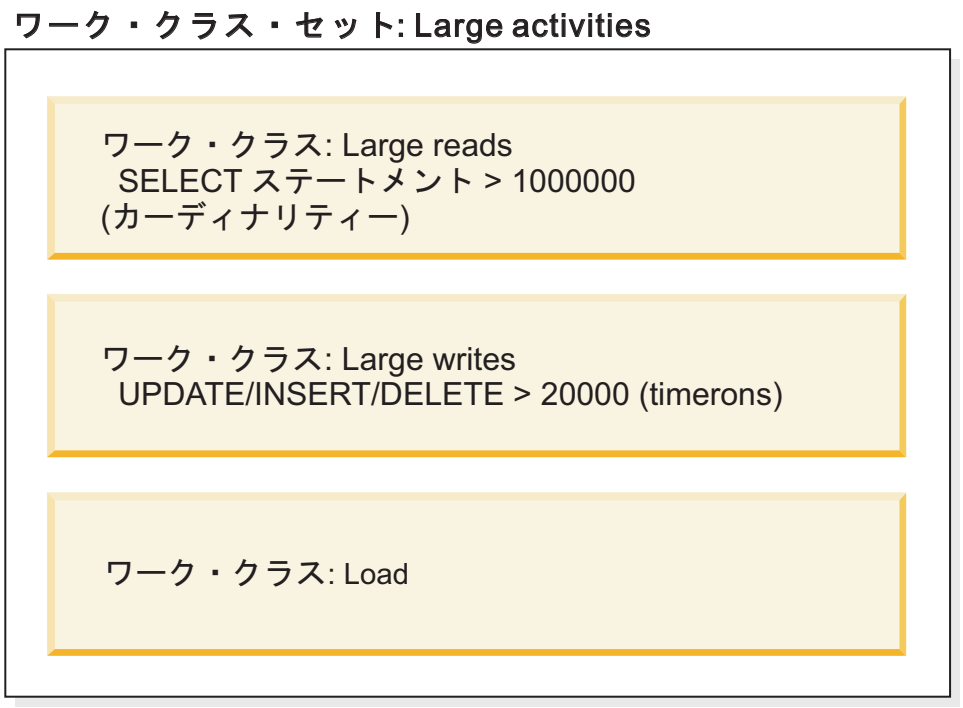


図 8. ワーク・クラスおよびワーク・クラス・セットの例

システム上でワーク・クラス・セットが有効であるには、ワーク・アクション・セットを定義して、それをワーク・クラス・セットと関連付けなければなりません。ワーク・アクション・セットを使用することで、ワーク・クラス・セットをサービス・スーパークラス、ワークロード、またはデータベースと関連付けて、どのアクションが分類内のデータベース・アクティビティーに適用されるかを示すことができます。ワーク・クラス・セットに対してワーク・アクション・セットを作成しない場合、データ・サーバーはワーク・クラス・セットを無視します。

ワーク・クラスの作成

ワーク・クラスを作成するには、CREATE WORK CLASS SET ステートメントまたは ALTER WORK CLASS SET ステートメントを使用します。

始める前に

ワーク・クラスを作成するためには、WLMADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』
- 命名規則

手順

ワーク・クラスを作成するには、次のようにします。

1. ワーク・クラスを作成して同時に新規ワーク・クラス・セットも作成する場合、または新規ワーク・クラスを既存のワーク・クラス・セットに追加する場合は、以下のようにします。

- 新規ワーク・クラス・セットに追加する新規ワーク・クラスを作成するには、CREATE WORK CLASS SET ステートメントの WORK CLASS キーワードを使用します。
- 既存のワーク・クラス・セットに追加する新規ワーク・クラスを作成するには、ALTER WORK CLASS SET ステートメントの ADD WORK CLASS キーワードを使用します。

新規ワーク・クラスに、以下に挙げるプロパティーを 1 つ以上指定します。

- ワーク・クラスの名前。この名前は、ワーク・クラス・セット内で固有でなければなりません。
- ワーク・クラスの属性。これらの属性は、アクティビティーをワーク・クラスと関連付けるために使用されます。
  - ワーク・クラスを使用する作業のタイプ。この特性を指定するには WORK TYPE パラメーターを使用します。
    - READ。非更新 SELECT アクティビティー、およびすべての XQuery アクティビティーを表します。 READ キーワードを指定するときには、オプションの for-from-to-clause 引数または data-tag-clause 引数を指定することもできます。
      - オプションの for-from-to-clause 引数を使用することで、ステートメントのコスト (timeron 単位) またはステートメントのカーディナリティー (戻される行数) の範囲を指定します。最初の値には数値を指定する必要があります。2 番目の値には、数値を指定するか、または値 UNBOUNDED を指定してアクティビティーのコストまたはカーディナリティーに上限を設けないことを指定できます。この引数は、WRITE キーワード、DML キーワード、および ALL キーワードに対しても指定できます。

        例えば、5000 timeron 以上のコストの SELECT アクティビティーをこのワーク・クラスに関連付けるには、以下のように指定します。

        ```
        WORK TYPE READ FOR TIMERONCOST FROM 5000 TO UNBOUNDED
        ```

      - アクティビティーによってアクセスされると見積もられるデータを識別するデータ・タグを指定するには、オプションの data-tag-clause 引数を使用します。 1 から 9 までの値を指定できます。data-tag-clause が指定されない場合、照会でアクセスされるデータの種類に制限は課されません。この引数は、WRITE キーワード、DML キーワード、および ALL キーワードに対しても指定できます。

        例えば、データ・タグ値 1 が割り当てられた表スペースのデータにアクセスすると見積もられる SELECT アクティビティーをこのワーク・クラスに関連付けるには、次のように指定します。

        ```
        WORK TYPE READ DATA TAG LIST CONTAINS 1
        ```

    - WRITE。データベース内のデータを更新する SQL アクティビティーを表します。 WRITE キーワードを指定するときには、オプションの for-from-to-clause 引数または data-tag-clause 引数を指定することもできます。

例えば、値 5 でタグ付けされたデータをコンパイル時に扱うと見積もられるすべてのデータ書き込みアクティビティーをこのワーク・クラスに関連付けるには、次のように指定します。

```
WORK TYPE WRITE FOR CARDINALITY FROM 50 TO 100
```

- CALL。CALL アクティビティーを表します。

  CALL キーワードを指定するときに、ROUTINES IN SCHEMA キーワードも指定して、特定のスキーマ内のルーチンへの CALL アクティビティーのみがこのワーク・クラスに関連付けられるようにすることができます。例えば、ACCOUNTS スキーマ内のルーチンへの呼び出しのみをこのワーク・クラスに関連付ける場合、以下のように指定します。

  ```
  WORK TYPE CALL ROUTINES IN SCHEMA ACCOUNTS
  ```

- DML。READ および WRITE キーワードの両方によってカバーされる SQL アクティビティーを表します。

  例えば、見積コスト (timeron 単位) が 500 から 1000 の範囲で、コンパイル時に値 8 でタグ付けされたデータを扱うと見積もられるすべての DML アクティビティーを関連付けるには、次のように指定します。

  ```
  WORK TYPE DML FOR TIMERONCOST FROM 500 TO 1000 DATA TAG LIST CONTAINS 8
  ```

- DDL。以下のアクティビティーを表します。
  - ALTER
  - CREATE
  - COMMENT
  - DECLARE GLOBAL TEMPORARY TABLE
  - DROP
  - FLUSH PACKAGE CACHE
  - GRANT
  - REFRESH TABLE
  - RENAME
  - REVOKE
  - SET INTEGRITY

  例えば、すべての DDL アクティビティーをこのワーク・クラスに関連付けるには、以下のように指定します。

  ```
  WORK TYPE DDL
  ```

- LOAD。LOAD アクティビティーを表します。

  例えば、LOAD アクティビティーをこのワーク・クラスに関連付けるには、以下のように指定します。

  ```
  WORK TYPE LOAD
  ```

- ALL。先行するキーワードすべてによって示されるすべての作業タイプを表します。

  ワーク・クラス・タイプに ALL を指定するときに、ROUTINES IN SCHEMA キーワードも指定して、特定のスキーマ内のルーチンへの

CALL アクティビティーのみがこのワーク・クラスに関連付けられるようにすることができます。for-from-to-clause 引数を指定して、指定されている見積コスト (timeron) またはカーディナリティーの DML アクティビティーすべてをこのクラスに入れるようにすることもできます。例えば、300 行から 1500 行のカーディナリティーを持つ DML アクティビティーと、NEWHIRES スキーマから呼び出されるルーチンの両方をこのワーク・クラスに関連付けるには、以下のステートメントを指定します。なお、あるデータ・タグ値でタグ付けされた表スペースのデータにアクセスするすべての DML アクティビティーを示すために、data-tag-clause 引数を指定することもできます。このワーク・クラスのタイプは ALL であるため、これは、LOAD アクティビティーおよび DDL アクティビティーといった、スキーマまたはカーディナリティーを持たない他のアクティビティーにも適用されます。

```
WORK TYPE ALL FOR CARDINALITY FROM 300 TO 1500 ROUTINES
IN SCHEMA NEWHIRES
```

– オプション。ワーク・クラス・セット内でのワーク・クラスの位置。ワーク・クラス・セット内でのワーク・クラスの位置によって、アクティビティーをワーク・クラスにクラス分けする際にワーク・クラスが評価される順序が決定されます。ワーク・クラスの割り当てが行われるとき、データ・サーバーはまずオブジェクト (サービス・スーパークラスまたはデータベースのいずれか) に関連するワーク・クラス・セットを判別し、それからそのワーク・クラス・セット内で、ワーク・アクションが関連付けされた最初に一致するワーク・クラスを選択します。POSITION キーワードを使用して、以下のいずれかを指定します。

  - LAST。ワーク・クラスは、ワーク・クラス・セット内でワーク・クラスのリストの最後尾に配置されます。以下に例を示します。

    ```
    WORK TYPE ... POSITION LAST
    ```

  - BEFORE *work-class-name*。ワーク・クラスは、ワーク・クラス・セット内の、指定のワーク・クラスの前に作成されます。以下に例を示します。

    ```
    WORK TYPE ... POSITION BEFORE LARGEDDL
    ```

  - AFTER *work-class-name*。ワーク・クラスは、ワーク・クラス・セット内の、指定のワーク・クラスの後に作成されます。以下に例を示します。

    ```
    WORK TYPE ... POSITION AFTER LARGEDDL
    ```

  - AT *integer*。ワーク・クラスは、ワーク・クラス・セット内の、整数値で指定された位置に作成されます。以下に例を示します。

    ```
    WORK TYPE ... POSITION AT 3
    ```

2. 変更をコミットします。 変更をコミットすると、ワーク・クラスは SYSCAT.WORKCLASSES ビューに追加されます。

ワーク・クラスの変更

ワーク・クラスを変更する必要がある場合には、ALTER WORK CLASS SET ステートメントを使用します。

始める前に

ワーク・クラスを変更するためには、WLMADM または DBADM 権限が必要です。

その他の前提条件は、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

手順

ワーク・クラスを変更するには、次のようにします。

1. ALTER WORK CLASS SET ステートメントの ALTER キーワードを使用して、以下のプロパティー (複数可) を変更します。 これらのプロパティーについてサポートされる値の説明は、 62 ページの『ワーク・クラスの作成』を参照してください。
   - FOR キーワード。例えば、FOR キーワードに指定される値を CARDINALITY から TIMERONCOST に変更できます。
   - FROM *from-value* TO *to-value* 引数。例えば、引数を `FROM 50 TO 100` から `FROM 500 TO 1500` に変更できます。
   - ROUTINES IN SCHEMA または ROUTINES IN ALL キーワード (CALL アクティビティーの場合)。例えば、現在ワーク・クラスでスキーマの指定がない場合、スキーマを追加できます。また、キーワード ALL を指定して、ルーチンのスキーマに関わりなく、ワーク・クラスがすべての CALL ステートメントに適用されるようにすることもできます。ALL はデフォルトです。
   - DATA TAG LIST CONTAINS キーワード。例えば、このキーワードに指定される値を ANY から 8 に変更できます。
   - POSITION キーワード。この後に、LAST、BEFORE、AFTER、AT というキーワードが続きます。 POSITION BEFORE または POSITION AFTER を指定する場合は、変更したワーク・クラスの配置に使用するワーク・クラスも指定する必要があります。 POSITION AT を指定する場合は、位置の番号を含める必要があります。例えば、AT キーワードを使用して、最後の位置から任意の位置に、または LAST キーワードを使用して任意の位置から最後の位置に、ワーク・クラスを移動させることができます。
2. 変更をコミットします。 変更をコミットすると、ワーク・クラスは SYSCAT.WORKCLASSES ビューで更新されます。

ワーク・クラスのドロップ

必要でなくなったワーク・クラスは、ワーク・クラス・セットからドロップすることができます。

始める前に

ワーク・クラスをドロップするためには、WLMADM または DBADM 権限が必要です。

その他の前提条件は、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

手順

ワーク・クラスをドロップするには、次のようにします。

1. ALTER WORK CLASS SET ステートメントの DROP キーワードを使用します。 ワーク・クラス・セットと関連付けられているワーク・アクション・セット内に、ドロップしたいワーク・クラスに依存しているワーク・アクションがある場合には、そのワーク・クラスをドロップすることはできません。この場合、まず従属ワーク・アクションすべてをドロップしてから、ワーク・クラスをドロップする必要があります。
2. 変更をコミットします。 変更をコミットすると、ワーク・クラスは SYSCAT.WORKCLASSES ビューから除去されます。

ワーク・クラス・セットの作成

ワーク・クラス・セットを作成するには、CREATE WORK CLASS SET ステートメントを使用します。

始める前に

ワーク・クラス・セットを作成するためには、WLMADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』

手順

ワーク・クラス・セットを作成するには、次のようにします。

1. CREATE WORK CLASS SET ステートメントを使用して、以下に挙げるワーク・クラス・セットのプロパティーを指定します。
   - ワーク・クラス・セットの名前。指定する名前は、データベースの中で固有でなければなりません。
   - オプション: ワーク・クラス・セットの 1 つ以上のワーク・クラス。詳しくは、 62 ページの『ワーク・クラスの作成』を参照してください。
2. 変更をコミットします。 変更をコミットすると、ワーク・クラス・セットが SYSCAT.WORKCLASSSETS ビューに追加されます。

ワーク・クラス・セットの変更

ワーク・クラス・セットの属性をワーク・クラス・セットの作成後に変更することはできません。ただし、ALTER WORK CLASS SET ステートメントを使用して、ワーク・クラス・セットに含まれるワーク・クラスを追加、変更、およびドロップすることはできます。

始める前に

ワーク・クラス・セットを変更するためには、WLMADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』

- 命名規則

手順

1. ワーク・クラス・セットにワーク・クラスを追加する場合は、ADD キーワードを使用します。 ワーク・クラスを追加するときに指定できる各キーワードについては、 62 ページの『ワーク・クラスの作成』を参照してください。
2. ワーク・クラスを変更する場合は、ALTER キーワードを使用します。 ワーク・クラスを変更する方法については、 65 ページの『ワーク・クラスの変更』を参照してください。
3. ワーク・クラスをドロップする場合は、DROP キーワードを使用します。 ワーク・クラス・セットからワーク・クラスをドロップする方法については、 66 ページの『ワーク・クラスのドロップ』を参照してください。ワーク・クラス・セットからすべてのワーク・クラスをドロップする場合は、そのワーク・クラス・セット自体をドロップすることができます。詳しくは、『ワーク・クラス・セットのドロップ』を参照してください。
4. 変更をコミットします。 変更をコミットすると、SYSCAT.WORKCLASSES ビューが更新されて、追加、変更、またはドロップされたワーク・クラスがあればそれらが示されます。

ワーク・クラス・セットのドロップ

ワーク・クラス・セットをドロップするには、DROP WORK CLASS SET ステートメントを使用します。

始める前に

ワーク・クラス・セットをドロップするためには、WLMADM または DBADM 権限が必要です。

このタスクについて

ワーク・クラス・セットに関連付けられているワーク・アクション・セットがない場合にのみ、ワーク・クラス・セットをドロップできます。ワーク・クラス・セットをドロップする場合は、まずそれに従属するワーク・アクション・セットをドロップする必要があります。

手順

ワーク・クラス・セットをドロップするには、次のようにします。

1. DROP WORK CLASS SET ステートメントを使用します。
2. 変更をコミットします。 変更をコミットすると、ワーク・クラス・セットが SYSCAT.WORKCLASSSETS ビューから除去されます。さらに、そのワーク・クラス・セットの一部だったすべてのワーク・クラスが、SYSCAT.WORKCLASSES ビューから除去されます。

ワーク・クラス・セット内のワーク・クラスの評価順序

ワーク・クラス・セットは、データベース・アクティビティーと一致する複数のワーク・クラスを持つことができます。ワーク・クラス・セットからアクティビティ

ーの分類先となるワーク・クラスを選択するために、データ・サーバーは評価順序に応じてワーク・クラスを評価し、アクティビティーと一致する最初のワーク・クラスで停止します。

一致するワーク・クラスが存在しない場合、データベース・アクティビティーはどのワーク・クラスにも属さず、そのアクティビティーに適用されるワーク・アクションはありません。

ワーク・クラス・セットの作成時または変更時に、ワーク・クラス・セット内のワーク・クラスの評価順序を調整することができます。ワーク・クラス・セットの作成時または変更時には、以下の 3 つの方法の 1 つを使用して、ワーク・クラス・セット内でのワーク・クラスの位置を決めます。

- リスト内でワーク・クラスの絶対位置を指定します。

  例えば、POSITION AT 2 とすることができます。この場合、ワーク・クラスはワーク・クラス・セット内の 2 番目の位置に配置され、2 番目の位置にあったワーク・クラスは 3 番目の位置に、3 番目のワーク・クラスは 4 番目の位置に、という具合になります。 CREATE WORK CLASS SET または ALTER WORK CLASS SET ステートメントによりワーク・クラスに指定された位置が、ワーク・クラス・セット内のワーク・クラスの合計数より大きい数値の場合、ワーク・クラスはリストの最後に位置します。
- POSITION BEFORE または POSITION AFTER キーワードを使用して、既にワーク・クラス・セット内にあるワーク・クラスとの相対的なワーク・クラスの位置を指定します。
- ワーク・クラスの作成時に位置を省略します。

  この状況では、新規ワーク・クラスはリストの末尾に配置されます。ワーク・クラス・セットのリスト内のワーク・クラスに指定する位置は、必ずしも SYSCAT.WORKCLASSES ビュー内の EVALUATIONORDER 列の実際の値ではありません。データ・サーバーは、ギャップを避けるために順序値を自動的に割り当てます。

ワーク・クラスは受け取られる順序で処理されますが、これは評価順序に影響を与える可能性があります。例えば、以下のステートメントを発行すると仮定します。

```
ALTER WORK CLASS SET WCS ALTER WORK CLASS C1 POSITION AT 1
   ALTER WORK CLASS C2 POSITION AT 1
```

この結果、C1 ワーク・クラスは評価順序 2 を持ち、C2 ワーク・クラスは評価順序 1 を持ちます。処理された最後のワーク・クラスは C2 だったためです。

ワーク・クラスへのアクティビティーの割り当て

ワーク・クラス・セットが、ワーク・アクション・セットを経てデータベース、ワークロード、またはサービス・スーパークラスに関連付けられている場合、実行、即時に実行、またはオープン要求の処理の実行前、あるいはロード・ユーティリティーの実行直前に、データベース・アクティビティーはワーク・クラス・セット内のワーク・クラスで指定されたいずれかの基準に一致するかどうかを判別するために検査されます。

ワーク・クラスは、ワーク・クラス・セット内でその評価順序によってソートされます。この評価順序に基づいて、データベース・アクティビティーはデータベース・アクティビティーの属性 (アクティビティー・タイプやカーディナリティーなど) に基づく各ワーク・クラスに照らして検査されます。これは一致するものがあるか、またはワーク・クラス・セット内のワーク・クラスのリストの末尾に達するまで続けられます。

以下のワーク・クラスがワーク・クラス・セット内にあると仮定します。

- 評価順序: 1。ワーク・クラス名: MyLoad。ワーク・クラス・タイプ: LOAD
- 評価順序: 2。ワーク・クラス名: SmallRead。ワーク・クラス・タイプ: READ。他の属性: 見積コスト < 300 timeron
- 評価順序: 3。ワーク・クラス名: AllDML。ワーク・クラス・タイプ: DML
- 評価順序: 4。ワーク・クラス名: LargeRead。ワーク・クラス・タイプ: READ。他の属性: 見積コスト > 301 timeron
- 評価順序: 5。ワーク・クラス名: MyDDL。ワーク・クラス・タイプ: DDL

見積コスト 200 timeron の SELECT ステートメントを受け取る場合、それは SmallRead ワーク・クラスに割り当てられます。 DDL アクティビティー (CREATE TABLE など) が着信する場合、それには MyDDL ワーク・クラスが割り当てられます。見積コスト 500 timeron の SELECT ステートメントを受け取る場合、AllDML は LargeRead ワーク・クラスの前に位置するので、それは AllDML ワーク・クラスに割り当てられます。詳しくは、 73 ページの『例: ALL キーワードを使用して定義されたワーク・クラスの処理』を参照してください。

しきい値ごとにサポートされる作業の分類

ワーク・アクションで使用できるしきい値タイプはどれも任意のワーク・クラスに関連付けできますが、すべてのタイプのデータベース・アクティビティーがこれらしきい値タイプのすべてについてサポートされるわけではありません。

例えば、DDL のワーク・クラスを作成する場合、そのワーク・クラスを ESTIMATEDSQLCOST しきい値ワーク・アクションと関連付けます。このしきい値は、DDL に分類される要求のいずれにも適用されません。これは、DDL ステートメントが見積もりコストを持たないためです。 ALL のワーク・クラスを作成する場合、そのワーク・クラスを ESTIMATEDSQLCOST しきい値ワーク・アクションに関連付けます。すべてのデータベース・アクティビティーは ALL ワーク・クラスに属しますが、しきい値は見積もりコストを持つデータベース・アクティビティーにのみ適用されます。

次の表に、どのワーク・クラス・カテゴリーがどのしきい値タイプでサポートされるかを示します。

表 23. しきい値ごとにサポートされる作業の分類

	155 ページの『ACTIVITYTOTALTIME しきい値』	160 ページの『ESTIMATEDSQLCOST しきい値』	166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』	156 ページの『CPUTIME しきい値』
READ (SET ステートメントを含んだ組み込み READ SQL)	可	可	可[1]	可

表 23. しきい値ごとにサポートされる作業の分類 (続き)

	155 ページの『ACTIVITYTOTALTIME しきい値』	160 ページの『ESTIMATEDSQLCOST しきい値』	166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』	156 ページの『CPUTIME しきい値』
WRITE (SET ステートメントを含んだ組み込み WRITE SQL)	可	可	可[1]	可
CALL	可	不可	不可	可
DML (SET ステートメントを含んだ組み込み READ または WRITE SQL)	可	可	可[1]	可
DDL	可	不可	可[1]	不可
LOAD	可	不可	可[1]	不可
ALL	可	一部可	可[1]	一部可

注:

1. ユーザー定義関数 (UDF) 内で実行される、およびこれらの作業の分類を含むアクティビティーは、CONCURRENTDBCOORDACTIVITIES しきい値の影響を受けません。

表 24. しきい値ごとにサポートされる作業の分類 (続き)

	160 ページの『SQLROWSREAD しきい値』	163 ページの『SQLROWSRETURNED しきい値』	164 ページの『SQLTEMPSPACE しきい値』
READ (SET ステートメントを含んだ組み込み READ SQL)	可	可	可
WRITE (SET ステートメントを含んだ組み込み WRITE SQL)	可	可	可
CALL	不可	不可 (注を参照)	不可
DML (SET ステートメントを含んだ組み込み READ または WRITE SQL)	可	可	可
DDL	不可	不可	不可
LOAD	不可	不可	不可
ALL	一部可	一部可	一部可

注:

- 呼び出されたプロシージャーのステートメントから行が戻されることがありますが、これらの行は CALL ステートメントの結果として戻されるわけではないため、SQLROWSRETURNED しきい値による制御は受けません。

例: アクティビティー・タイプごとのワークロードの分析

実行中のアクティビティーのタイプに応じて、DB2 ワークロード管理表関数を使用して環境内のワークロードを調べることができます。

状況によっては、LOAD アクティビティーなどの特定のタイプのアクティビティーの動作を把握したい場合があります。例えば、以下のようにして現在システム内に存在する LOAD アクティビティー数を知ることができます。

```
SELECT COUNT(*)
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(CAST(NULL AS BIGINT), -2))
AS ACTS
WHERE ACTIVITY_TYPE = 'LOAD'
```

以下の例に示されているように、WLM_GET_WORK_ACTION_SET_STATS 表関数を使用すると、DB2 ワークロード管理統計を最後にリセットして以降にサブミットされた特定のタイプのアクティビティー数をカウントできます。READ アクティビティーおよび LOAD アクティビティー用の READCLASS ワーク・クラスと LOADCLASS ワーク・クラスがあるとします。* は、READCLASS または LOADCLASS ワーク・クラスに分類されない他のすべてのアクティビティーを表します。

```
SELECT SUBSTR(WORK_ACTION_SET_NAME,1,18) AS WORK_ACTION_SET_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(WORK_CLASS_NAME,1,15) AS WORK_CLASS_NAME,
       LAST_RESET,
       SUBSTR(CHAR(ACT_TOTAL),1,14) AS TOTAL_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS('', -2)) AS WASSTATS
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME, PART

WORK_ACTION_SET_NAME PART WORK_CLASS_NAME LAST_RESET                TOTAL_ACTS
-------------------- ---- -------------- -------------------------- ----------
AdminActionSet       0    ReadClass      2005-11-25-18.52.49.343000 8
AdminActionSet       1    ReadClass      2005-11-25-18.52.50.478000 0
AdminActionSet       0    LoadClass      2005-11-25-18.52.49.343000 2
AdminActionSet       1    LoadClass      2005-11-25-18.52.50.478000 0
AdminActionSet       0    *              2005-11-25-18.52.50.478000 0
AdminActionSet       1    *              2005-11-25-18.52.50.478000 0
```

LOAD アクティビティーを特定のサービス・サブクラスにマップするワーク・アクション・セットを作成すると、LOAD アクティビティーの平均存続時間を表示できます。例えば、LOAD アクティビティーを、サービス・スーパークラス MYSUPERCLASS の下のサービス・サブクラス LOADSERVICECLASS にマップするとします。その場合、WLM_GET_SERVICE_SUBCLASS_STATS 表関数を次のように照会できます。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3)) AS AVGLIFETIME
FROM TABLE
   (WLM_GET_SERVICE_SUBCLASS_STATS('MYSUPERCLASS', 'LOADSERVICECLASS', -2))
AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART

SUPERCLASS_NAME     SUBCLASS_NAME      PART AVGLIFETIME
------------------- ------------------ ---- ----------------------
SYSDEFAULTUSERCLASS LOADSERVICECLASS   0                  4691.242
SYSDEFAULTUSERCLASS LOADSERVICECLASS   1                  4644.740
SYSDEFAULTUSERCLASS LOADSERVICECLASS   2                  4612.431
SYSDEFAULTUSERCLASS LOADSERVICECLASS   3                  4593.451
```

例: 特定のアクティビティー・タイプを管理するためのワーク・クラス・セットの使用

以下の例では、ワーク・クラス・セットを使用して DML アクティビティーを管理する方法について示します。

毎日非常に多くのアプリケーションを NONAME データベースで実行しており、最近少しのパフォーマンス問題が発生しているとします。これらの問題の一部に対応するには、データベースで同時に実行可能な大規模な照会 (つまり、見積コストが 9999 timeron より大きい、または見積カーディナリティーが 9999 行より大きいすべての照会) の数を制御する必要があると判断します。

データベースに対して実行可能な大規模な照会の数を制御するには、以下のようにします。

1. 次の 2 つのワーク・クラスが入っている MYWORKCLASSSET ワーク・クラス・セットを作成します。見積コストが大きい照会用および見積カーディナリティーが大きい照会用のワーク・クラスです。以下に例を示します。

   ```
   CREATE WORK CLASS SET MYWORKCLASSSET
    (WORK CLASS LARGEESTIMATEDCOST WORK TYPE DML
   FOR TIMERONCOST FROM 10000 TO UNBOUNDED,
   WORK CLASS LARGECARDINALITY WORK TYPE DML
   FOR CARDINALITY FROM 10000 TO UNBOUNDED)
   ```

2. データベース・レベルで MYWORKCLASSSET ワーク・クラス・セットのワーク・クラスに適用される 2 つのワーク・アクションが入った DATABASEACTIONS ワーク・アクション・セットを作成します。

   ```
   CREATE WORK ACTION SET DATABASEACTIONS FOR DATABASE
   USING WORK CLASS SET LARGEQUERIES
   (WORK ACTION ONECONCURRENTQUERY ON WORK CLASS LARGEESTIMATEDCOST
   WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 1 STOP EXECUTION,
   WORK ACTION TWOCONCURRENTQUERIES ON WORK CLASS LARGECARDINALITY
   WHEN CONCURRENTDBCOORDACTIVITIES > 2 AND QUEUEDACTIVITIES > 3 STOP EXECUTION)
   ```

さらに、いくつかの大規模な管理アプリケーションがデータベースに対して毎日実行されており、これらのアプリケーションを 1 つのリソース・プールで実行することを希望するとします。この目標を達成するために、これらのアプリケーションのために ADMINAPPS というサービス・スーパークラスを作成することができます。アプリケーションごとに、それを ADMINAPPS サービス・スーパークラスにマップするためのワークロードを作成します。

照会 (SELECT ステートメント) が迅速に実行されることは重要であるため、ADMINAPPS サービス・スーパークラスに、これらの照会用の SELECTS というサービス・サブクラスを作成することにします。

SELECT ステートメントを SELECTS サービス・サブクラスにマップするには、以下のようにします。

1. データベースを更新しないすべての SELECT ステートメント用のワーク・クラスが入った SELECTDML ワーク・クラス・セットを作成します。

   ```
   CREATE WORK CLASS SET SELECTDML (WORK CLASS SELECTCLASS WORK TYPE READ)
   ```

2. ADMINAPPSACTIONS ワーク・アクション・セットを作成します。このワーク・アクション・セットには、サービス・スーパークラス・レベルでワーク・アクション・セット SELECTDML のワーク・クラスに適用されるワーク・アクションが入ります。

   ```
   CREATE WORK ACTION SET ADMINAPPSACTIONS FOR SERVICE CLASS ADMINAPPS
   USING WORK CLASS SET SELECTDML
   (WORK ACTION MAPSELECTS ON WORK CLASS SELECTCLASS MAP ACTIVITY TO SELECTS)
   ```

例: ALL キーワードを使用して定義されたワーク・クラスの処理

以下の例では、ALL として定義されるワーク・クラスを処理する方法について示します。これは、データベース内で認識されるすべてのアクティビティーを網羅する可能性があるワーク・クラスです。

DB2 ワークロード管理インフラストラクチャーは、考えられるすべてのデータベース・アクティビティーのドメインから、以下に挙げる特定のサブセットを認識しま

す。すなわち、ロード操作、CALL ステートメント、すべての DDL ステートメント、およびすべての DML ステートメントです。DB2 ワークロード管理は、これらの*認識されるアクティビティー* のモニターおよび制御を完全にサポートします。

ALL のタイプのワーク・クラスをマッピング・ワーク・アクションとともに使用した場合、認識されるすべてのデータベース・アクティビティーは、ワーク・アクションで指定されたサービス・サブクラスにマップされます。 ALL の作業タイプのワーク・クラスをしきい値のワーク・アクションとともに使用した場合、しきい値のタイプによって、しきい値が適用されるデータベース・アクティビティーが決定されます。次のような例について考慮します。

以下のワーク・クラスを使用して Example というワーク・クラス・セットを作成するとします。ワーク・クラスの評価順序は、以下のとおりです。

1. SMALLDML (見積コストが 1000 timeron より小さいすべての DML タイプ SQL 用)。
2. LOADUTIL (ロード・ユーティリティー用)。
3. ALLACTIVITY (すべてのデータベース・アクティビティー用)。

ALLACTIVITY は最後に評価されるワーク・クラスであり、最初の 3 つのワーク・クラスに対応していないデータベース・アクティビティーを網羅します。

このワーク・クラス・セットの作成用の DDL は次のとおりです。

```
CREATE WORK CLASS SET EXAMPLE
(WORK CLASS SMALLDML WORK TYPE DML FOR TIMERONCOST FROM 0 TO 999,
WORK CLASS LOADUTIL WORK TYPE LOAD,
WORK CLASS ALLACTIVITY WORK TYPE ALL)
```

EXAMPLESERVICECLASS というサービス・スーパークラスがあり、これに SMALLACTIVITY および OTHERACTIVITY という 2 つのサービス・サブクラスが含まれているとします。すべての小さなデータベース・アクティビティーが SMALLACTIVITY サービス・サブクラスで実行されるように、さらに (ロード・ユーティリティーを除く) 認識される他のすべてのデータベース・アクティビティーが OTHERACTIVITY サービス・サブクラスで実行されるようにシステムをセットアップしたいとします。ロード・ユーティリティーをその他のサービス・サブクラスには再マップせずに、代わりにデフォルトのサービス・サブクラスで実行したいものとします。

これらの目標を達成するために、EXAMPLESERVICECLASS サービス・スーパークラスに対してワーク・アクション・セット SERVICECLASSACTIONS をセットアップします。 SERVICECLASSACTIONS ワーク・アクション・セットには、以下のワーク・アクションが含まれます。

表 25. SERVICECLASSACTIONS ワーク・アクション・セット

ワーク・アクション	適用されるワーク・クラス	アクション
MAPDML	SMALLDML	SMALLACTIVITY サービス・サブクラスにマップします
COUNTLOAD	LOADUTIL	LOAD アクティビティーの数をカウントします

表 25. *SERVICECLASSACTIONS* ワーク・アクション・セット *(続き)*

ワーク・アクション	適用されるワーク・クラス	アクション
MAPOTHER	ALLACTIVITY	OTHERACTIVITY サービス・サブクラスにマップします

このワーク・アクション・セットを作成するための DDL は次のとおりです。

```
CREATE WORK ACTION SET SERVICECLASSACTIONS FOR SERVICE CLASS EXAMPLESERVICECLASS
USING WORK CLASS SET EXAMPLE
(WORK ACTION MAPDML ON WORK CLASS SMALLDML MAP ACTIVITY TO SMALLACTIVITY,
WORK ACTION COUNTLOAD ON WORK CLASS LOADUTIL COUNT ACTIVITY,
WORK ACTION MAPOTHER ON WORK CLASS ALLACTIVITY MAP ACTIVITY TO OTHERACTIVITY)
```

この構成を使用すると、すべての小さな DML は SMALLACTIVITY サービス・サブクラスの下で実行されます。 COUNTLOAD ワーク・アクションは、デフォルトのサービス・サブクラスの下で実行される LOADUTIL ワーク・クラスに適用されます。認識される他のすべてのデータベース・アクティビティーは、OTHERACTIVITY サービス・サブクラスの下で実行されます。

注: ALLACTIVITY ワーク・クラスが評価順序の先頭にあったとすれば、認識されるすべてのアクティビティーは OTHERACTIVITY サービス・サブクラスにマップされることになります。

ここで、データベースに対してワーク・アクション・セットを定義し、システム上で並行して実行可能なものを制御するしきい値を適用するとします。以下のワーク・アクションが含まれる DATABASEACTIONS というワーク・アクション・セットを作成することができます。このワーク・アクション・セットを作成するための DML は次のとおりです。

```
CREATE WORK ACTION SET DATABASEACTIONS FOR DATABASE USING WORK CLASS SET EXAMPLE
(WORK ACTION CONCURRENTSMALLDML ON WORK CLASS SMALLDML
WHEN CONCURRENTDBCOORDACTIVITIES > 1000 AND QUEUEDACTIVITIES > 10000
COLLECT ACTIVITY DATA STOP EXECUTION,
WORK ACTION CONCURRENTLOAD ON WORK CLASS LOADUTIL
WHEN CONCURRENTDBCOORDACTIVITIES > 2 AND QUEUEDACTIVITIES > 10
COLLECT ACTIVITY DATA STOP EXECUTION,
WORK ACTION CONCURRENTOTHER ON WORK CLASS ALLACTIVITY
WHEN CONCURRENTDBCOORDACTIVITIES > 100 AND QUEUEDACTIVITIES > 100
COLLECT ACTIVITY DATA STOP EXECUTION,
WORK ACTION MAXCOSTALLOWED ON WORK CLASS ALLACTIVITY
WHEN ESTIMATEDSQLCOST > 1000000 COLLECT ACTIVITY DATA STOP EXECUTION)
```

表 26. *DATABASEACTIONS* ワーク・アクション・セット

ワーク・アクション	適用されるワーク・クラス	しきい値タイプおよび値	アクション
CONCURRENTSMALLDML	SMALLDML	並行性が 1000 ステートメントまで。キューが 10 000 ステートメントまで。	• 実行の停止 • アクティビティー・データの収集
CONCURRENTLOAD	LOADUTIL	並行性が 2 オカレンスまで。キューが 10 オカレンスまで。	• 実行の停止 • アクティビティー・データの収集

表 26. DATABASEACTIONS ワーク・アクション・セット (続き)

ワーク・アクション	適用されるワーク・クラス	しきい値タイプおよび値	アクション
CONCURRENTOTHER	ALLACTIVITY	並行性が 100 アクティビティーまで。キューが 100 アクティビティーまで。	• 実行の停止 • アクティビティー・データの収集
MAXCOSTALLOWED	ALLACTIVITY	見積 SQL コストが 1 000 000 timeron まで。	• 実行の停止 • アクティビティー・データの収集

これらのワーク・アクションが適用されると、最大で 1000 個の小さい DML タイプの SQL ステートメント (SMALLDML ワーク・クラスのため) を一度に実行することができます。さらに、最大で 10 000 個のステートメントをキューに入れることができます。一度に実行できるロード・ユーティリティーのオカレンスは 2 つのみであり、最大 10 のオカレンスをキューに入れることができます。 LOAD でなく、小さい DML でもないアクティビティーは一度に 100 のみ実行できます。また、これらのアクティビティーのうち一度にキューに入れることができるのは 100 のみです。いずれの状態においても、キューのしきい値に違反している場合、データベース・アクティビティーを実行することはできず、エラー・メッセージが返されます。

さらに、MAXCOSTALLOWED ワーク・アクションが ALLACTIVITY クラスに適用されます。これは、1 000 000 timeron より大きい見積コストを持つデータベース・アクティビティー (つまり、DML および XQueries ステートメント) は実行できないことを意味します。 MAXCOSTALLOWED ワーク・アクションは ALLACTIVITY ワーク・クラスに適用されますが、このワーク・アクションは見積コストが 1 000 000 timeron より大きいデータベース・アクティビティーにのみ影響を与えます。このワーク・アクションは、見積コストを持たないアクティビティー (DDL など) には影響しません。

第 3 章 アクティビティー管理

データ・サーバー上で実行される作業を識別したら、リソースを割り当てて制御を実施することによりその作業をアクティブに管理できる状態にあります。

サービス・クラスでのリソース割り当て

サービス・クラスは、作業を実行できる実行環境を定義します。この実行環境は使用可能なリソースを割り振るほか、作業の実行許可を決定するしきい値を含むことができます。

すべての作業はサービス・クラスで実行されるので、ワークロードを使用して作業をサービス・スーパークラスに割り当てるか、ワークロード、REMAP ACTIVITY しきい値アクション、または MAP ACTIVITY ワーク・アクションを使用して、作業をサービス・スーパークラス内のサービス・サブクラスに割り当てます。ワークロードを定義する際、そのワークロードに関連付けられる作業が実行されるサービス・クラスを指示します。デフォルトでは、作業をデフォルトのユーザー・サービス・クラス (SYSDEFAULTUSERCLASS) にマップするデフォルトのユーザー・ワークロード (SYSDEFAULTUSERWORKLOAD) も存在します。ユーザー定義ワークロードを使用してユーザー定義サービス・クラスに明示的にマップされていない作業があれば、デフォルトのユーザー・サービス・クラスで実行されるようにするためです。

以下の図に示すとおり、サービス・クラスなしでは、要求を編成して認識可能な論理グループにすることはできません。

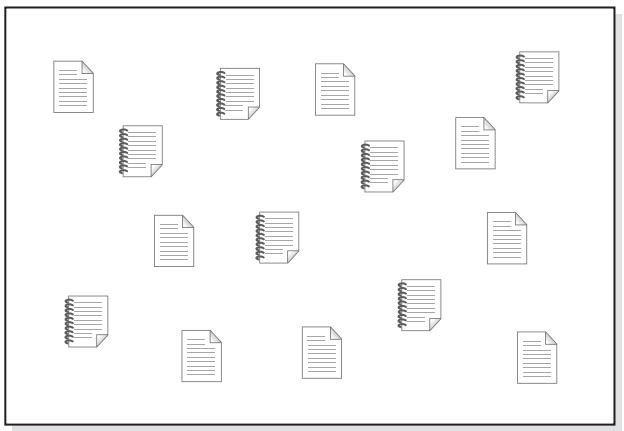


図 9. 未編成の作業

さまざまなサービス・スーパークラスを作成し、さまざまなタイプの作業用の実行環境を提供してから、該当する要求をサービス・スーパークラスに割り当てることができます。 2 つの別個の基幹業務 (金融および在庫) のアプリケーションを所有していると想定します。各基幹業務には、組織に対する責任を実行するための独自のアプリケーションがあるかもしれません。要求は、ワークロード管理目標にとって意味を成すカテゴリーに編成することができます。以下の図では、異なるサービ

ス・スーパークラスが異なる基幹業務に割り当てられています。

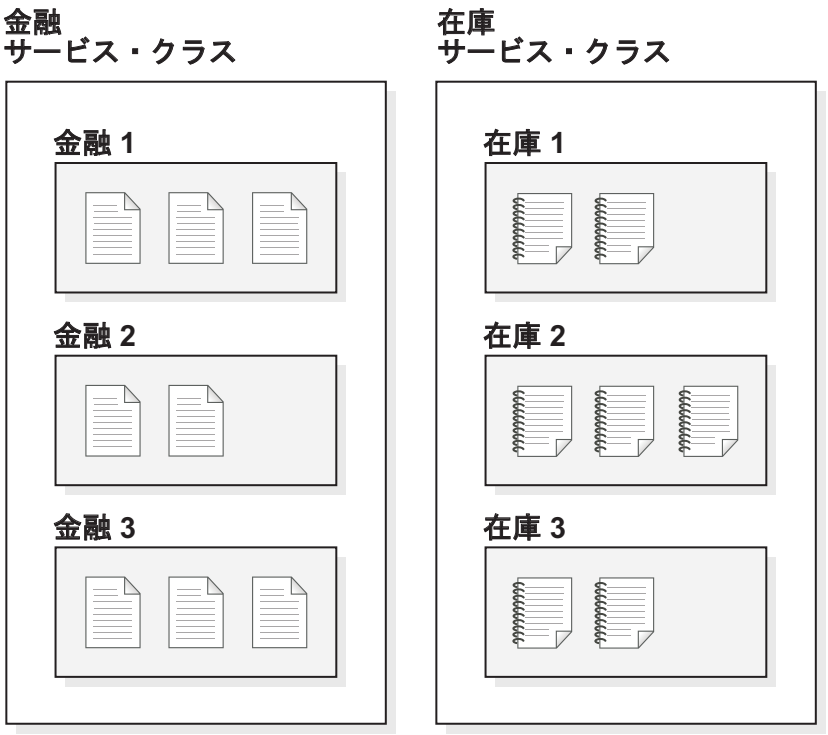


図 10. サービス・クラスによって編成される作業

前の図では、両方のサービス・スーパークラスにあるアクティビティーはさらに分割されています。サービス・クラスは、サービス・スーパークラスとその下のサービス・サブクラスという 2 層の階層を提供します。この階層により、実行環境をより複雑に分割して、実際のモデルをより適切にエミュレートすることができます。特に指定がない限り、サービス・サブクラスはサービス・スーパークラスからの特性を継承します。サービス・サブクラスは、サービス・スーパークラスの作業をさらに分割するために使用します。

優先順位付けとリソース制御

サービス・クラス・オブジェクトを作成または変更する際、次のようないくつかのリソース制御を定義できます。

表 27. サービス・クラスにより得られるリソース制御

制御	説明
Agent priority	この制御は、サービス・クラスで実行中のエージェント・スレッドのプロセッサー優先順位レベルを設定します。この優先順位は、このデータ・サーバーで実行中の他のスレッドおよびプロセスに対する相対 (デルタ) 優先順位として、オペレーティング・システムに渡されます。 **注:** アウトバウンド相関関係子の使用中は、この制御を設定できません。
プリフェッチ優先順位	この制御はプリフェッチ要求に対する優先順位を割り当て、プリフェッチ要求がデータ・サーバーによって扱われる順序に影響を与えます。

表 27. サービス・クラスにより得られるリソース制御 (続き)

制御	説明
バッファー・プール優先順位	この制御はサービス・クラスにバッファー・プール優先順位を割り当て、サービス・クラス内のアクティビティーによってフェッチされたページがスワップアウトされる優先度に影響を与えます。
アウトバウンド相関関係子	この制御を使用すると、ワークロードのリソースの一部を、AIX ワークロード・マネージャーや Linux ワークロード管理のようなオペレーティング・システムのワークロード・マネージャーで制御できます。タグは、エージェントから外部ワークロード・マネージャーに渡され、マネージャーで定義されたリソース・グループにマップされます。 DB2 ワークロード管理をオペレーティング・システムのワークロード・マネージャーと併せて使用すると、追加の制御を使用できます。AIX ワークロード・マネージャーを使用する場合は、サービス・クラスごとにプロセッサー・リソースの最小、最大、または相対シェアを設定することにより、各サービス・クラスに割り振られるプロセッサー・リソース量を制御できます。 Linux ワークロード管理を使用する場合は、Linux デフォルト・クラスに対する相対シェアを各サービスクラスに設定することにより、CPU リソースの量を制御できます。 **注:** エージェント優先順位の使用中は、この制御を設定できません。

サービス・サブクラス

サービス・スーパークラスは作業における最高位の層ですが、アクティビティーはサービス・サブクラスでだけ実行されます。各サービス・スーパークラスには、明示的に定義されたサブクラスに割り当てないアクティビティーを実行するための、デフォルトのサービス・サブクラスが定義されています。サービス・スーパークラスが作成されると、このデフォルトのサブクラスが作成されます。さらに作業を分離することが必要であれば、それに応じてサービス・クラス内に追加のサブクラスを作成することができます。ヒストグラム、および COLLECT ACTIVITY DATA、COLLECT AGGREGATE ACTIVITY DATA、COLLECT AGGREGATE REQUEST DATA オプションを除き、特に指定されていない限り、サービス・サブクラスはそのサービス・スーパークラスの属性を継承します。スーパークラスのリソースは、その中のすべてのサブクラスによって共有されます。

定義できるサブクラスは単一のレベルだけです (つまり、サブクラスはサービス・スーパークラスの下位にしか定義できず、別のサブクラスの下位には定義できません)。

以下の図は、ワークロードおよびサービス・クラスを使用したカスタム DB2 ワークロード管理構成の例です。

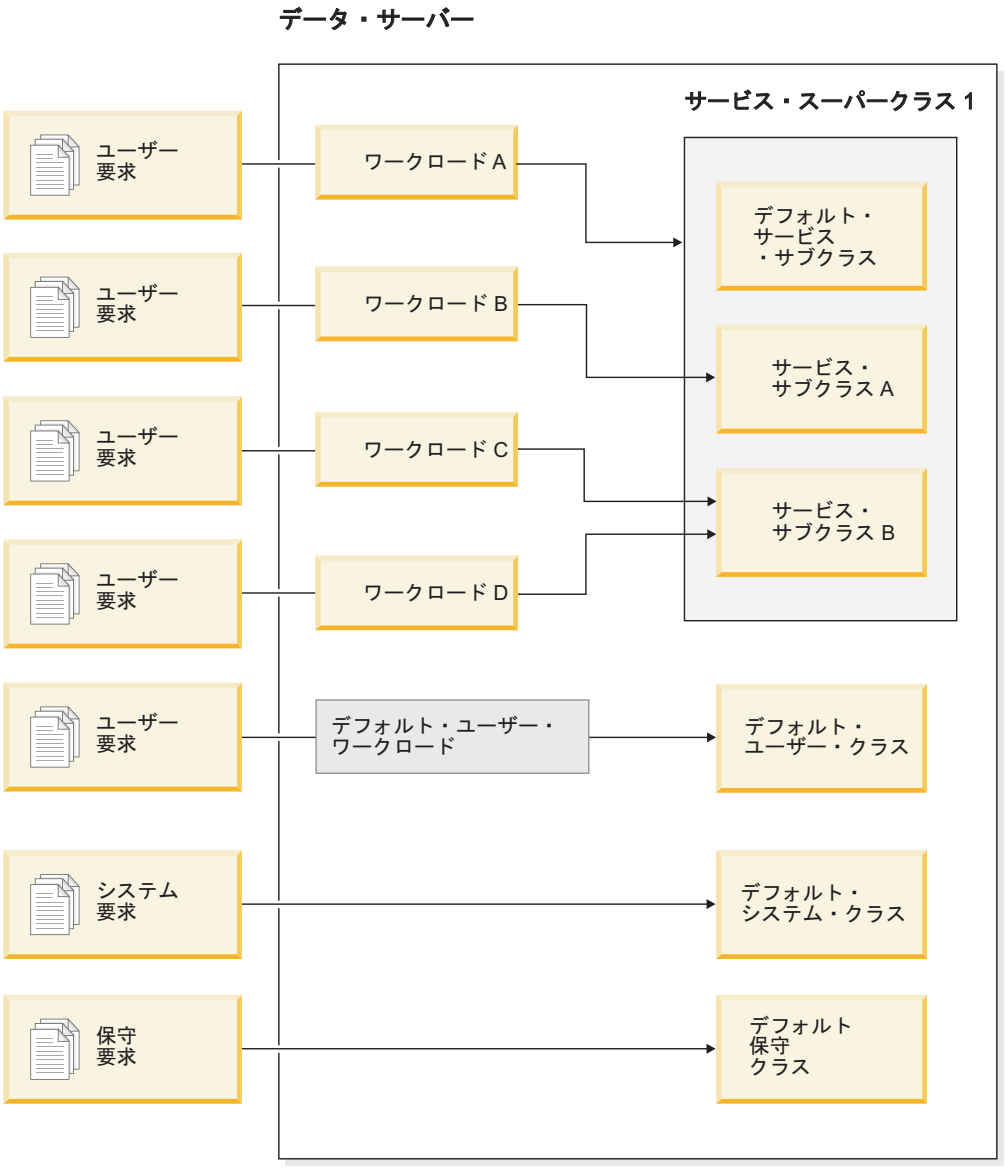


図 11. ワークロードおよびサービス・クラスを使用したカスタム DB2 ワークロード管理構成

ユーザー要求がデータ・サーバーに入ると、それは特定のワークロードに所属するものとして識別され、サービス・スーパークラスまたはサブクラスに割り当てられます。さらに、特別なデフォルトのシステム・サービス・クラス (SYSDEFAULTSYSTEMCLASS) の下で実行されるシステム要求 (例えばプリフェッチなど) や、デフォルトの保守サービス・クラス (SYSDEFAULTMAINTENANCECLASS) の下で実行される DB2 主導型の保守要求 (ヘルス・モニターからの自動 RUNSTATS など) もあります。

SYSCAT.SERVICECLASSES カタログ・ビューを照会して、サービス・クラスを表示できます。

デフォルトのサービス・スーパークラスおよびサブクラス

新規の各データベースまたはアップグレード後の各データベースには、事前定義された 3 つのデフォルトのサービス・スーパークラスが含まれます。それらは、デフォルトのユーザー・クラス、デフォルトの保守クラス、およびデフォルトのシステム・クラスです。

デフォルトのサービス・スーパークラスはいずれも、使用不可にしたりドロップしたりすることはできません。

デフォルトのサービス・スーパークラスはすべて、1 つのデフォルト・サービス・サブクラスとともに作成されます。デフォルトのサービス・スーパークラスの追加のサービス・サブクラスを作成することはできません。デフォルトのサービス・サブクラスは、次のように常に SYSDEFAULTSUBCLASS という名前で作成されます。

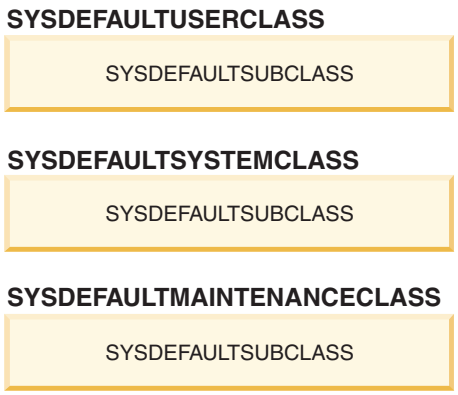


図 12. 2 層のサービス・クラス階層

デフォルトのサービス・スーパークラスへの接続によって発行される作業はすべて、そのサービス・スーパークラスのデフォルトのサービス・サブクラスで処理されます。

デフォルトのサービス・スーパークラスおよびそのデフォルトのサービス・サブクラスがドロップされるのは、データベースがドロップされる場合のみです。 DROP SERVICE CLASS ステートメントを使用してこれらをドロップすることはできません。

デフォルトのユーザー・サービス・スーパークラス (SYSDEFAULTUSERCLASS)
デフォルトでは、すべてのユーザー・アクティビティーが SYSDEFAULTUSERCLASS で実行されます。

デフォルトの保守サービス・スーパークラス (SYSDEFAULTMAINTENANCECLASS)
デフォルトの保守サービス・スーパークラスは、データベース保守および管理タスクを実行する内部 DB2 接続をトラッキングします。 DB2 非同期バックグラウンド・プロセス (ABP) エージェントからの接続は、このサービス・スーパークラスにマップされます。 ABP エージェントは、データベース保守タスクを実行する内部エージェントです。非同期索引のクリーンアップ (AIC) は ABP 主導タスクの一例です。ABP エージェントは、データ・

サーバー上でユーザー接続の数が増加すると、リソースの使用量およびサブエージェントの数を自動的に削減します。ユーザー接続によって発行されるユーティリティーは、通常のサービス・クラスを使用してマップされます。サービス・クラスのしきい値を SYSDEFAULTMAINTENANCECLASS にインプリメントすることはできません。

デフォルトの保守サービス・スーパークラスによってトラッキングされる内部接続には、以下のものが含まれます。

- ABP 接続 (AIC を含む)
- ヘルス・モニターによって開始されたバックアップ
- ヘルス・モニターによって開始された RUNSTATS
- ヘルス・モニターによって開始された REORG

デフォルトのシステム・サービス・スーパークラス (SYSDEFAULTSYSTEMCLASS)

デフォルトのシステム・サービス・スーパークラスは、システム・レベルのタスクを実行する内部 DB2 接続およびスレッドをトラッキングします。このサービス・スーパークラスに対してサービス・サブクラスを定義したり、ワークロードまたはワーク・アクションを関連付けたりすることはできません。さらに、サービス・クラスのしきい値を SYSDEFAULTSYSTEMCLASS にインプリメントすることはできません。デフォルトのシステム・サービス・スーパークラスによってトラッキングされる DB2 スレッドおよび接続には、以下のものが含まれます。

- ABP デーモン
- セルフチューニング・メモリー・マネージャー (STMM)
- プリフェッチャー・エンジン・ディスパッチ可能単位 (EDU) (db2pfchr)
- ページ・クリーナー EDU (db2pclnr)
- ログ読み取りプログラム EDU (db2loggr)
- ログ書き込みプログラム EDU (db2loggw)
- ログ・ファイル読み取りプログラム EDU (db2lfr)
- デッドロック検出機能 EDU (db2dlock)
- イベント・モニター (db2evm)
- イベント・モニター高速書き込みプログラム (db2fw)
- システム・レベルのタスクを実行する接続

アクティビティーからサービス・クラスへのマッピング

すべてのデータベース接続は、最初の作業単位の開始時にワークロードに割り当てられます。ワークロード・オカレンスが開始されると、ワークロード定義で指定されたサービス・クラス名に基づいて、そのワークロード・オカレンスの下で実行されているすべてのアクティビティーがサービス・クラスにマップされます。

接続がワークロード定義に定義された基準を満たしている場合、データ・サーバーはそのワークロード定義に接続を割り当てます。例えば、アプリケーション A からの接続はすべてワークロード定義 Alpha に属するように、アプリケーション B からの接続はすべてワークロード定義 Beta に属するように、ワークロード管理構成をセットアップすることができます。

ワークロード・オカレンスをサービス・スーパークラスに割り当てると、そのワークロード・オカレンスにサブミットされたアクティビティーを、ワーク・アクション・セットを使用してそのサービス・スーパークラスのユーザー定義サービス・サブクラスに再割り当てできます。

CREATE WORKLOAD ステートメントの SERVICE CLASS キーワードを指定することによって、ワークロードを使用してアクティビティーを接続からサービス・スーパークラスにマップできます。アクティビティーに当てはまるワーク・クラスまたはワーク・アクションがない場合、アクティビティーはサービス・スーパークラスのデフォルトのサービス・サブクラスで実行されます。さらに、CREATE WORKLOAD ステートメントの SERVICE CLASS キーワードで UNDER キーワードを指定することによって、ワークロードを使用してアクティビティーを接続からサービス・スーパークラスのサービス・サブクラスにマップすることもできます。この場合、接続は依然としてサービス・スーパークラスに属していますが、その接続から発行されたすべてのアクティビティーは、ワークロード定義で指定されたサービス・サブクラスに自動的にマップされます。

サービス・サブクラスにマップされて実行を開始したアクティビティーは、別のサービス・サブクラスに再マップ (しきい値によって) されない限り、そのサービス・サブクラスにとどまります。再マップはアクティビティーのリソース割り当てを変更できるプロセスで、アクティビティーを異なるサービス・サブクラスにマップします。ソースとターゲットの両方のサービス・サブクラスが、同じスーパークラス下に存在しなければなりません。再マップされたアクティビティーは、新しいサービス・サブクラスで実行を続けます。

以下の図では、接続、ワークロード、およびサービス・スーパークラスの間の関係を示しています。ワークロード A の定義を満たす接続は、サービス・スーパークラス 1 にマップされます。ワークロード B または C の定義を満たす接続はサービス・スーパークラス 2 にマップされます。ワークロード D の定義を満たす接続は SYSDEFAULTUSERCLASS サービス・スーパークラスにマップされます。

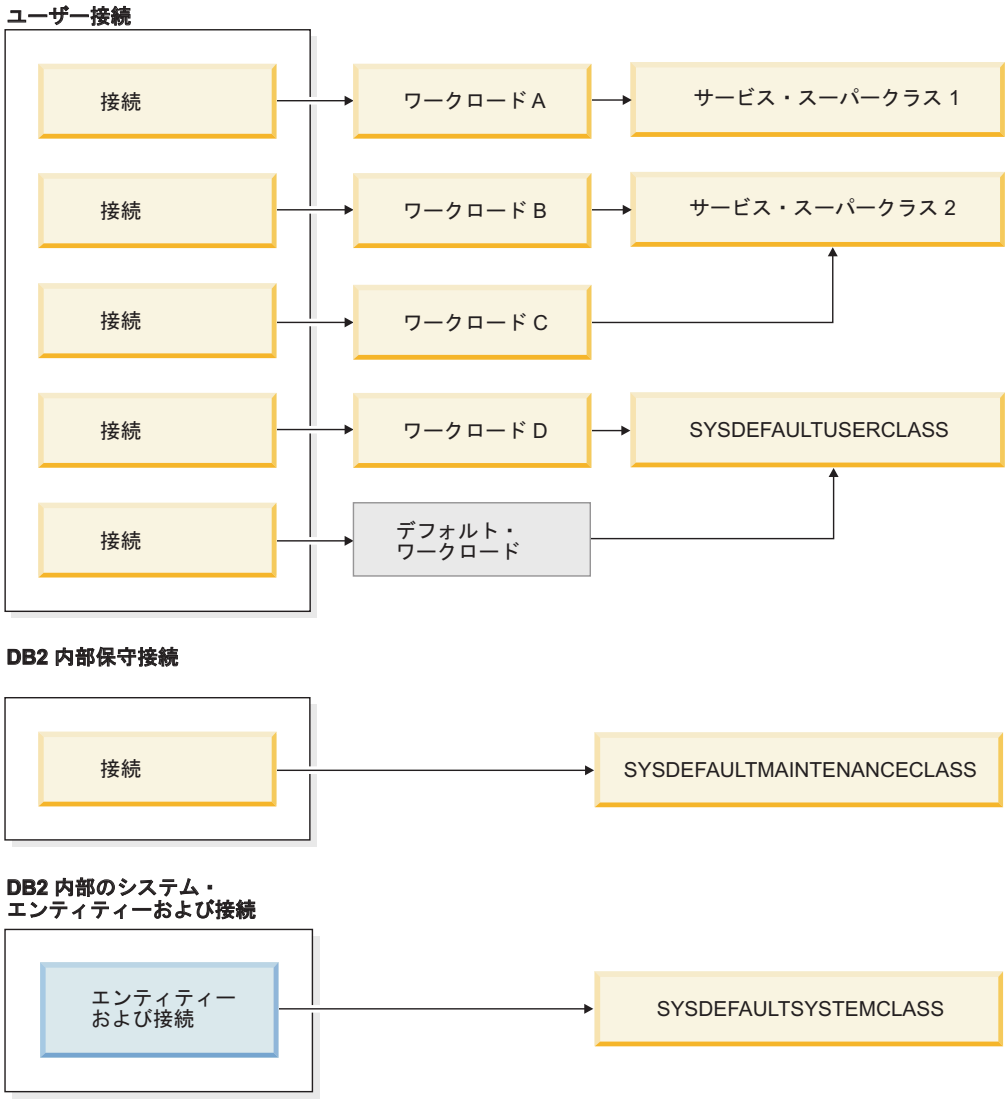


図 13. サービス・スーパークラスへのデータベース接続のマッピング

アクティビティーをさらに区別する

DB2 ワークロード・マネージャー構成がより複雑な場合、アクティビティー・タイプまたは他のアクティビティー属性に基づいてアクティビティーを別々に処理することも可能です。例えば、以下のアクションのいずれかを実行することができます。

- DML を DDL とは異なるサービス・サブクラスに配置する。
- 見積コストが 100 timeron 未満のすべての読み取りタイプ照会を、その他すべての読み取りタイプ照会とは異なるサービス・サブクラスに配置する。

より複雑な構成では、アクティビティーを接続からサービス・スーパークラスにマップするよう、ワークロードをセットアップすることができます。続いて、ワーク・アクション (サービス・スーパークラスに適用されるワーク・アクション・セットに入っている) を使用して、アクティビティーをそのタイプまたは属性に基づいて、サービス・スーパークラスの特定のサービス・サブクラスに再マップすることができます。

特に、MAP ACTIVITY ワーク・アクションが入っているワーク・アクション・セットをサービス・スーパークラスに適用することができるかもしれません。サービス・スーパークラスにマップされ、かつ MAP ACTIVITY ワーク・アクションに関連付けられているワーク・クラスと一致するすべてのアクティビティーは、ワーク・アクションによって指定されたサービス・サブクラスにマップされます。

ワークロード定義がそれ自体をサービス・サブクラスに関連付ける場合、そのワークロードを介してサブミットされるアクティビティーはどれも、親サービス・スーパークラスに割り当てられるワーク・アクション・セットによって影響を受けることがありません。

- ワークロードによって、アクティビティーをサービス・スーパークラスの 1 つのサービス・サブクラスにマップできる。
- 同じサービス・スーパークラス内の別のサービス・サブクラスへとアクティビティーをマップするワーク・アクションも、このアクティビティーに適用される。

ワークロードまたはワーク・アクションによってアクティビティーがサービス・サブクラスにマップされない場合、そのアクティビティーは、そのアクティビティーのサービス・スーパークラスのデフォルトのサブクラス (SYSDEFAULTSUBCLASS) にマップされます。

データベース・アクティビティーがそれぞれのサービス・スーパークラスおよびサービス・サブクラスにマップされている場合、特定のサービス・クラス内のすべてのアクティビティーを制御することができます。統計は、そのサービス・クラスのデータベース・アクティビティーをモニターするために使用できるサービス・クラス・レベルで使用可能です。

以下の図では、ワークロードを介してサービス・スーパークラスまたはサービス・サブクラスにマップされているデータベースへの要求を示します。ワーク・アクションを使用してアクティビティーをサービス・サブクラスにマップする方法については、111 ページの『ワーク・アクションとワーク・アクション・セット』を参照してください。

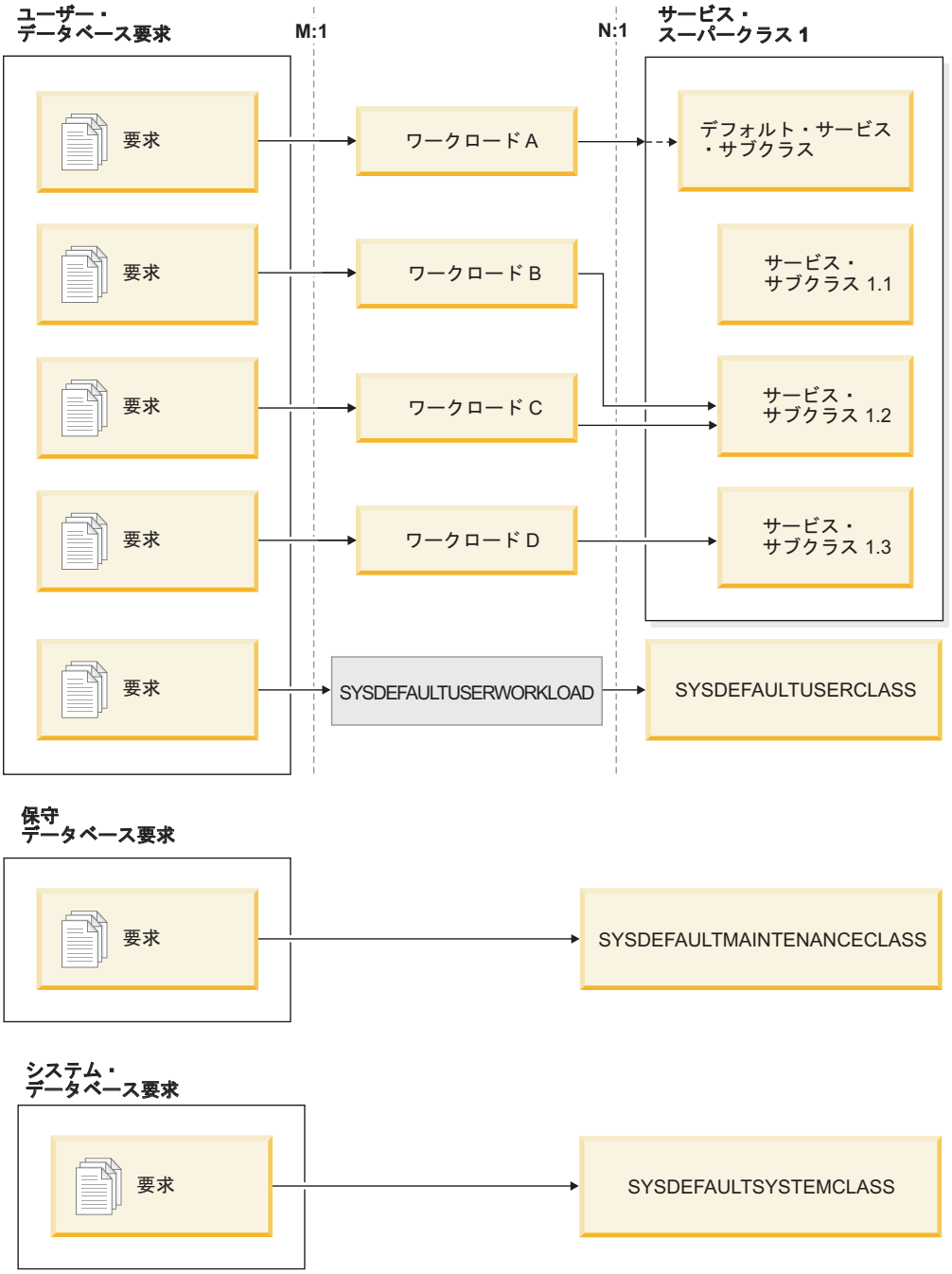


図 14. サービス・スーパークラスにマップされているデータベース接続

ユーザー定義のワークロード定義にマップしない接続は、デフォルトのユーザー・ワークロード定義 SYSDEFAULTUSERWORKLOAD にマップされます。デフォルトで、デフォルトのワークロード定義 (SYSDEFAULTUSERWORKLOAD) からの接続は、ユーザー要求に対するデフォルトのサービス・スーパークラスである SYSDEFAULTUSERCLASS サービス・スーパークラスにマップされます。SYSDEFAULTUSERWORKLOAD ワークロードは、別のサービス・クラスにマップするように変更することができます。内部 DB2 保守接続は、保守要求のデフォルトのサービス・スーパークラスである SYSDEFAULTMAINTENANCECLASS にマップされます。内部システム・エンティティーおよび接続は、システム・レベルの

タスクを実行する内部 DB2 接続およびスレッドのデフォルトのサービス・スーパークラスである SYSDEFAULTSYSTEMCLASS にマップされます。

サービス・クラスのエージェント優先順位

各 DB2 サービス・クラスを、データ・サーバー上のプロセッサー優先順位を制御するエージェント相対優先順位に関連付けることができます。この優先順位は、サービス・クラスで機能するすべてのエージェントに対して設定され、他のすべての DB2 エージェントのエージェント優先順位に対する相対的なものです。

サービス・クラスに対してエージェントの優先順位の値を指定しなかった場合、そのサービス・クラスのすべてのエージェントの優先順位は、他のすべての DB2 エージェントと同じになります。

DB2 サービス・クラスのエージェント優先順位の設定によって調整されるのは、サービス・クラスに入れられる新規作業用のエージェントの優先順位のみです。サービス・クラスで実行される非エージェント・スレッドは、指定されたエージェント優先順位の値を使用しません。 DB2 ワークロード管理は、 fenced モード・プロセス (FMP) 内で実行中の作業に対してサービス・クラスのエージェント優先順位を割り当てません。fenced プロシージャーではサービス・クラス内でそれらのロジックが実行されません。これらの fenced プロシージャーは DB2 FMP 内で実行され、この作業は DB2 エージェントによって実行されません。DB2 WLM は DB2 エージェントを制御します。

DB2 サービス・クラスを AIX ワークロード・マネージャーや Linux ワークロード管理などのオペレーティング・システムのワークロード・マネージャーと統合する場合、オペレーティング・システムのクラスに (プロセッサー・シェアとして) 使用するプロセッサー優先順位をオペレーティング・システムのワークロード・マネージャーを使用して指定し、この値を DB2 サービス・クラスの OUTBOUND CORRELATOR 値を介して DB2 サービス・クラスに継承させることができます。オペレーティング・システムのワークロード・マネージャーを使用して指定したプロセッサー優先順位は、DB2 サービス・クラスで実行されるエージェントの優先順位を制御します。この場合、サービス・クラスにエージェント優先順位が設定されていても無視されます。

注: エージェントの優先順位を使用して、システム上の作業サブセットによって使用される CPU 量を制限することはできません。競合するワークロードが CPU リソースを必要とする際、エージェントの優先順位によってそのワークロード間での CPU 処理に優先度が適用されます。しかし、エージェントの優先順位によって、競合するワークロード間で CPU リソースが明示的に割り振られたり管理されたりすることはありません。このため、競合するワークロードが要求する CPU リソースが等しくなく、より高い優先順位の作業よりも、より低い優先順位の作業が CPU リソースを消費する要求を多く行う場合、エージェント優先順位はあまり役に立ちません。しかし、競合するワークロードが等しく CPU リソースを要求しており、CPU 使用量を明示的に調整または制御するよりも、単に CPU 使用量の優先度を適用したい場合は、エージェント優先順位は効果的です。CPU 使用量に対してさらに明示的な制御を行う場合は、Linux WLM の統合または並行性制御を使用する方がより効果的です。

重要: 推奨されない **agentpri** データベース・マネージャー構成パラメーターを DB2 ワークロード管理と共に使用しないでください。この構成パラメーターを使用することで、DB2 インスタンス内のすべてのエージェントのプロセッサー絶対優先順位を固定値に設定できます。しかし、**agentpri** を使用してエージェントの絶対優先順位を設定すると、DB2 サービス・クラスのエージェント優先順位を設定して、またはオペレーティング・システムのワークロード・マネージャーを使用して、エージェントの相対優先順位を変更することができなくなります。 **agentpri** を設定した場合、サービス・クラスのエージェント優先順位とオペレーティング・システムのワークロード・マネージャーは、エージェントの優先順位に影響を与えません。

UNIX オペレーティング・システムと Linux の場合、有効な値は DEFAULT と -20 から 20 までです (SQLSTATE 42615)。負の値は相対的な優先順位が高いことを示します。正の値は相対的な優先順位が低いことを示します。

Windows オペレーティング・システムの場合、有効な値は DEFAULT と -6 から 6 までです (SQLSTATE 42615)。負の値は相対的な優先順位が低いことを示します。正の値になるほど相対的に高い優先順位を示します。

AIX オペレーティング・システムで、AGENT PRIORITY を使用してサービス・クラスのエージェントにより高い相対優先順位を設定するためには、インスタンスの所有者に CAP_NUMA_ATTACH 機能および CAP_PROPAGATE 機能が必要です。これらの機能を付与するには、root としてログオンし、次のコマンドを実行します。

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

Solaris 10 以降で、AGENT PRIORITY を使用してサービス・クラスのエージェントにより高い相対優先順位を設定するためには、インスタンス所有者に proc_priocntl 特権が必要です。この特権を付与するには、root としてログオンし、次のコマンドを実行します。

```
usermod -K defaultpriv=basic,proc_priocntl db2user
```

この例では、ユーザー db2user のデフォルトの特権セットに proc_priocntl が追加されます。

また、DB2 を Solaris の非グローバル・ゾーンで実行する場合には、proc_priocntl 特権をそのゾーンの制限特権セットに追加しなければなりません。この特権をゾーンに付与するには、root としてログオンし、次のコマンドを実行します。

```
global# zonecfg -z db2zone
zonecfg:db2zone> set limitpriv="default,proc_priocntl"
```

この例では、ゾーン db2zone の制限特権セットに proc_priocntl が追加されます。

Solaris 9 では、エージェントの相対優先順位を引き上げるための DB2 の機能はありません。サービス・クラスのエージェント優先順位を使用するには、Solaris 10 以降にアップグレードしてください。

サービス・クラスのプリフェッチ優先順位

プリフェッチャーは、ディスクからデータを検索し、このデータをバッファー・プールに保管して、エージェントがデータに即時にアクセスできるようにします。各サービス・スーパークラスおよびサブクラスが異なるプリフェッチ優先順位を持つように割り当てることができます。

エージェントは、先読み要求をデータベース・プリフェッチ・キューに送信します。プリフェッチャーは、これらの先読み要求をキューから取り、そのデータをバッファー・プールに取り出します。特定のデータを必要とする場合、エージェントは最初にバッファー・プールを検査して、データが使用可能かどうかを確認します。使用可能でない場合、エージェントはディスクからデータを取り出します。プリフェッチャーは、長い時間がかかるディスク入出力操作を実行し、エージェントを解放して処理を並行して実行できるようにします。

サービス・クラスにルーティングされるすべての接続において、プリフェッチ要求はサービス・クラスに割り当てられたプリフェッチ優先順位に従って処理されます。各サービス・クラスは、高、中、または低の 3 つのプリフェッチ優先順位のいずれかと関連付けることができます。 CREATE または ALTER SERVICE CLASS ステートメントのいずれかで、PREFETCH PRIORITY キーワードを使用して、サービス・クラスのプリフェッチ優先順位を指定します。

サービス・スーパークラスに DEFAULT を指定すると、サービス・スーパークラスのプリフェッチ優先順位は中に設定されます。サービス・スーパークラスのすべてのサービス・サブクラスで異なるプリフェッチ優先順位を指定できますが、サービス・サブクラスに対してデフォルトのプリフェッチ優先順位を使用する場合、サービス・サブクラスはそのサービス・スーパークラスからプリフェッチ優先順位の設定を継承します。

高優先順位プリフェッチ要求は、中優先順位プリフェッチ要求より前に処理され、中優先順位プリフェッチ要求は、低優先順位プリフェッチ要求より前に処理されます。プリフェッチ優先順位はプリフェッチ要求の処理順序に影響を与えますが、それぞれの処理速度には影響しません。

サービス・クラスのバッファー・プール優先順位

サービス・サブクラスのバッファー・プール優先順位を設定すると、ある特定のサービス・クラス内のアクティビティーが占有する可能性があるバッファー・プール内のページの比率を調整できます。この設定で、そのサービス・クラス内のアクティビティーのスループットとパフォーマンスを改善できます。

各 DB2 サービス・クラスをバッファー・プール相対優先順位に関連付けることができます。バッファー・プール相対優先順位は、サービス・クラス内のアクティビティーがバッファー・プールにフェッチしたページがスワップアウトされる優先度を制御します。 バッファー・プール優先順位を高くすると、特定のサービス・クラスのエージェントが使用中のページの比率が潜在的に大きくなります。

バッファー・プール優先順位を指定しない場合、または BUFFERPOOL PRIORITY DEFAULT を指定した場合、サービス・クラスに割り当てられるバッファー・プール優先順位は DEFAULT になります。サービス・スーパークラスの場合、DEFAULT は値 LOW にマップされます。サービス・サブクラスの場合、

DEFAULT は親サービス・スーパークラスのバッファー・プール優先順位の値にマップされます。デフォルトのすべてのサブクラスのバッファー・プール優先順位は DEFAULT であり、これを変更することはできません。

以前のバージョンの DB2 からアップグレードした場合、既存のサービス・クラスのバッファー・プール優先順位は DEFAULT に設定されます。

バッファー・プール優先順位の設定による利点

バッファー・プールで発生している競合が妥当な量であれば、サービス・クラスのバッファー・プール優先順位を設定することによるパフォーマンス上の利点を実感しやすいでしょう。最大の恩恵が得られるのは、全体ヒット率が 85% 以下のバッファー・プール競合の場合と思われます。全体ヒット率が 90% を超えている場合は、対象にするほどのバッファー・プール競合は発生していない可能性が高いので、バッファー・プール優先順位を設定しても、たいていの場合、ほとんど利点はありません。どんな利点を実感するかは、データ・サーバーが実行するワークロードのタイプに依存します。

いくつかのワークロードでは、先行ページ・クリーニングもオンにすると、バッファー・プール優先順位の設定の効果が高まります。これは、バッファー・プール優先順位の設定は非ダーティー・ページに対してのみ有効であり、ダーティー・ページをディスクに書き出すことに関しては、先行ページ・クリーニングがよりアグレッシブだからです。先行ページ・クリーニングをオンにするのは、パフォーマンス上の利点が得られる場合のみにしてください。

非同期ページ・クリーニング (別名クラシック・ページ・クリーニング) を使用する場合は、**chngpgs_thresh** データベース構成パラメーターの設定値を小さくすると、同様にバッファー・プール優先順位設定の効果が上がるでしょう。このパラメーターの値が小さいと、バッファー・プール内に十分な量のクリーン・ページができるからです。

妥当な量のプリフェッチが行われている場合は、プリフェッチ優先順位の設定の有無にかかわらず、プリフェッチの効果がバッファー・プール優先順位の設定の効果を上回る可能性があります。例えば、バッファー・プール優先順位の高いサービス・クラスを定義したとします。そのサービス・クラスでは、わずかのプリフェッチしか行われません。バッファー・プール優先順位は低いがアクティビティーがかなりの量のプリフェッチを実行するサービス・クラスと比べると、このバッファー・プール優先順位設定の効果は小さい可能性があります。プリフェッチの利点により、バッファー・プール優先順位の低いサービス・クラス内のアクティビティーが、バッファー・プール優先順位の高いサービス・クラス内のアクティビティーよりパフォーマンスが高いこともあるということです。とはいえ、こうした環境でも、バッファー・プール優先順位を設定することは、ワークロード管理ストラテジーの補足になることに変わりはないので、バッファー・プール優先順位を設定するようにしてください。

サービス・クラスにおける接続およびアクティビティーの状態

サービス・クラスは、サービス・クラスごとに接続統計を収集します。あるサービス・クラスにどの接続およびアクティビティーがあるのか、またその接続またはアクティビティーの状態を表示することができます。

接続の状態

サービス・クラスにおける接続の状態として、以下が考えられます。

CONNECTED
この接続はデータベースに正常に接続されていますが、まだそのワークロードおよびサービス・スーパークラスには関連付けられていません。

DECOUPLED
この接続にはコーディネーター・エージェントが割り当てられていません (コンセントレーターの場合)。

DISCONNECTPEND
この接続はデータベースから切断中です。

FORCED
この接続は強制的に切断されました。

INTERRUPTED
この接続は中断されました。

MAPPED
この接続はワークロードにマップされていて、サービス・スーパークラスに加わりました。この接続はアクティビティーをサブミットして実行することができるようになりました。

QUEUED
この接続コーディネーター・エージェントは、CONCURRENTDBCOORDACTIVITIES しきい値などのキューイングを提供する、DB2 接続またはアクティビティーしきい値によってキューに入れられています。マルチメンバー・データベース環境では、この状態は、コーディネーター・エージェントがしきい値チケットを取得するために別のメンバーに対して RPC を行ったものの、まだ応答を受け取っていないことを示している可能性があります。

TRANSIENT
この接続は、接続しきい値に達したサービス・クラスに加わろうとしています。この接続はサービス・クラスに加わるためのキューに入れられています。サービス・クラスが接続しきい値に違反していなければ、その接続はサービス・クラスに加わります。過渡状態の接続は、アクティビティーをサブミットして実行することができません。

TERMINATING
この接続は、クライアントからの接続リセットを受け取ったか、強制終了またはエラー状態のために終了中です。

UOWEXEC
この接続は要求の処理中です。

UOWWAIT
この接続は、クライアントからの要求の待機中です。

アクティビティーの状態

サービス・クラスにおけるアクティビティーの状態として、以下が考えられます。

CANCEL_PENDING
アクティビティーの要求をアクティブに処理しているエージェントを持たないアクティビティーをキャンセルした場合、アクティビティーは CANCEL_PENDING 状態に置かれ、次に受信される要求でキャンセルされます。

EXECUTING
このアクティビティーは実行中です。

IDLE アクティビティーの要求をアクティブに処理しているエージェントがありません。

INITIALIZING
このアクティビティーは作成され、実行のために準備中です。

QUEUED
このアクティビティーは、データベース・レベルまたはサービス・クラス・レベルの並行性しきい値のために実行できません。このアクティビティーは実行が許可されるまでキューに入れられます。

TERMINATING
このアクティビティーは終了します。

UNKNOWN
アクティビティーの状態は不明です。

サービス・クラスによって管理されないシステム・レベルのエンティティー

サービス・クラスは、データベース・レベルでオブジェクトをモニターおよび制御するために使用されます。ただし、すべての DB2 エンティティーがデータベースで直接機能するわけではありません。

サービス・クラスはデータベース内で機能し、データベースのカタログ表に保管されるため、データベースで機能しないエンティティーをサービス・クラスによって管理することはできません。システム・コントローラーおよびヘルス・モニター・デーモンなどのインスタンス・レベルのエンティティーは、インスタンス・レベルで機能し、他のデータベースと直接関連付けられることはありません。インスタンス・アタッチメントおよびゲートウェイ接続を実行するエージェントも、サービス・クラスによって管理されません。インスタンス・アタッチメント・エージェントおよびゲートウェイ・エージェントはデータベース内では機能しないため、これらはサービス・クラスによって管理されません。

以下のリストは、データベース内で機能しないエンティティーの部分リストであり、サービス・クラスによって管理されません。

- DB2 システム・コントローラー (db2sysc)
- IPC リスナー (db2ipccm)
- TCP リスナー (db2tcpcm)
- FCM デーモン (db2fcms、db2fcmr)
- DB2 再同期エージェント (db2resync)
- アイドル・エージェント (データベースとの関連がないエージェント)

- インスタンス・アタッチメント・エージェント
- ゲートウェイ・エージェント
- その他すべてのインスタンス・レベルの EDU

サービス・クラスの作成

DDL ステートメント CREATE SERVICE CLASS を使用して、サービス・スーパークラスとその下のサービス・サブクラスを作成します。

始める前に

サービス・クラスを作成するためには、WLMADM または DBADM 権限が必要です。

その他の前提条件について、以下のトピックも参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』
- 559 ページの『付録 A. 一般的な命名規則』

手順

サービス・クラスを作成するには、次のようにします。

1. CREATE SERVICE CLASS ステートメントに、以下に挙げるサービス・クラスのプロパティーを 1 つ以上指定します。
   - サービス・クラスの名前を指定。

     注: サービス・クラスの名前は、一度設定すると変更することができません。
     - サービス・スーパークラスを作成している場合は、データベース内のすべてのサービス・スーパークラスの間で固有の名前を使用する必要があります。

       サービス・スーパークラスが作成されると、それに関連付けられるデフォルトのサービス・サブクラスが自動的に作成されます。他のサービス・サブクラスは、親サービス・スーパークラスが作成されてからでなければ作成できません。
     - サービス・サブクラスを作成している場合は、サービス・スーパークラス内のすべてのサービス・サブクラスの間で固有の名前を使用する必要があります。サービス・サブクラスをサービス・スーパークラスと同じ名前にすることはできません。
   - サービス・サブクラスを作成している場合は、親サービス・スーパークラスの名前を指定。特定のサービス・スーパークラスの下に作成されたサービス・サブクラスは、別のサービス・スーパークラスには関連付けることができません。
   - サービス・クラスのバッファー・プールに対する優先順位を指定。サービス・クラスのアクティビティーによってフェッチされているページがどのようにスワップアウトされるかに影響を与えます。サービス・スーパークラスの場合、DEFAULT の値は内部的に LOW にマップされます。 DEFAULT に設定されるサービス・サブクラスは、その親スーパークラスからバッファー・プール優先順位を継承します。

- プリフェッチ優先順位を指定。サービス・クラス内のエージェントがプリフェッチ要求をサブミットできる優先順位を指定できます。指定された値に応じて、プリフェッチ要求は優先順位が HIGH、MEDIUM、LOW のプリフェッチ・キューにルーティングされます。デフォルトのプリフェッチ優先順位は MEDIUM です。
- DB2 サービス・クラスを AIX クラスまたは Linux クラスに関連付ける場合は、OUTBOUND CORRELATOR の相関関係子として使用する string-constant を指定します。 NULL 値は、オペレーティング・システムのワークロード・マネージャーの関連付けがないことを示します。

  OUTBOUND CORRELATOR が設定されると、次のアクティビティーが開始されるときは、DB2 サービス・クラス内のすべてのスレッドが OUTBOUND CORRELATOR を使用してオペレーティング・システムのワークロード・マネージャーに関連付けられます。

  サービス・サブクラスで OUTBOUND CORRELATOR が NONE に設定されていて、関連するサービス・スーパークラスに OUTBOUND CORRELATOR が指定されている場合、サービス・サブクラスは親サービス・スーパークラスで指定されている OUTBOUND CORRELATOR を継承します。
- 収集するアクティビティー・データを指定。アクティビティー・データの収集が使用可能になっている場合、アクティビティーに関する情報は、アクティビティーの終了時にコーディネーター・メンバーから該当するイベント・モニターに送信されます。実行されたステートメント、そのコンパイル環境、および適用可能な入力データ値に関する情報などのデータをイベント・モニターに書き出すことができます。アクティビティー・データを収集しないように指定することもできます。デフォルトでは、アクティビティー・データは収集されません。
- 収集された集約アクティビティー情報を指定。サービス・クラスに使用される集約アクティビティー情報は、CREATE SERVICE CLASS 操作がコミットされた後にのみ変更されます。
- 指定したサービス・スーパークラスに関連付けられている接続によってサブミットされる要求について収集する、要求メトリックのタイプ。デフォルトでは、ワークロードに関連付けられているアクティビティーの基本メトリックが常に収集されます。
- サービス・サブクラスがそのヒストグラムのテンプレートとして使用するヒストグラム・テンプレート。指定されるヒストグラム・テンプレートは、SYSCAT.HISTOGRAMTEMPLATEUSE ビューに反映されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、292 ページの『ワークロード管理のヒストグラム』を参照してください。
- サービス・クラスを使用可能にするか使用不可にするかを指定。
  - サービス・クラスが使用可能として作成された場合 (デフォルト) は、そのサービス・クラスに接続とアクティビティーをマップすることができます。サービス・クラスが使用不可として作成された場合は、そのサービス・クラスへの新規接続およびアクティビティーのマップが拒否されます。
  - サービス・スーパークラスを使用不可として作成すると、そのサービス・スーパークラスに関連付けられているすべてのサービス・サブクラスは、

SYSCAT.SERVICECLASSES ビューを照会したときに使用可能として表示されていたとしても、使用不可の動作をします。

2. 変更をコミットします。 変更をコミットすると、サービス・クラスが SYSCAT.SERVICECLASSES ビューに追加されます。

サービス・クラスの変更

サービス・クラスの定義を変更する場合は、ALTER SERVICE CLASS ステートメントを使用します。

始める前に

サービス・クラスを変更するためには、SQLADM、WLMADM、または DBADM 権限が必要です。 COLLECT 節以外の節を指定するには、許可 ID に WLMADM または DBADM 権限が組み込まれている必要があります。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

このタスクについて

既にリソースを獲得済みで、実行中のアクティビティーは、通常、ALTER ステートメントの影響を受けません。これらのアクティビティーは、獲得したリソースを保持し、完了まで実行されます。(ALTER SERVICE CLASS ステートメントを使用して行われるサービス・クラスの CPU シェアまたは CPU リミットの変更は、既に実行中のアクティビティーの場合でも直ちに有効になることに注意してください。) ただし、ALTER SERVICE CLASS 操作の途中でリモート・メンバーにサブエージェント要求が送信されると、コーディネーター・エージェントとサブエージェントから見えるサービス・クラス定義に違いが発生する可能性があります。次の例について考えてみます。この例では、サービス・クラスのプリフェッチ優先順位が初めは MEDIUM に設定されています。

表 28. コーディネーター・エージェントとサブエージェントの間での、変更済みサービス・クラスの表示の違い

イベントの順序	接続 1	接続 2
1	コーディネーター・エージェントがリモート・メンバーに要求を送信する (サービス・クラスのプリフェッチ優先順位は事前に MEDIUM に設定されていた)	
2		ALTER SERVICE CLASS が発行され、プリフェッチ優先順位が HIGH に設定される
3		COMMIT が発行される (変更されたサービス・クラスのプロパティーはカタログ・メンバーでコミットされ、すべてのデータベース・メンバーでメモリーにロードされる)

表 28. コーディネーター・エージェントとサブエージェントの間での、変更済みサービス・クラスの表示の違い (続き)

イベントの順序	接続 1	接続 2
4	リモート・サブエージェントが要求を受け取る。この時点になって、サブエージェントはサービス・クラス定義の新しいプリフェッチ優先順位 HIGH を設定する	

上の表で説明した状態は一時的な状態であり、この状態の影響を受けるのは ALTER SERVICE CLASS 操作中にサブエージェント要求を発行する接続だけです。新しい接続はすべて、プリフェッチ優先順位が HIGH になった更新後のサービス・クラス定義を認識します。

手順

サービス・クラスを変更するには、次のようにします。

1. ALTER SERVICE CLASS ステートメントに、以下に挙げるサービス・クラスのプロパティーを 1 つ以上指定します。
   - サービス・クラスを使用可能にするか使用不可にするかを指定。使用可能のサービス・クラスを使用不可に変更した場合、既存の接続やアクティビティーはサービス・クラスに残り、使用不可になる前に割り振られたリソースを完了まで使用します。サービス・クラスに送られる作業がシステムを圧迫している場合や、サービス・クラスに送られるすべての作業を拒否したい場合には、サービス・クラスを使用不可にできます。

     サービス・スーパークラスが使用不可に設定されると、以下のことが起こります。

     a. サービス・スーパークラスが使用不可になります。

     b. そのスーパークラスのサービス・サブクラスが使用不可になります。

     サービス・サブクラスは、親サービス・スーパークラスが使用不可になっている間だけ使用不可になります。サービス・スーパークラスが使用可能になると、サービス・サブクラスは、カタログ表で定義された以前の状態に戻ります。

     サービス・サブクラスが使用不可になっても、親サービス・スーパークラスは影響を受けません。また、同じサービス・スーパークラスに関連付けられている他のサービス・サブクラスも影響を受けません。

     デフォルトのサービス・サブクラスは、明示的に使用不可にすることはできません。デフォルトのサービス・サブクラスで新しい要求が実行されないようにするためには、関連付けられているサービス・スーパークラスを使用不可にする必要があります。
   - サービス・クラスのエージェント優先順位を指定。エージェント優先順位が DEFAULT に設定されている場合、サービス・クラス内のエージェントには、オペレーティング・システムがすべての DB2 スレッドに割り当てるのと同じ優先順位が割り当てられます。AGENT PRIORITY パラメーターに DEFAULT

以外の値を設定した場合は、デフォルトの優先順位に、次のアクティビティー開始時点の設定値を加えた値に等しい優先順位が、エージェント・スレッドの優先順位として設定されます。例えば、デフォルトの優先順位が 20 でエージェント優先順位を -10 に設定した場合、結果的にエージェントの優先順位は 20 + (-10) = 10 に設定されます。

SYSCAT.SERVICECLASSES カタログ・ビューの DEFAULT のエージェント優先順位は、-32768 と表されます。

Linux および UNIX では、有効な値は、-20 から 20 です (負の値は、相対的により高い優先順位を示します)。 Windows オペレーティング・システムでは、有効な値は、-6 から 6 です (負の値は、相対的により低い優先順位を示します)。

- プリフェッチ優先順位を指定。サービス・クラス内のエージェントがプリフェッチ要求をサブミットできる優先順位を指定できます。指定された値に応じて、プリフェッチ要求は優先順位が HIGH、MEDIUM、LOW のプリフェッチ・キューにルーティングされます。デフォルトのプリフェッチ優先順位は MEDIUM です。プリフェッチ要求がサブミットされた後にプリフェッチ優先順位が変更された場合、その要求の優先順位は変更されません。
- サービス・クラスのアクティビティーによってフェッチされるページがスワッピングされる可能性に影響を与えるサービス・クラスのバッファー・プール優先順位を指定。サービス・スーパークラスの場合、DEFAULT の値は内部的に LOW にマップされます。 DEFAULT に設定されるサービス・サブクラスは、その親スーパークラスからバッファー・プール優先順位を継承します。
- DB2 サービス・クラスを AIX クラスまたは Linux クラスに関連付ける場合は、OUTBOUND CORRELATOR の相関関係子として使用する string-constant を指定します。 NULL 値は、オペレーティング・システムのワークロード・マネージャーの関連付けがないことを示します。

  OUTBOUND CORRELATOR が NULL 以外の値から NULL 値に変更されると、次のアクティビティーが開始されるときに、DB2 サービス・クラス内のすべてのスレッドでオペレーティング・システムのワークロード・マネージャーとの関連付けが解除されます。

  サービス・サブクラスで OUTBOUND CORRELATOR が NONE に設定されていて、関連するサービス・スーパークラスに OUTBOUND CORRELATOR が指定されている場合、サービス・サブクラスは親サービス・スーパークラスで指定されている OUTBOUND CORRELATOR を継承します。

  サービス・スーパークラスで OUTBOUND CORRELATOR を使用する場合は、サービス・スーパークラスのエージェント優先順位を DEFAULT に設定する必要があります。

  サービス・サブクラスで (サービス・サブクラス定義の一部として明示的に、またはサービス・スーパークラスから継承することによって暗黙的に) OUTBOUND CORRELATOR を使用する場合は、サービス・サブクラスのエージェント優先順位を DEFAULT に設定する必要があります。

- 指定したサービス・サブクラス用として望ましい TEMPORARY 表スペースのリストに SYSTEM TEMPORARY 表スペースを追加、またはドロップすることを指定します。
- 収集するアクティビティー・データを指定。アクティビティー・データの収集が使用可能になっている場合、アクティビティーに関する情報は、アクティビティーの終了時にコーディネーター・メンバーから該当するイベント・モニターに送信されます。実行されたステートメント、そのコンパイル環境、および適用可能な入力データ値に関する情報などのデータをイベント・モニターに書き出すことができます。アクティビティー・データを収集しないように指定することもできます。デフォルトでは、アクティビティー・データは収集されません。
- 収集された集約アクティビティー情報を指定。サービス・クラスに使用される集約アクティビティー情報は、ALTER SERVICE CLASS 操作がコミットされた後にのみ変更されます。
- 指定のサービス・スーパークラスの下のサブクラスにマップされた接続がサブミットする要求のモニター要求メトリックス収集レベル。サービス・スーパークラスの下で実行される要求に対する効率的な収集設定値は、サービス・クラス収集レベルと **mon_req_metrics** データベース構成パラメーターの両方を組み合わせたものであることに注意してください。
- COLLECT AGGREGATE ACTIVITY DATA を使用した集約アクティビティー・データ収集、または COLLECT AGGREGATE REQUEST DATA を使用した集約要求データ収集を使用可能にしているサービス・サブクラスが使用するヒストグラム・テンプレートを変更するかどうか。サービス・サブクラスが使用するヒストグラム・テンプレートを更新すると、サービス・クラスまたはワーク・アクションが参照するヒストグラム・テンプレートを表示する SYSCAT.HISTOGRAMTEMPLATEUSE ビューの対応する行が更新されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、 292 ページの『ワークロード管理のヒストグラム』を参照してください。

2. 変更をコミットします。 変更をコミットすると、サービス・クラスが SYSCAT.SERVICECLASSES ビューで更新されます。

サービス・クラスのドロップ

DDL ステートメント DROP SERVICE CLASS を使用してサービス・クラスをドロップします。

始める前に

サービス・クラスをドロップするためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

デフォルトのサービス・スーパークラス (SYSDEFAULTUSERCLASS、SYSDEFAULTMAINTENANCECLASS、 SYSDEFAULTSYSTEMCLASS) およびそれに関連付けられているサービス・サブクラスはドロップできません。デフォルトのサービス・スーパークラスや関連するサービス・サブクラスをドロップするには、データベースをドロップします。

以下の条件のいずれかに該当する場合、定義したサービス・クラスはドロップできません。

- 使用可能になっている
- ユーザー定義のサービス・サブクラスが含まれている
- 何らかのワークロード、ワーク・アクション、またはしきい値によって参照されている
- ワークロード・オカレンスによって参照されている
- 現時点で何らかの接続またはアクティビティーがそのサービス・クラスにマップされている
- サービス・クラスが REMAP ACTIVITY アクションのターゲットとして設定されている

手順

サービス・クラスをドロップするには、次のようにします。

1. サービス・クラスを指さないようにワークロードを変更するか、あるいはワークロード定義を使用不可にします。あるいは、DROP WORKLOAD ステートメントを使用して、サービス・クラスに関連付けられているワークロードをすべてドロップします。各ワークロードをドロップした後、COMMIT ステートメントを発行します。

   サービス・クラスで既に実行中のアクティビティーは、継続して実行されます。WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用して、サービス・クラスに現在マップされているエージェントをリストできます。これらのアクティビティーを完了させることを希望しない場合は、表関数によって戻されたアプリケーション ID と **FORCE APPLICATION** コマンドを使用して、アプリケーションをデータベースから強制終了させることができます。
2. ドロップするサービス・クラスに関連付けられている適用可能なワーク・アクションをすべてドロップします。
   - サービス・スーパークラスをドロップしている場合で、そのサービス・スーパークラスにワーク・アクション・セットが関連付けられている場合は、ワーク・アクション・セットを使用不可にしてドロップしてください。ワーク・アクション・セットを使用不可にするには ALTER WORK ACTION SET ステートメントを、ワーク・アクション・セットをドロップするには DROP WORK ACTION SET ステートメントを使用します。ワーク・アクション・セットをドロップした後、COMMIT ステートメントを発行します。
   - サービス・サブクラスをドロップしている場合で、そのサービス・サブクラスにワーク・アクションがマップされている場合は、ALTER WORK ACTION SET ステートメントの DROP WORK ACTION 節を使用してワーク・アクションをドロップしてください。あるいは別の方法として、DROP WORK ACTION SET ステートメントを使用し、サービス・サブクラスにマップしているワーク・アクションが含まれるワーク・アクション・セットをドロップします。各ワーク・アクションをドロップした後、またはワーク・アクション・セットをドロップした後、COMMIT ステートメントを発行します。
3. ドロップするサービス・クラスに関連付けられているしきい値をすべて使用不可にして、ドロップします。各しきい値をドロップした後、COMMIT ステートメ

ントを発行します。 しきい値を使用不可にするには ALTER THRESHOLD ステートメントを、しきい値をドロップするには DROP THRESHOLD ステートメントを使用します。

4. ドロップするオブジェクトに応じて、以下を行います。
   - サービス・サブクラスをドロップする場合は、DROP SERVICE CLASS ステートメントを使用してサービス・サブクラスをドロップします。
   - サービス・スーパークラスをドロップする場合は、DROP SERVICE CLASS ステートメントを使用して、そのサービス・スーパークラスに関連付けられているすべてのサービス・サブクラスをドロップし、各サービス・サブクラスがドロップされた後で COMMIT ステートメントを発行します。次いで、DROP SERVICE CLASS ステートメントを発行してサービス・スーパークラスをドロップします。

     注: サービス・スーパークラスのデフォルトのサービス・サブクラスは、手動ではドロップできません。サービス・スーパークラスのデフォルトのサービス・サブクラスは、サービス・スーパークラスをドロップするときにドロップされます。
5. ALTER SERVICE CLASS ステートメントを使用して、サービス・クラスを使用不可にします。 サービス・スーパークラスをドロップしている場合は、このアクションによって、そのサービス・スーパークラスに関連付けられているすべてのサービス・サブクラスが使用不可になります。サービス・クラスを使用不可にすると、そのサービス・クラスにはそれ以上アクティビティーを関連付けることができなくなります。サービス・クラスを使用不可にした後、COMMIT ステートメントを発行します。
6. 変更をコミットします。 変更をコミットすると、サービス・クラスが SYSCAT.SERVICECLASSES ビューから除去されます。

例: サービス・クラスの使用

以下の例では、サービス・クラスを使用してデータベース・ワークロードを制御する方法について示します。

この例は、International Beer Emporium という架空の企業を使用して説明されます。International Beer Emporium は販売、会計、技術、検査、および生産という 5 つの主要な部門で構成されている中堅企業です。 5 つの部門はすべて、同じ製品カタログ・データベースを共有しています。

DB2 ワークロード管理ソリューションの初期インプリメンテーション

製品カタログ・データベースは、ほとんどいつも良好に稼働しています。ただし、ユーザーから、最大接続数を超えたためにアプリケーションがデータベースに接続できないという苦情が上がることがあります。 DB2 バージョン 9.7 へのアップグレード後に、データベース管理者の Bob がサービス・クラスを試してみることにします。 Bob は、5 つの部門ごとに製品カタログ・データベースの使用パターンを調べ、データベースの接続が時々不足する理由を確かめたいと思っています。以下に、Bob がサービス・クラスをセットアップする場合に従うステップを示します。

1. まず最初に、Bob は各部門のサービス・スーパークラスを作成します (各サービス・スーパークラスごとにデフォルトのサービス・サブクラスも自動的に作成されます)。

- 販売部門には SALES が作成されます。

  ```
  CREATE SERVICE CLASS SALES
  ```

- 会計部門には ACCOUNTING が作成されます。

  ```
  CREATE SERVICE CLASS ACCOUNTING
  ```

- 技術部門には ENGINEERING が作成されます。

  ```
  CREATE SERVICE CLASS ENGINEERING
  ```

- 検査部門には TESTING が作成されます。

  ```
  CREATE SERVICE CLASS TESTING
  ```

- 生産部門には PRODUCTION が作成されます。

  ```
  CREATE SERVICE CLASS PRODUCTION
  ```

2. Bob は、各部門ごとに適切な許可 ID を持つセッション・ユーザー・グループを作成します。
   - 許可 ID SALESGRP を持つセッション・ユーザー・グループが作成されます。このグループには、販売部門にいるすべてのユーザーの許可 ID が含まれます。
   - 許可 ID ACCTNGRP を持つセッション・ユーザー・グループが作成されます。このグループには、会計部門にいるすべてのユーザーの許可 ID が含まれます。
   - 許可 ID ENGINGRP を持つセッション・ユーザー・グループが作成されます。このグループには、技術部門にいるすべてのユーザーの許可 ID が含まれます。
   - 許可 ID TESTGRP を持つセッション・ユーザー・グループが作成されます。このグループには、検査部門にいるすべてのユーザーの許可 ID が含まれます。
   - 許可 ID PRODGRP を持つセッション・ユーザー・グループが作成されます。このグループには、生産部門にいるすべてのユーザーの許可 ID が含まれます。
3. Bob はワークロードを作成し、各グループからの接続を関連したサービス・クラスにマップします。
   - セッション・ユーザー・グループが SALESGRP に設定されたワークロード WL_SALES が作成されます。 WL_SALES は、その接続をサービス・スーパークラス SALES にマップします。

     ```
     CREATE WORKLOAD WL_SALES SESSION_USER GROUP ('SALESGRP')
     SERVICE CLASS SALES
     ```

   - セッション・ユーザー・グループが ACCTNGRP に設定されたワークロード WL_ACCOUNTING が作成されます。 WL_ACCOUNTING は、その接続をサービス・スーパークラス ACCOUNTING にマップします。

     ```
     CREATE WORKLOAD WL_ACCOUNTING SESSION_USER GROUP ('ACCTNGRP')
     SERVICE CLASS ACCOUNTING
     ```

   - セッション・ユーザー・グループが ENGINGRP に設定されたワークロード WL_ENGINEERING が作成されます。 WL_ENGINEERING はその接続をサービス・クラス ENGINEERING にマップします。

     ```
     CREATE WORKLOAD WL_ENGINEERING SESSION_USER GROUP ('ENGINGRP')
     SERVICE CLASS ENGINEERING
     ```

- セッション・ユーザー・グループが TESTGRP に設定されたワークロード WL_TEST が作成されます。 WL_TEST はその接続をサービス・クラス TESTING にマップします。

  ```
  CREATE WORKLOAD WL_TEST SESSION_USER GROUP ('TESTGRP')
  SERVICE CLASS TESTING
  ```

- セッション・ユーザー・グループが PRODGRP に設定されたワークロード WL_PRODUCTION が作成されます。 WL_PRODUCTION はその接続をサービス・クラス PRODUCTION にマップします。

  ```
  CREATE WORKLOAD WL_PRODUCTION SESSION_USER GROUP ('PRODGRP')
  SERVICE CLASS PRODUCTION
  ```

Bob は、デフォルトのサービス・クラスおよびワークロード設定を使用します。また、サービス・クラスに対する何らかの制御を行う前に、データベースの使用パターンを監視したいと考えています。結果として作成されるサービス・スーパークラス定義は次のとおりです。

表 29. サービス・クラス定義

サービス・クラス
SALES
ACCOUNTING
ENGINEERING
TESTING
PRODUCTION
SYSDEFAULTUSERCLASS
SYSDEFAULTMAINTENANCECLASS
SYSDEFAULTSYSTEMCLASS

前述のとおりにインプリメントされた DB2 ワークロード管理ソリューションを使用すると、各部門からの作業はそれ自体のサービス・スーパークラスにルーティングされます。どの部門からのものかが明確でない作業は、デフォルトのサービス・スーパークラス SYSDEFAULTUSERCLASS にマップされます。この構成を使用して、Bob は各サービス・クラスでの作業をモニターし、部門のデータベース使用パターンを判別することができます。

DB2 ワークロード管理インプリメンテーションの最初の改良

最新の接続ピークの後、Bob は WLM_GET_SERVICE_SUPERCLASS_STATS 表関数を使用してサービス・スーパークラスの統計を照会し、各サービス・スーパークラスごとに接続の最高水準点の値を調べます。 Bob は、検査部門を除くすべての部門において接続の最高水準点がほぼ 100 であることに気付きます。ただし、検査部門の統計では、検査チームがある時に 800 を超える接続を確立したことが示されています。

検査部門では、月に一度、月ごとの徹底的な製品テストが行われます。このときに、部門において最大 1000 の同時接続が確立されます。データベース・マネージャー構成パラメーター **max_connections** は 1000 に設定されているため、検査部門が使用可能なデータベースへの接続のほとんどを使用していることになります。システムに 1000 の接続が存在する場合、その後のすべての接続は拒否されます。

システム上のメモリー制約のため、**max_connections** および **maxagents** 構成値をデータ・サーバー上で増やして、より多くの接続を許可することはできません。

Bob は、検査部門がすべての接続を使用してしまわないように、検査部門からの接続数を制限して、他の 4 つの部門がそれぞれのビジネス目標を達成するためにデータベースへの十分な接続を得られるようにすることを決定します。

他の 4 つの部門は通常、それぞれ 150 を超える同時接続を必要としません。さらに、Bob はデフォルト・ユーザー、デフォルト保守、およびデフォルト・システムの各サービス・スーパークラスに接続が含まれることはほとんどないことにも気付きます。それで、これらのデフォルト・サービス・スーパークラスには 100 の接続で十分であると判断します。使用可能な 1000 個の接続を持つ **max_connections** プールから 700 の接続 (600 は 4 つの部門用、100 はデフォルト・クラス用) が割り振られ、検査部門が使用できる接続は 300 になります。検査部門の接続を最大 300 に制限することによって、他の部門からのユーザーは接続要求を拒否されることがなくなります。

検査グループの同時接続を最大 300 に制限するために、Bob は TESTING サービス・クラス用にしきい値が 300 の MAXSERVICECLASSCONNECTIONS を作成します。

```
CREATE THRESHOLD MAXSERVICECLASSCONNECTIONS FOR SERVICE CLASS TESTING ACTIVITIES
ENFORCEMENT DATABASE PARTITION
WHEN TOTALSCMEMBERCONNECTIONS > 300 STOP EXECUTION
```

この変更を実施した後、DB2 ワークロード管理構成は次のようになります。

表 30. TESTING サービス・スーパークラス用のしきい値を追加した後の構成

サービス・クラス	MAXSERVICECLASSCONNECTIONS しきい値
SALES	N/A
ACCOUNTING	N/A
ENGINEERING	N/A
TESTING	300
PRODUCTION	N/A
SYSDEFAULTUSERCLASS	N/A
SYSDEFAULTMAINTENANCECLASS	N/A

TESTING サービス・クラスに含めることができる最大同時接続は 300 のみであるため、このしきい値を上回る接続要求はすべて拒否されます。一方、MAXSERVICECLASSCONNECTIONS しきい値は他のサービス・クラスには適用されないため、これらのサービス・クラスはデータ・サーバーへの残りの 700 個の使用可能な接続を共有します。これらのサービス・クラス間において接続の競合は存在しません。このため、Bob は接続しきい値をそれらのサービス・クラスには設定しません。

DB2 ワークロード管理インプリメンテーションの 2 回目の改良

販売、会計、技術、および生産部門からの接続が拒否されることはなくなりましたが、これらの部門のユーザーは依然として、検査部門が徹底的な製品検査を行って

いるときにパフォーマンスが低下することについて苦情を述べています。 Bob は、検査部門がその製品検査サイクル中に実行する照会を調べ、その照会に大量のデータが関係する複雑な結合が含まれていることに気付きます。これらの照会では、非常に多くのプリフェッチ・アクティビティーが生成されます。これにより、他の部門からの接続のプリフェッチ要求が処理できなくなります。 Bob は検査部門からの接続のプリフェッチ優先順位を下げることにし、TESTING サービス・クラスのプリフェッチ優先順位の設定を LOW に変更します。

```
ALTER SERVICE CLASS TESTING PREFETCH PRIORITY LOW
```

DB2 ワークロード管理構成は次のようになります。

表 31. TESTING サービス・スーパークラス用のプリフェッチ優先順位を変更した後の構成

サービス・クラス	MAXSERVICECLASSCONNECTIONS しきい値	プリフェッチ優先順位
SALES	N/A	DEFAULT
ACCOUNTING	N/A	DEFAULT
ENGINEERING	N/A	DEFAULT
TESTING	300	LOW
PRODUCTION	N/A	DEFAULT
SYSDEFAULTUSERCLASS	N/A	DEFAULT
SYSDEFAULTMAINTENANCECLASS	N/A	DEFAULT

TESTING サービス・クラスのプリフェッチ優先順位を LOW に設定すると、検査部門から発行された接続からのプリフェッチ要求は、他の部門からのすべてのプリフェッチ要求が処理された後に初めて扱われるようになります。この変更により、他の部門の照会スループットは増加し、検査部門の製品検査フェーズ中のスループットは低下します。

DB2 ワークロード管理インプリメンテーションの 3 回目の改良

プリフェッチ問題が解決した後、技術部門は Brewmeister という実験的なアプリケーション用にいくつかの接続を必要としていることを Bob に伝えます。そのアプリケーションは実験用であるため、Bob はそれが多くのデータベース接続を消費しないようにするとともに、システムがビジーである場合はそのアプリケーションからの照会がプリフェッチャーの競合を起こさないようにしたいと思っています。これらの目標を達成するため、実験的アプリケーション用の新規のサービス・サブクラスを ENGINEERING サービス・スーパークラスの下に作成し、さらにアプリケーションからの接続を新規のサービス・サブクラスにマップするためのワークロードを作成します。 Bob はサービス・クラスおよびワークロードを次のように更新します。

- サービス・サブクラス EXPERIMENT がサービス・スーパークラス ENGINEERING の下に作成されます。

  ```
  CREATE SERVICE CLASS EXPERIMENT UNDER ENGINEERING
  ```

- しきい値が 50 の MAXSERVICECLASSCONNECTIONS がサービス・サブクラス EXPERIMENT 用に作成されます。

  ```
  CREATE THRESHOLD MAXSERVICECLASSCONNECTIONS FOR SERVICE CLASS EXPERIMENT
  UNDER ENGINEERING ACTIVITIES
  ENFORCEMENT DATABASE WHEN TOTALMEMBERCONNECTIONS > 50 STOP EXECUTION
  ```

- アプリケーション BREWMEISTER からの接続をサービス・サブクラス EXPERIMENT にマップするためにワークロード WL_EXPERIMENT が作成されます。

  ```
  CREATE WORKLOAD WL_EXPERIMENT APPLNAME ('BREWMEISTER') SERVICE CLASS EXPERIMENT
  UNDER ENGINEERING
  ```

- EXPERIMENT サービス・サブクラスのプリフェッチ優先順位は LOW に設定されます。

  ```
  ALTER SERVICE CLASS EXPERIMENT UNDER ENGINEERING PREFETCH PRIORITY LOW
  ```

DB2 ワークロード管理構成は次のようになります。

表 32. EXPERIMENT サービス・サブクラスを使用した構成

サービス・クラス	MAXSERVICECLASSCONNECTIONS しきい値	プリフェッチ優先順位
SALES	N/A	DEFAULT
ACCOUNTING	N/A	DEFAULT
ENGINEERING	N/A	DEFAULT
EXPERIMENT	50	LOW
TESTING	300	LOW
PRODUCTION	N/A	DEFAULT
SYSDEFAULTUSERCLASS	N/A	DEFAULT
SYSDEFAULTMAINTENANCECLASS	N/A	DEFAULT

この構成により、BREWMEISTER アプリケーションは 50 個のデータベースへの同時接続のみを保持することができます。さらに、このアプリケーションからのプリフェッチ要求は、優先順位が低いプリフェッチ・キューに送信されます。これで、誤ってデータベース・システムの処理を不能にしてしまうことはないとわかり、技術部門はアプリケーションを使用した実験を安全に行うことができるようになりました。

例: サービス・クラス関連のシステム・スローダウンの分析

システム・スローダウン (例、一部のアプリケーションを完了するのに予想以上の時間がかかる、など) に気付き、その問題がサービス・クラスの構成に関連しているかどうかが不確かな場合、表関数のデータを使用して問題を調べたり、必要に応じて訂正したりすることができます。

最初に、サービス・クラスで何が起きているか概要を把握します。この概要には、アクティビティーの平均存続期間、異常終了ではなく正常に完了したアクティビティーの数、およびシステム内の並行コーディネーター・アクティビティーの最高水準点が含まれるはずです。この情報を取得するには、表関数 WLM_GET_SERVICE_SUBCLASS_STATS から取得したデータを使って、複数のサービス・クラスとデータベース・メンバーにまたがる集約を伴う一般照会を作成することができます。すべてのデータベース・メンバーのすべてのサービス・クラスにおいてデータを収集することを指定するには、最初および 2 番目の引数を空ストリングに設定し、3 番目の引数を -2 (ワイルドカード文字) に設定します。照会は次のようになります。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(SUM(COORD_ACT_COMPLETED_TOTAL)),1,13) AS ACTSCOMPLETED,
       SUBSTR(CHAR(SUM(COORD_ACT_ABORTED_TOTAL)),1,11) AS ACTSABORTED,
       SUBSTR(CHAR(MAX(CONCURRENT_ACT_TOP)),1,6) AS ACTSHW,
       CAST(CASE WHEN SUM(COORD_ACT_COMPLETED_TOTAL) = 0 THEN 0
                 ELSE SUM(COORD_ACT_COMPLETED_TOTAL * COORD_ACT_LIFETIME_AVG)
                 / SUM(COORD_ACT_COMPLETED_TOTAL) END / 1000 AS DECIMAL(9,3))
       AS ACTAVGLIFETIME
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS ('', '', -2)) AS SCSTATS
GROUP BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
```

以前には、この照会で以下の結果が報告されたとします。

```
SUPERCLASS_NAME     SUBCLASS_NAME      ACTSCOMPLETED ACTSABORTED ACTSHW ACTAVGLIFETIME
------------------- ------------------ ------------- ----------- ------ --------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 8             0           1               3.750
BI_APPS             SYSDEFAULTSUBCLASS 4             0           1               5.230
BATCH               SYSDEFAULTSUBCLASS 1             0           1              25.600
```

この照会によって返されるデータは、スローダウンが BI_APPS サービス・クラスで発生していることを示すのに十分かもしれません。そのアクティビティーの平均存続期間が通常よりもかなり高いことを示しているためです。この状態は、その特定のサービス・クラスで使用可能なリソースが使い尽くされようとしていることを示している可能性があります。

すべてのデータベース・メンバーのサービス・クラスの平均では問題を切り分けることができない場合、各メンバーの平均値を分析することを考慮してください。各メンバーの平均を全体の平均に集約すると、データベース・メンバー間の大きな不整合が隠されてしまうおそれがあります。この場合、すべてのメンバーがコーディネーター・メンバーとして使用されていることが前提になっています。この前提が正しくない場合、コーディネーター・メンバー以外で計算される平均存続期間はゼロになります。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEM,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3)) AS AVGLIFETIME
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '', -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME

SUPERCLASS_NAME     SUBCLASS_NAME      MEMB AVGLIFETIME
------------------- ------------------ ---- -----------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0          3.425
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1          2.752
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2          8.230
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3          0.593
```

この例では、メンバー 2 は通常よりも多くの作業を受け取っている可能性があります。アクティビティーの平均存続期間が他のデータベース・メンバーの平均存続期間よりも非常に高いためです。

システム・スローダウンが発生する原因としては多くの異なる状態が考えられます。以下の原則を使用して、DB2 ワークロード管理の表関数によって提供される情報を最大限に活用してください。

- アプリケーション・ロジックおよび環境のレベル (分離レベルなど) での大量のロック競合を解決します。

- サービス・クラスがそのしきい値レベル (並行要求の数など) に近づいている場合、しきい値を増やす必要があるかもしれません。
- サービス・クラスに割り当てられたリソースが使い尽くされようとしていて OUTBOUT CORRELATOR が設定されている場合は、オペレーティング・システム・サービス・クラスへのマッピングが問題の原因となっている可能性があります (つまり、サービス・クラスに対応するオペレーティング・システムのサービス・クラスが十分なプロセッサー・リソースを得られていないということです)。
- サービス・クラスで実行されているアクティビティーの数が予想よりも多い可能性があり、これにより、通常よりも多くのリソースが消費されている可能性があります。完了したアクティビティーの数を確認して、サービス・クラスで実行中の作業量が妥当であるかどうかを判別してください。
- 予想よりも多くのアクティビティーがサブミットされており、並行性しきい値が定義されている場合、アクティビティーがキューでより多くの時間を費やしている可能性があります。アクティビティーの平均キュー時間が平均存続期間と同じだけ増加しているかどうかを確認してください。同じ量だけ増加している場合、キューの動作は予想どおりです。しかし、存続時間が容認できないものである場合は、さらに多くのリソースをサービス・クラスに割り振り、並行性しきい値を減らすことを考慮してください。

例: サービス・クラスによるエージェント使用の調査

DB2 ワークロード管理に備わっている WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用すると、サービス・クラス間でのエージェントの相対的な分散を判別できます。

エージェントなどのデータ・サーバー・リソースがユーザーまたはアプリケーションのグループによって使用過多になるという状況が起こり得ます。例えば特定のユーザーのグループが、使用可能なほとんど全部のエージェントを使用していて、このグループ以外のユーザーからこの状況について知らされるとします。

実行すべき最初のステップは、各サービス・クラスを操作しているエージェント数を判別することです。以下のような照会を使用できます。

```
SELECT SUBSTR(AGENTS.SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(AGENTS.SERVICE_SUBCLASS_NAME,1,19) AS SUBCLASS_NAME,
       COUNT(*) AS AGENT_COUNT
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', CAST(NULL AS BIGINT), -2))
AS AGENTS
WHERE AGENT_STATE = 'ACTIVE'
GROUP BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME

SUPERCLASS_NAME     SUBCLASS_NAME       AGENT_COUNT
------------------- ------------------- -----------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS            7
TEST                SYSDEFAULTSUBCLASS           20
```

特定のサービス・クラスで正当な数を超えるエージェントが使用されていると判断する場合、ワークロードまたはサービス・クラスで許可されるアクティビティー数を制限するアクションを実行できます。別の方法としては、サービス・クラスに対する接続数を制限できます。

ワーク・アクション・セットによるアクティビティーのタイプへの制御の適用

ワーク・アクション・セットは、特定のサービス・スーパークラス内か、特定のワークロード内の特定のタイプのアクティビティーまたはデータベース全体に対して制御を適用するワーク・アクションを含んでいます。

ワーク・アクションは、ワーク・クラスに適用できるアクションを提供します。ワーク・クラスは、LOAD アクティビティーや READ アクティビティーのような特定のタイプのアクティビティーを表します。ワーク・アクションは、アクティビティーが実行を開始する前にワーク・アクションが適用されていたワーク・クラスに該当するアクティビティーに適用されます。ただし、ワーク・アクションが PREVENT EXECUTION である場合、アクティビティーは実行できず、その他のワーク・アクションはそのアクティビティーに適用されません。

ワーク・アクション・セットをデータベースに適用した場合、あるワーク・クラスに該当するアクティビティーに適用できるアクションのタイプがいくつかあります (しきい値定義、実行の回避、アクティビティー・データの収集、アクティビティーのカウントなど)。ワーク・アクションのしきい値の定義は、データベースのワーク・アクションとして最も強力なものです。例えば、SQL が 100 000 を超える行数を読み取ったり返したりしないようにする場合を考慮します。 SQL READ ステートメントを識別するワーク・アクション・セットのために単一のワーク・クラスを定義し、戻り行数が 100 000 を超えた場合に実行を停止するしきい値を持つワーク・アクションを定義できます。実行可能なアクションについて詳しくは、 125 ページの『ワーク・アクションとワーク・アクション・セットのドメイン』を参照してください。

ワーク・アクション・セットをワークロードに適用する場合、さまざまなタイプのアクションをアクティビティーに適用することができます。例えば、しきい値の定義、実行の回避、アクティビティー・データおよび集約アクティビティー・データの収集、アクティビティーのカウントなどです。

サービス・スーパークラスのワーク・アクション・セットを定義した場合、アクティビティーに適用できるアクションのタイプには、サービス・サブクラスへのアクティビティーのマッピング、実行の回避、アクティビティーまたは集約アクティビティー・データの収集、アクティビティーのカウントというさまざまなものが含まれます。通常、ワーク・アクション・セットはアクティビティーをサービス・サブクラスにマップし、アクティビティーの管理の助けとなるようにそのサブクラスにしきい値を定義しています。

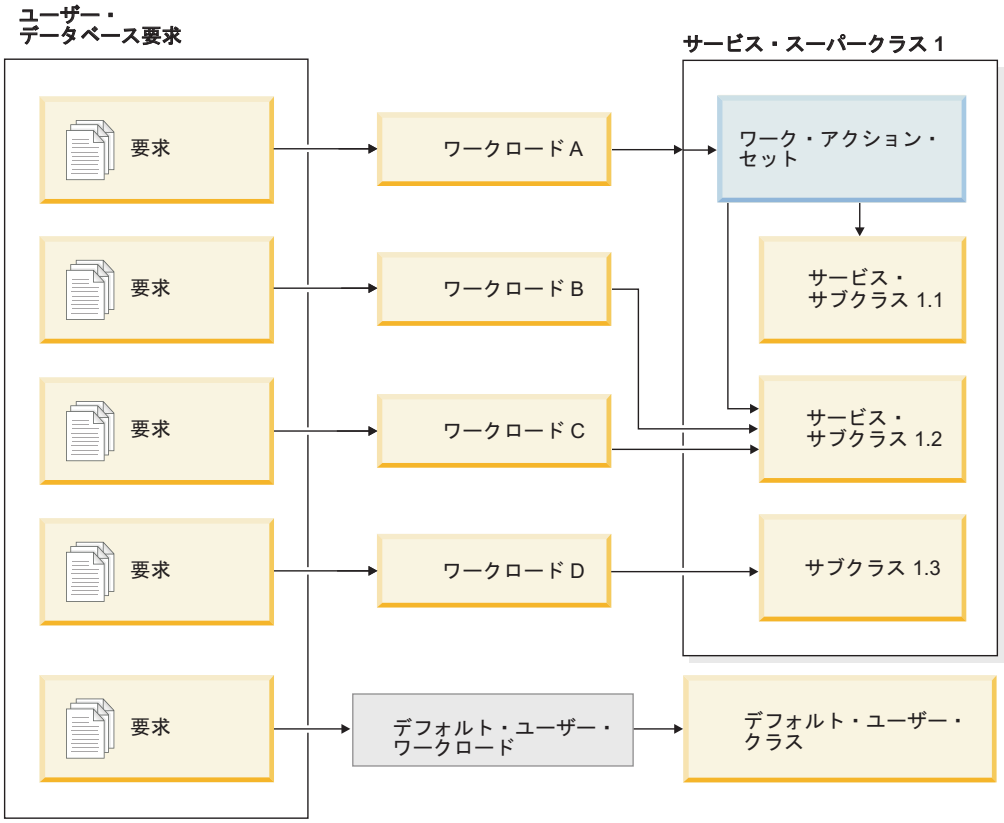


図 15. サービス・スーパークラスのためのワーク・アクション・セットのマッピング

どのようにワーク・クラス、ワーク・クラス・セット、ワーク・アクション、ワーク・アクション・セットが一緒に動作し、他の DB2 オブジェクトに関連付けられるか

ワーク・クラスおよびワーク・アクションは一緒に動作し、特定のアクションを特定のアクティビティー・タイプに適用します。これがどのように動作するか、例を使って説明します。

以下の図は、ワーク・クラス、ワーク・クラス・セット、ワーク・アクション、ワーク・アクション・セットがどのように一緒に動作し、他の DB2 オブジェクトと関連付けられるかについての概要を示しています。

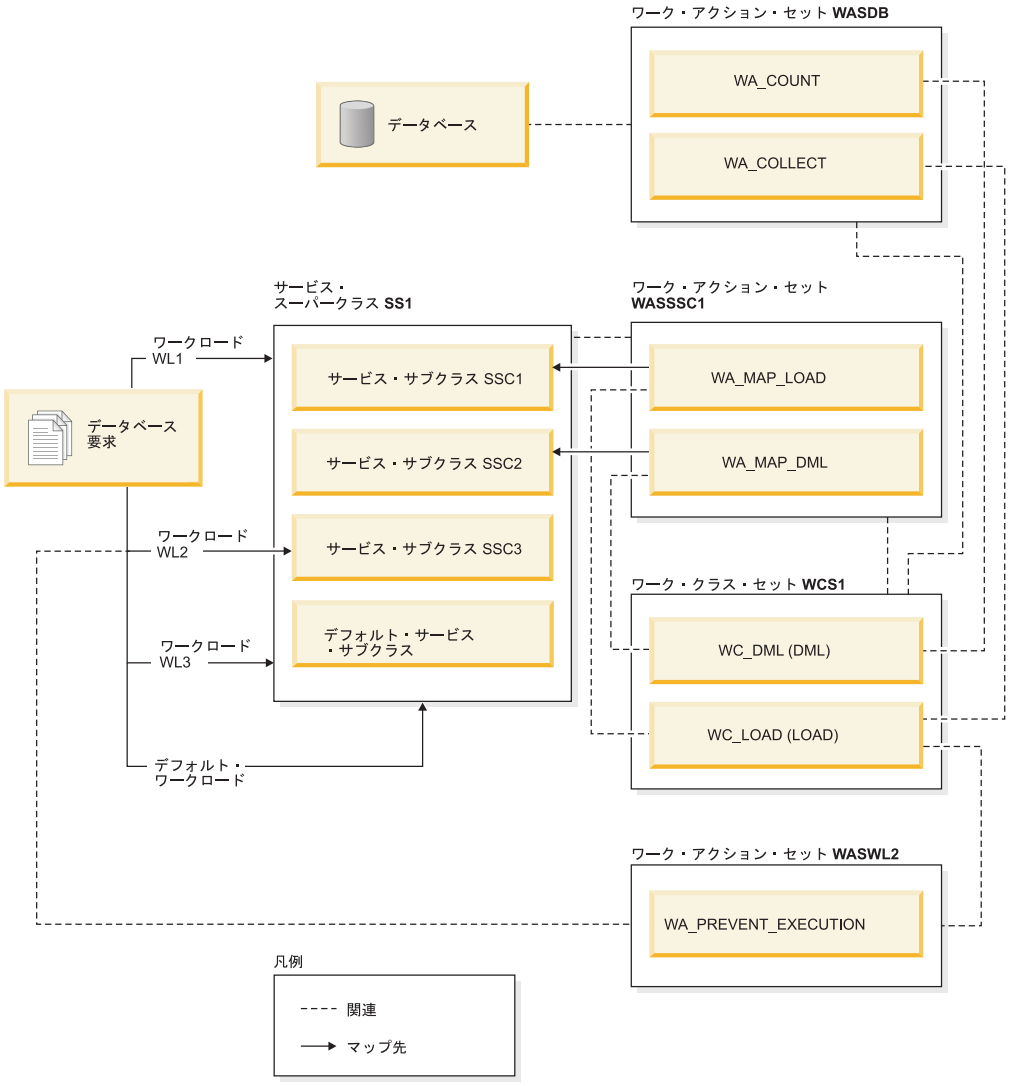


図 16. ワーク・アクション・セットとワーク・クラス・セットの概要

この図では、一部のデータベース・アクティビティーは、ワークロード WL1、ワークロード WL3、およびデフォルトのユーザー・ワークロードである SYSDEFAULTUSERWORKLOAD から、サービス・スーパークラス SS1 にマップされています。ワーク・アクション・セット WASDB はデータベースに適用されているので、(ワークロードにかかわらず) データベースに入れられるすべてのアクティビティー、および WC_DML または WC_LOAD ワーク・クラスに分類されるすべてのアクティビティーには、WASDB ワーク・アクション・セット内のワーク・アクションが適用されます。つまり、DML 作業タイプのアクティビティーはカウントされ、LOAD 作業タイプのアクティビティーはそれに関するアクティビティー・データが収集されて、アクティブ・イベント・モニターに書き込まれます (使用可能な場合)。

ワーク・アクション・セット WASSSC1 はサービス・スーパークラス SS1 に適用されます。デフォルトのユーザー・ワークロード、WL1 ワークロード、または WL3 ワークロードに割り当てられており、WC_DML ワーク・クラスと WC_LOAD ワーク・クラスに分類されるアクティビティーも、WA_MAP_DML および WA_MAP_LOAD ワーク・アクションが適用されます。それらのワークロードは

SS1 サービス・スーパークラスに作業を送信するためです。つまり、作業タイプが LOAD のアクティビティーは、WA_MAP_LOAD ワーク・アクションにより SSC1 サービス・サブクラスにマップされ、作業タイプが DML のアクティビティーは、WA_MAP_DML ワーク・アクションにより SSC2 サービス・サブクラスにマップされます。

WL2 ワークロードに割り当てられるアクティビティーは、サービス・サブクラス (SSC3) に直接マップされます。ワークロードがアクティビティーをサービス・サブクラスに直接マップする場合、ワーク・アクション・セット WASSSC1 のワーク・アクションはそれらのアクティビティーに適用されません。しかし、WASWL2 は WL2 に適用されるので、WL2 に割り当てられているアクティビティーおよび WL_LOAD に分類されるアクティビティーには WASWL2 ワーク・アクション・セット内のワーク・アクションが適用されます。つまり、PREVENT EXECUTION ワーク・アクションにより、LOAD アクティビティーの実行は許可されません。

ワーク・アクションとワーク・アクション・セット

ワーク・アクションは、ワーク・クラスと併用される場合、特定のタイプのアクティビティーを制御するために使用できます。例えば、さまざまなワーク・アクションを LOAD アクティビティーに適用して、それらが DML とは異なる仕方で処理されるようにすることができます。ワーク・アクションはワーク・アクション・セットにグループ化されます。

ワーク・アクション

ワーク・アクションは以下の属性で構成されます。

- ユーザー提供のワーク・アクション名。これはワーク・アクション・セット内で固有でなければなりません。
- ワーク・アクションが適用されるワーク・クラス ID。ワーク・クラスに対しては複数のワーク・アクションを定義できますが、それぞれのワーク・アクションは、そのワーク・クラスに対して異なるアクションを実行するものでなければなりません。
- ワーク・クラスに一致するデータベース・アクティビティーに適用されるアクション。ワーク・アクションの有効なアクションのタイプは、ワーク・アクションが属するワーク・アクション・セットが、データベースに適用されるか、ワークロードに適用されるか、それともサービス・スーパークラスに適用されるかに応じて異なります。ワーク・アクション・セットがデータベースに適用される場合、データベースに入るすべての作業はワーク・アクション・セットによって評価されます。ワーク・アクション・セットがワークロードに適用される場合、そのワークロードに関連付けられたワークロード・オカレンスを介してサブミットされるすべての作業は、そのワーク・アクション・セットによって評価されます。ワーク・アクション・セットがサービス・スーパークラスに適用される場合、実行用にサービス・スーパークラスに対して直接サブミットされるすべての作業は、そのワーク・アクション・セットによって評価されます。つまり、ターゲット・サービス・クラスとしてサービス・スーパークラスを特に指定しているワークロード定義では、そのサブミットされた作業はサービス・スーパークラスで定義されたワーク・アクション・セットによって評価されます。以下に例を示します。

– データベースまたはワークロードに適用されるワーク・アクション・セットには、しきい値ワーク・アクションを含めることができます。しきい値ワーク・アクションが定義されているワーク・クラスにアクティビティーが割り当てられた場合、しきい値はそのアクティビティーに適用されます。
– サービス・スーパークラスに適用されるワーク・アクション・セットには、アクティビティーをサービス・スーパークラス内のサービス・サブクラスにマップするワーク・アクションを含めることができます。アクティビティーがワーク・クラス・セット内の特定のワーク・クラスに対応し、ワーク・アクション・セットがそのワーク・クラスに定義されたマッピング・ワーク・アクションを持つ場合、そのアクティビティーは、ワーク・アクションにより指定されたサービス・サブクラスにマップされます。

サポートされるアクションのリストについては、 125 ページの『ワーク・アクションとワーク・アクション・セットのドメイン』を参照してください。

- 指定したアクションのターゲットのオブジェクト。アクションに応じて、オブジェクトは、アクティビティーのマップ先のサービス・サブクラスとすることもできますし、どのしきい値をアクティビティーに適用するかを指定するしきい値とすることもできます。あるいは、アクションが実行を妨げるもの、収集アクションの 1 つ、あるいはカウント・アクティビティーである場合には、NULL にすることもできます。
- 特定のインターバル中、このワーク・アクションの割り当て先のワーク・クラスに関連するアクティビティーがどれだけの時間に渡って稼働しなければならなかったか (ミリ秒数) についての統計情報を収集するためのヒストグラムを記述するテンプレート。この情報は、ワーク・アクション・タイプが COLLECT AGGREGATE ACTIVITY DATA (BASE または EXTENDED のいずれか) の場合にのみ収集されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、 292 ページの『ワークロード管理のヒストグラム』を参照してください。
- ワーク・アクションが使用可能かどうか。
- ワーク・アクションを識別する自動生成 ID。

ワーク・アクションを作成するには、CREATE WORK ACTION SET ステートメントの WORK ACTION キーワード、または ALTER WORK ACTION SET ステートメントの ADD キーワードのいずれかを使用できます。ワーク・アクションを変更するには、ALTER WORK ACTION SET ステートメントの ALTER キーワードを使用できます。ワーク・アクションをワーク・アクション・セットから除去するには、ALTER WORK ACTION SET ステートメントの DROP キーワードを使用するか、ワーク・アクション・セット全体をドロップします。

ワーク・アクションを表示するには、SYSCAT.WORKACTIONS ビューを照会します。

ワーク・アクション・セット

ワーク・アクション・セットは以下の属性で構成されます。

- データベース内で固有のワーク・アクション・セット名。
- アクションのグループが適用される 1 つ以上のワーク・クラスを含むワーク・クラス・セットの名前。

ワーク・クラス・セットの定義は、それらに定義されているワーク・アクション・セットからは分離されているため、ワーク・クラス・セットに対して複数のワーク・アクション・セットを定義することができます。

- ワーク・アクション・セットが関連付けられるオブジェクトのタイプ (データベース、サービス・スーパークラス、またはワークロード)。
- アクションとワーク・クラス・セットが適用されるサービス・スーパークラスの名前 (サービス・スーパークラスと関連付けられたワーク・アクション・セットの場合)。
- ワーク・アクション・セットが使用可能かどうか。
- ユーザー・コメント。
- 1 つ以上のワーク・アクション (ワーク・アクション・セットには必ずしもワーク・アクションが含まれている必要はありません)。
- ワーク・アクション・セットを一意的に識別する自動生成 ID。

ワーク・アクション・セットを作成するには CREATE WORK ACTION SET ステートメントを、ワーク・アクション・セットを変更するには ALTER WORK ACTION SET ステートメントを、およびワーク・アクション・セットをドロップするには DROP WORK ACTION SET ステートメントを使用することができます。

ワーク・アクション・セットを表示するには、SYSCAT.WORKACTIONSETS ビューを照会します。

ワーク・アクション・セットを作成する場合は、ワーク・アクション・セットが適用されるオブジェクトを指定する必要があります。有効なオブジェクト・タイプは、データベース、ワークロード、またはサービス・スーパークラスです。ワーク・アクション・セットと共に機能するワーク・クラス・セットも指定する必要があります。これによりワーク・クラス・セット内のワーク・クラスを使用して、ワーク・アクションを適用するアクティビティーのタイプを識別することができます。

サービス・スーパークラスでワーク・アクション・セットを作成する場合には、次のような重要な点に注意する必要があります。

1. そのデータベース・アクティビティーがサービス・サブクラスに直接マップするようにワークロードがセットアップされている場合、そのサービス・スーパークラスと関連付けられたワーク・アクション・セットは、そのワークロードにより実行されるアクティビティーに使用されることはありません。言い換えれば、ワークロードがアクティビティーをサービス・サブクラスに直接マップしている場合、ワーク・アクション・セットはバイパスされます。ワーク・アクション・セット内のどのワーク・アクションも、サービス・サブクラスに直接マップされているアクティビティーには適用されません。
2. スーパークラスでのワーク・アクション・セット内のワーク・アクションによってサービス・サブクラスにマップされないアクティビティーはすべて、サービス・スーパークラスの SYSDEFAULTSUBCLASS で実行されます。
3. (例えば PREPARE 要求、**RUNSTATS** などの) アクティビティーとして認識されないスーパークラスに関連するワークロードによってサブミットされたすべての作

業は、サービス・スーパークラスの SYSDEFAULTSUBCLASS で実行されます。これは、ワーク・アクション・セットがそれらの作業に影響を与えないためです。

ワーク・アクションの作成

CREATE WORK ACTION SET ステートメントまたは ALTER WORK ACTION SET ステートメントを使用して、ワーク・アクションを作成します。

始める前に

ワーク・アクションを作成するためには、WLMADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』
- 559 ページの『付録 A. 一般的な命名規則』

ワーク・アクションを作成するときは、次のようにします。

- ワーク・アクションをワーク・クラスに関連付けます。ワーク・クラスは、ワーク・アクション・セットが適用されるワーク・クラス・セット内に既に存在していなければなりません。
- しきい値を指定するワーク・アクションの場合、そのワーク・アクション・セットがワークロードまたはデータベースに対して定義されていなければなりません。ワーク・アクションでサポートされるしきい値のリストは、130 ページの『ワーク・アクションで使用できるしきい値』を参照してください。
- マッピング・ワーク・アクションを作成する場合は、そのワーク・アクション・セットがサービス・スーパークラスに対して定義されていなければなりません。マップ先のサービス・サブクラスは、このワーク・アクション・セットが定義されるサービス・スーパークラス内に既に存在している必要があります。さらに、デフォルトのサービス・サブクラスを指定することはできません。
- 複数のワーク・アクションを単一のワーク・クラスに適用することはできますが、それぞれのワーク・アクションは異なるタイプでなければなりません。例えば、マッピング・ワーク・アクションおよび収集ワーク・アクションを同じワーク・クラスに適用することができます。ただし、同じワーク・クラスに適用できるのは、同じタイプのワーク・アクション 1 つのみです。例えば、複数のマッピング・ワーク・アクションを同じワーク・クラスに適用することはできません。唯一の例外は、ワーク・アクションがしきい値を表している場合です。複数のしきい値ワーク・アクションを 1 つのワーク・クラスに適用できますが、それぞれのしきい値は異なるタイプでなければなりません。
- 集約アクティビティー・データ収集ワーク・アクションを作成する場合は、そのワーク・アクション・セットがサービス・スーパークラスまたはワークロードに対して定義されていなければなりません。

手順

ワーク・アクションを作成するには、次のようにします。

1. CREATE WORK ACTION SET ステートメントの *work-action-definition* キーワード、または ALTER WORK ACTION SET ステートメントの *work-action-definition* キーワードを使用します。 ワーク・アクションに、以下を 1 つ以上指定します。
   - ワーク・アクションの名前。ワーク・アクションの名前は、ワーク・アクション・セット内で固有でなければなりません。
   - このワーク・アクションが適用されるワーク・クラスの名前。このワーク・クラスは、このワーク・アクション・セットが関連付けられているワーク・クラス・セット内にあるワーク・クラスの 1 つである必要があります。例えば、このワーク・アクションをワーク・クラス LARGEDML に適用するには、以下のように指定します。

     ```
     ON WORK CLASS LARGEDML
     ```

   - このワーク・アクションのワーク・クラスと一致するアクティビティーに適用されるアクション。
     – このワーク・アクション・セットがサービス・スーパークラスに関連付けられている場合は、MAP ACTIVITY キーワードを指定すると、ワーク・アクションはアクティビティーをそのサービス・スーパークラス内のサービス・サブクラスにマップします。デフォルトでは、マッピング・ワーク・アクションにより、ネストされたアクティビティーは、親と同じサービス・サブクラスにマップされます。ルーチン内で開かれたカーソルは、ネストされたアクティビティーの一例です。

       例えば、ワーク・アクションがサービス・サブクラス SMALLREAD にマップされるようにし、すべてのネストされたアクティビティーも同じサービス・サブクラスにマップされるようにする場合は、次のように指定します。

       ```
       MAP ACTIVITY TO SMALLREAD
       ```

       次のように指定することもできます。

       ```
       MAP ACTIVITY WITH NESTED TO SMALLREAD
       ```

       ワーク・アクションがこのサービス・サブクラスにマップされるようにし、ネストされたアクティビティーはそのサービス・サブクラスにマップされないようにする場合は、次のように指定します。

       ```
       MAP ACTIVITY WITHOUT NESTED TO SMALLREAD
       ```

       ワーク・アクションを WITHOUT NESTED として定義した場合、ネストされたアクティビティーは親アクティビティーと同じサービス・サブクラスに自動的にマップされるのではなく、それら自体のアクティビティー・タイプに従って処理されるようになります。例えば、CALL アクティビティーがサービス・サブクラス subsc1 にマップされ、このルーチン内にオープン・カーソルがあるとします。このオープン・カーソルが、別のマッピング・ワーク・アクションが適用される別のワーク・クラスに分類される場合、このオープン・カーソルは別のサービス・サブクラスへマップされます。
   - このワーク・アクション・セットがデータベースまたはワークロードに関連付けられている場合は、WHEN キーワードを指定して、アクティビティーにし

きい値を適用し、アクティビティーによりしきい値の違反が生じた場合に処置が取られるようにすることができます。ワーク・アクションには、以下のしきい値を指定できます。

- ACTIVITYTOTALTIME
- ESTIMATEDSQLCOST
- CONCURRENTDBCOORDACTIVITIES
- CPUTIME
- SQLROWSREAD
- SQLTEMPSPACE
- SQLROWSRETURNED

しきい値の違反が生じた場合に以下の処置が取られることを指定できます。

- しきい値の違反の原因となったアクティビティーに関するアクティビティー・データを収集するかどうか。収集する場合、アクティビティー・データはアクティビティーの実行完了時に、アクティブなアクティビティー・イベント・モニターに書き込まれます。デフォルトでは、アクティビティーに関するデータは収集されません。このアクティビティーに関するデータを収集する場合には、コーディネーター・メンバー、特定のメンバー、またはすべてのデータベース・メンバーから収集することができます。このデータの収集には、ステートメントおよびそのコンパイル環境に関する詳細を含めるオプションと含めないオプションがあります。ステートメントとコンパイル環境についての詳細を収集する場合は、アクティビティーに使用された入力データ値も指定できます。
- しきい値の違反の原因となったアクティビティーの実行継続を許可するかどうか。デフォルトでは、アクティビティーは停止されます。

例えば、ワーク・アクションに、2 000 timeron を超えるコストの DML ステートメントをチェックさせ、しきい値の違反が起こった場合にそのアクティビティーに関する基本的なデータを収集して、実行を継続させる場合には、以下のように指定します。

```
WHEN ESTIMATEDSQLCOST > 2000 COLLECT ACTIVITY DATA CONTINUE
```

- このワーク・アクションに定義されたワーク・クラスに対応するアクティビティーが実行されないようにするには、PREVENT EXECUTION キーワードを使用できます。
- 別のアクション (データの収集やアクティビティーのマッピングなど) でオーバーヘッドをさらに課すことなく、このワーク・クラスに関連したデータベース・アクティビティー数をカウントするには、COUNT ACTIVITY キーワードを指定できます。
- このワーク・クラス下に分類されるアクティビティーのアクティビティー・データを収集するには、COLLECT ACTIVITY DATA キーワードを指定します。収集する場合、アクティビティー・データはアクティビティーの実行完了時に、アクティブなアクティビティー・イベント・モニターに書き込まれます。デフォルトでは、アクティビティーに関するデータは収集されません。このアクティビティーに関するデータを収集する場合には、コーディネーター・メンバーまたはすべてのメンバーから収集することができます。ステートメントおよびコンパイル環境情報といったアクティビティーの詳細を収集する場合

は、WITH DETAILS キーワードを指定してそうすることができます。AND VALUES キーワードを使用して、入力データ値 (入力データ値を持つアクティビティーについて) をアクティビティー・イベント・モニターに送信することもできます。

例えば、サービス・スーパークラスに適用されるワーク・アクション・セットがあるとします。このワーク・アクションに割り当てられたすべてのアクティビティーについて、アクティビティー・データを該当するイベント・モニターに書き込むようにします。このデータには、すべての集約アクティビティー情報、コンパイル環境に関する情報、および入力データ値があればそれも含めます。以下のように指定します。

```
COLLECT ACTIVITY DATA ON ALL WITH DETAILS AND VALUES
```

- このワーク・アクション・セットがサービス・スーパークラスまたはワークロードに関連付けられている場合は、COLLECT AGGREGATE ACTIVITY DATA キーワードを指定して、ワーク・クラスに分類されるアクティビティーの集約アクティビティー・データを収集することができます。収集する場合、集約アクティビティー・データはキャプチャーされて該当するイベント・モニターへ送信されます。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。

  例えば、サービス・スーパークラスに適用されるワーク・アクション・セットがあるとします。このワーク・アクションに割り当てられたすべてのアクティビティーについて、該当するイベント・モニターに集約アクティビティー・データを書き込むようにします。このデータには、基本データ、アクティビティー・データ操作言語 (DML) の見積コストのヒストグラム、およびアクティビティー DML の到着間隔時間のヒストグラムが含まれます。以下のように指定します。

  ```
  COLLECT AGGREGATE ACTIVITY DATA EXTENDED
  ```

- 対応するワーク・クラスに対して作成されるヒストグラムを記述するために、COLLECT AGGREGATE ACTIVITY DATA ワーク・アクションによって使用されるヒストグラム・テンプレート。ワーク・アクションによって使用されるヒストグラム・テンプレートを指定すると、SYSCAT.HISTOGRAMTEMPLATEUSE (そのサービス・クラスまたはワーク・アクションによって参照されるヒストグラム・テンプレートを表示するビュー) 内に対応する行が追加されます。例えば、デフォルトの到着間隔ヒストグラム・テンプレートの到着間隔統計を収集する場合、以下のように指定します。

  ```
  INTERARRIVALTIME HISTOGRAM TEMPLATE SYSDEFAULTHISTOGRAM
  ```

  ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、292 ページの『ワークロード管理のヒストグラム』を参照してください。

- ワーク・アクションを使用可能にするか使用不可にするか。デフォルトで、ワーク・アクションは使用可能な状態で作成されますが、ENABLE キーワードまたは DISABLE キーワードを使用して、使用可能にするか使用不可にするかを指定できます。ワーク・アクションが使用不可にされると、アクティビティーがデータベースまたはサービス・スーパークラス (どのオブジェクトに対してワーク・アクション・セットを作成したかによる) に入るときに、データ・サーバーはこのワーク・アクションを無視します。

2. 変更をコミットします。 変更をコミットすると、ワーク・アクションは SYSCAT.WORKACTIONS ビューに追加されます。ワーク・アクションがしきい値である場合、そのしきい値は SYSCAT.THRESHOLDS ビューに追加されます。

   新規ワーク・アクションは、コミットされた後にはじめてデータベースで有効になり、現在実行中のデータベース・アクティビティーには影響しません。

ワーク・アクションの変更

ワーク・アクションを変更する必要がある場合には、ALTER WORK ACTION SET ステートメントを使用します。

始める前に

ワーク・アクションを変更するためには、SQLADM、WLMADM または DBADM 権限が必要です。 COLLECT 節以外の節を指定するには、許可 ID に WLMADM または DBADM 権限が組み込まれている必要があります。

その他の前提条件は、21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

手順

ワーク・アクションを変更するには、次のようにします。

1. ALTER WORK ACTION SET ステートメントの ALTER キーワードを使用して、ワーク・アクションの以下の特性 (複数可) を変更します。
   - ワーク・アクションが適用されるワーク・クラスを変更することができます。ワーク・クラスは、ワーク・アクション・セットが適用されるワーク・クラス・セット内に既に存在していなければなりません。
   - ワーク・アクションがサービス・サブクラスにマップする場合、どのサービス・サブクラスにデータベース・アクティビティーがマップされるかを変更することができます。マッピングは、同じサービス・スーパークラス内のサービス・サブクラスにのみ変更できます。デフォルトのサービス・サブクラスへはマップできません。アクティビティー内のネストされたアクティビティーが同じサービス・サブクラスへマップされるかどうかを変更することもできます。例えば、ワーク・アクションが現在 WITH NESTED として定義されている場合、これを WITHOUT NESTED に変更することができます。この変更により、ネストされたアクティビティーは親アクティビティーと同じサービス・サブクラスに自動的にマップされるのではなく、それら自体のアクティビティー・タイプに従って処理されるようになります。例えば、CALL ステートメントがサービス・サブクラス SUBSC1 にマップされ、このルーチン内にオープン・カーソルがあるとします。このオープン・カーソルが、別のマッピング・ワーク・アクションの適用される別のワーク・クラスに属する場合、このオープン・カーソルは別のサービス・サブクラスへマップされます。
   - ワーク・アクション (つまり、マッピング、しきい値、実行の阻止、アクティビティーのカウント、収集のアクション) に指定されているアクション・タイプを変更することができます。ただし、有効な作業タイプに変更しなければなりません。例えば、ワーク・アクションがアクティビティーのサービス・サブクラスへのマッピングである場合、このワーク・アクションをしきい値に変更

したり、その逆を行ったりすることはできません。この理由は、この例の場合、マッピング・アクションのためにワーク・アクション・セットがサービス・スーパークラスに適用されていなければなりませんが、しきい値アクションは、サービス・スーパークラスに適用されるワーク・アクション・セットでは有効でないからです。しきい値ワーク・アクションであるワーク・アクションのタイプを変更した場合、またはワーク・アクションのタイプをしきい値に変更した場合、以下のことが生じます。

- ワーク・アクションがしきい値で、非しきい値に変更された場合、そのしきい値は SYSCAT.THRESHOLDS ビューから除去されます。
- ワーク・アクションがしきい値ではなく、しきい値に変更された場合、新規しきい値が SYSCAT.THRESHOLDS ビューに作成されます。

注: アクションがしきい値である場合、そのしきい値のタイプを異なるしきい値に変更することはできません。したがって、例えばワーク・アクションが SQLROWSRETURNED しきい値であった場合、これを SQLTEMPSPACE しきい値に変更することはできません。さらに、有効にされた CONCURRENTDBCOORDACTIVITIES ワーク・アクションしきい値のワーク・アクション・タイプを変更することもできません。

- COLLECT AGGREGATE ACTIVITY DATA ワーク・アクションによって使用されるヒストグラム・テンプレートを変更して、対応するワーク・クラスに作成されるヒストグラムを示すことができます。ワーク・アクションによって使用されるヒストグラム・テンプレートを更新すると、SYSCAT.HISTOGRAMTEMPLATEUSE ビュー (そのサービス・クラスまたはワーク・アクションによって参照されるヒストグラム・テンプレートを表示する) 内の対応する行も更新されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、 292 ページの『ワークロード管理のヒストグラム』を参照してください。
- ワーク・アクションを使用可能にするか使用不可にするかを変更することができます。デフォルトでは、ワーク・アクションは使用可能になっています。使用可能に設定された場合、データ・サーバーは、このワーク・アクションのワーク・クラスに分類されるアクティビティーに対して、このワーク・アクションを適用することを考慮します。ワーク・アクションを使用不可にすると、そのワーク・アクションはデータ・サーバーに無視されます。

2. 変更をコミットします。 変更をコミットすると、ワーク・アクションは SYSCAT.WORKACTIONS ビューで更新されます。

ワーク・アクションを使用不可にする

ワーク・クラスに適用されないように、ワーク・アクションを使用不可にすることができます。実行時に、使用不可にされたワーク・アクションは、まるで存在しないかのように扱われます。

始める前に

ワーク・アクションを使用不可にするためには、WLMADM または DBADM 権限が必要です。

手順

ワーク・アクションを使用不可にするには、次のようにします。

1. ワーク・アクション・セットを作成するのか変更するのかに応じて、以下のステートメントのいずれかを使用します。
   - CREATE WORK ACTION SET ステートメントの DISABLE キーワードおよび ADD キーワードを使用します。以下に例を示します。

     ```
     ADD WORK ACTION work-action-name ON WORK CLASS work-class-name ... DISABLE
     ```

   - ALTER WORK ACTION SET ステートメントの DISABLE キーワードおよび ALTER キーワードを使用します。以下に例を示します。

     ```
     ALTER WORK ACTION work-action-name ... DISABLE
     ```

2. 変更をコミットします。 変更をコミットすると、ワーク・アクションは SYSCAT.WORKACTIONS ビューで更新されます。

ワーク・アクションのドロップ

必要でなくなったワーク・アクションは、ワーク・アクション・セットからドロップすることができます。

始める前に

- ワーク・アクションをドロップするためには、WLMADM または DBADM 権限が必要です。
- その他の前提条件は、21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

手順

ワーク・アクションをドロップするには、次のようにします。

1. ALTER WORK ACTION SET ステートメントの DROP キーワードを使用します。 CONCURRENTDBCOORDACTIVITIES しきい値ワーク・アクションをドロップする場合は、1 回目の ALTER WORK ACTION SET 操作でそのワーク・アクションを使用不可に設定して変更をコミットし、キューに入れられているアクティビティーがないことを確認した後、2 回目の ALTER WORK ACTION SET 操作でしきい値をドロップしてください。
2. 変更をコミットします。 変更をコミットすると、ワーク・アクションは SYSCAT.WORKACTIONS ビューから除去されます。ワーク・アクションがしきい値ワーク・アクションである場合、しきい値も SYSCAT.THRESHOLDS ビューから除去されます。

   変更されたワーク・アクション・セットおよびワーク・アクションは、コミットされた後にはじめてデータベースで有効になり、現在実行中のデータベース・アクティビティーには影響しません。

ワーク・アクション・セットの作成

ワーク・アクションおよびワーク・アクション・セットを作成するには、CREATE WORK ACTION SET ステートメントを使用します。

始める前に

ワーク・アクション・セットを作成するためには、WLMADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』

ワーク・アクション・セットを作成するときは、次のようにします。

- ワーク・アクション・セットをワーク・クラス・セットに関連付けます。ワーク・クラス・セットが既に存在している必要があります。
- さらに、ワーク・アクション・セットをデータベース、ワークロード、またはサービス・スーパークラスとも関連付けます。ワーク・アクション・セットをサービス・スーパークラスに関連付けている場合、このサービス・クラスが既に存在している必要があります。ワーク・アクション・セットをデフォルトのシステム・サービス・クラス (SYSDEFAULTSYSTEMCLASS)、デフォルトのユーザー・クラス (SYSDEFAULTUSERCLASS) またはデフォルトの保守サービス・クラス (SYSDEFAULTMAINTENANCECLASS) に定義することはできません。ワーク・アクション・セットをワークロードに関連付けている場合、ワークロードが現行サーバーに存在している必要があります。ワーク・アクション・セットをデフォルトの管理ワークロード (SYSDEFAULTADMWORKLOAD) に関連付けることはできません。

手順

ワーク・アクション・セットを作成するには、次のようにします。

1. CREATE WORK ACTION SET ステートメントを次のオプションとともに使用します。
   - ワーク・アクション・セットの名前を指定。ワーク・アクション・セットの名前は、データベースの中で固有でなければなりません。
   - ワーク・アクション・セットが関連付けられるオブジェクトを指定。データベース、ワークロード、またはサービス・スーパークラスを指定できます。ワーク・アクション・セットをデータベースに関連付けるように指定した場合、そのワーク・アクション・セットに含まれるワーク・アクションは、いずれもマッピング・ワーク・アクションまたは収集集約アクションにはなれません。ワーク・アクション・セットをサービス・スーパークラスに関連付けるように指定した場合、そのワーク・アクション・セットに含まれるワーク・アクションは、いずれもしきい値にはなれません。ワーク・アクション・セットをワークロードに関連付けるように指定した場合、そのワーク・アクション・セット内のワーク・アクションは、いずれもマッピング・ワーク・アクションにはなれません。例えば、ワーク・アクション・セットを REPORTS サービス・スーパークラスに適用するには、次のように指定します。

     ```
     FOR SERVICE CLASS REPORTS
     ```

     ワーク・アクション・セットをデータベースに適用するには、次のように指定します。

     ```
     FOR DATABASE
     ```

ワーク・アクション・セットを WL1 という名前のワークロードに適用するには、次のように指定します。

```
FOR WORKLOAD WL1
```

- ワーク・アクション・セットが関連付けられるワーク・クラス・セットを指定。ワーク・クラス・セットのワーク・クラスによって、ワーク・アクション・セット内のワーク・アクションが適用されるデータベース・アクティビティーが分類されます。例えば、ワーク・アクション・セットを LARGEREADS ワーク・クラス・セットに関連付けるには、次のように指定します。

  ```
  USING WORK CLASS SET LARGEREADS
  ```

- オプション: ワーク・アクション・セットに 1 つ以上のワーク・アクションを作成。手順については、 114 ページの『ワーク・アクションの作成』を参照してください。
- オプション: ワーク・アクション・セットを使用可能にするか使用不可にするかを指定。デフォルトでは、ワーク・アクション・セットは使用可能です。ワーク・アクション・セットが使用不可の場合、データ・サーバーはアクティビティーの実行時にこのワーク・アクション・セット (またはそれに含まれるすべてのワーク・アクション) を考慮しません。

2. 変更をコミットします。 変更をコミットすると、ワーク・アクション・セットが SYSCAT.WORKACTIONSETS ビューに追加されます。

   新しいワーク・アクション・セットがデータベースで有効になるのはコミットされた後で、現在実行中のどのデータベース・アクティビティーにも影響しません。

ワーク・アクション・セットの変更

ワーク・アクション・セットに含まれるワーク・アクションを追加、変更、ドロップしたり、ワーク・アクション・セットを使用可能または使用不可にするには、ALTER WORK ACTION SET ステートメントを使用します。

始める前に

ワーク・アクション・セットを変更するためには、SQLADM、WLMADM または DBADM 権限が必要です。 COLLECT 節以外の節を指定するには、許可 ID に WLMADM または DBADM 権限が組み込まれている必要があります。

その他の前提条件については、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』
- 559 ページの『付録 A. 一般的な命名規則』

特定のワーク・クラス・セットと併用されるワーク・アクション・セットを作成するときは、別のワーク・クラス・セットと併用されるように変更することはできません。これは、ワーク・アクション・セットに含まれるワーク・アクションには、ワーク・クラス・セットに含まれるワーク・クラスと従属関係があるためです。このワーク・アクション・セットを適用するワーク・クラス・セットを変更する場合は、ワーク・アクション・セットをドロップして再作成する必要があります。

ワーク・アクション・セットに含まれるワーク・アクションのタイプはワーク・アクション・セットが定義されているオブジェクト (データベース、ワークロード、

またはサービス・スーパークラス) に依存するため、ワーク・アクション・セットの適用先となるオブジェクトを変更することはできません。ワーク・アクション・セットが関連付けられる先のオブジェクトを変更する場合は、ワーク・アクション・セットをドロップして、再作成する必要があります。

手順

ワーク・アクション・セットを変更するには、次のようにします。

1. ワーク・アクション・セットに新しいワーク・アクションを追加する場合は、ADD キーワードを使用します。 ワーク・アクション・セットにワーク・アクションを追加するときに指定できる各パラメーターについては、 114 ページの『ワーク・アクションの作成』を参照してください。
2. 既存のワーク・アクションを変更する場合は、ALTER キーワードを使用します。 ワーク・アクションを変更する方法については、 118 ページの『ワーク・アクションの変更』を参照してください。
3. ワーク・アクションをドロップする場合は、DROP キーワードを使用します。ワーク・アクション・セットからワーク・アクションをドロップする方法については、 120 ページの『ワーク・アクションのドロップ』を参照してください。
4. 現在使用可能でないワーク・アクション・セットを使用可能にすることもできますし、その逆もできます。 使用可能なワーク・アクション・セットを使用不可にする場合、変更をコミットした後は、データ・サーバーはそのワーク・アクション・セットを無視します。詳しくは、『ワーク・アクション・セットを使用不可にする』を参照してください。ワーク・アクション・セットを使用可能にする場合、変更をコミットした後は、そのワーク・アクション・セットは、データベースに入力される、適用可能な次のアクティビティーに適用されます。

   注: ワーク・アクション・セットを使用不可にしても、その中のワーク・アクションは使用不可になりません。ただし、ワーク・アクション・セットが他の作業に影響を与えることはなくなります。並行性に関するワーク・アクションしきい値を含むワーク・アクション・セットをドロップする場合はまず、並行性ワーク・アクションを使用不可にする必要があります。これを行った後、ワーク・アクション・セットをドロップできるようになります。これは、並行性しきい値をドロップするにはまず、それを使用不可にしなければならないためです。
5. 変更をコミットします。 変更をコミットすると、ワーク・アクション・セットが SYSCAT.WORKACTIONSETS ビューで更新されます。SYSCAT.WORKACTIONS ビューは、ワーク・アクションが追加、変更、またはドロップされると更新されます。

ワーク・アクション・セットを使用不可にする

ワーク・アクション・セットを使用不可にするには、CREATE WORK ACTION SET ステートメントまたは ALTER WORK ACTION SET ステートメントの DISABLE キーワードを使用します。

始める前に

ワーク・アクション・セットを使用不可にするためには、WLMADM または DBADM 権限が必要です。

このタスクについて

ワーク・アクション・セットを使用不可にしても、その中のワーク・アクションは使用不可になりません。ただし、ワーク・アクション・セットが他の作業に影響を与えることはなくなります。実行時に、使用不可にされたワーク・アクション・セットはあたかも存在しないかのようにして処理されます。

注: 並行性に関するワーク・アクションしきい値を含むワーク・アクション・セットをドロップする場合はまず、並行性ワーク・アクションを使用不可にする必要があります。これを行った後、ワーク・アクション・セットをドロップできるようになります。これは、並行性しきい値をドロップするにはまず、それを使用不可にしなければならないためです。
例えば、READCLASSES というワーク・クラスに関連付けられた READACTIVITIES というワーク・アクション・セットがあり、そのワーク・アクション・セットが READSERVICECLASS というサービス・スーパークラスに定義されていると仮定します。 SMALLREAD ワーク・アクション・セットの中には、すべての SELECT ステートメントをサービス・サブクラス SMALLREADSERVICECLASS に再マップするワーク・アクションがあります。 READACTIVITIES ワーク・アクション・セットを使用不可にすると、すべての SELECT ステートメントはあたかも READACTIVITIES ワーク・アクション・セットが存在していないかのように処理されて、デフォルトのサービス・サブクラスにマップされます。

手順

ワーク・アクション・セットを使用不可にするには、次のようにします。

1. ワーク・アクション・セットを作成するのか変更するのかに応じて、以下のステートメントのいずれかを使用します。
   - 使用不可になっているワーク・アクション・セットを作成するには、次のようにします。

     ```
     CREATE WORK ACTION SET work-action-set-name ... DISABLE
     ```

   - 既存のワーク・アクション・セットを使用不可にするには、次のようにします。

     ```
     ALTER WORK ACTION SET work-action-set-name ... DISABLE
     ```

2. 変更をコミットします。 変更をコミットすると、ワーク・アクション・セットが SYSCAT.WORKACTIONSETS ビューで更新されます。

ワーク・アクション・セットのドロップ

ワーク・アクション・セットをドロップするには、DROP WORK ACTION SET ステートメントを使用します。

始める前に

ワーク・アクション・セットをドロップするためには、WLMADM または DBADM 権限が必要です。

このタスクについて

ワーク・アクション・セットをドロップすると、ワーク・アクション・セットおよびそれに含まれるすべてのワーク・アクションがドロップされます。

ワーク・アクション・セットに CONCURRENTDBCOORDACTIVITIES しきい値ワーク・アクションが含まれる場合、そのワーク・アクション・セットをドロップできるようにするには、まずそのワーク・アクションを使用不可にする必要があります。

手順

ワーク・アクション・セットをドロップするには、次のようにします。

1. DROP WORK ACTION SET ステートメントを使用します。
2. 変更をコミットします。 変更をコミットすると、ワーク・アクション・セットが SYSCAT.WORKACTIONSETS ビューから除去されます。さらに、そのワーク・アクション・セットの一部だったすべてのワーク・アクションが、SYSCAT.WORKACTIONS ビューから除去されます。ワーク・アクション・セットにしきい値ワーク・アクションが含まれる場合、そのしきい値は SYSCAT.THRESHOLDS ビューから除去されます。

ワーク・アクションとワーク・アクション・セットのドメイン

データベース、サービス・スーパークラス、またはワークロードにワーク・アクション・セットを定義できます。ワーク・アクション・セットに定義できるワーク・アクションのタイプは、ワーク・アクション・セットの定義対象のオブジェクトのタイプに応じて異なります。

ワーク・アクション・セットがデータベースに定義される場合、ワーク・アクション・セット内のワーク・アクションは以下のいずれかのアクションでなければなりません。

- しきい値

  一致するワーク・クラスのアクティビティーそれぞれに以下のしきい値が適用されます。

  – ACTIVITYTOTALTIME
  – CPUTIME
  – ESTIMATEDSQLCOST
  – SQLROWSREAD
  – SQLROWSRETURNED
  – SQLTEMPSPACE

  グループとして一致するワーク・クラス内のすべてのアクティビティーに以下のしきい値が適用されます。

  – CONCURRENTDBCOORDACTIVITIES

  実際のしきい値は `WHEN` *`threshold-type`* 節により指定されます。複数のしきい値のワーク・アクションは、すべてのしきい値が異なるタイプである場合は単一の

ワーク・クラスに適用できます。このアクションが指定される場合、しきい値はワーク・クラスに関連付けられたすべてのデータベース・アクティビティーに適用されます。

- PREVENT EXECUTION

  このアクションが指定された場合、関連付けられたワーク・クラスに一致するすべてのデータベース・アクティビティーは実行を許可されません。

- COLLECT ACTIVITY DATA

  このアクションが指定された場合、このワーク・アクションが定義されているワーク・クラスに対応するデータベース・アクティビティーに関する情報は、アクティビティーの実行完了時に、アクティブな ACTIVITIES イベント・モニターに書き込まれます。詳しくは、『個々のアクティビティーのデータ収集』を参照してください。

- COUNT ACTIVITY

  このアクションが指定された場合、関連するワーク・クラスにマップするすべてのデータベース・アクティビティーにより、そのワーク・クラス・タイプの回転カウンターが増分されます。 (ワーク・クラスの回転カウンターは、アクティビティーがそのワーク・クラスに関連付けられるごとに 1 ずつ増分されます。) COUNT ACTIVITY ワーク・アクションにより、このカウンターは効率的な仕方で更新されます。ワーク・クラスに対応するアクティビティーに適用されるワーク・アクションがない場合、ワーク・クラス・アクティビティー・カウンターは増分されません。時々注意すべきアクションは、特定のタイプのアクティビティーのカウントの入手だけということもあります。詳しくは、『個々のアクティビティーのデータ収集』を参照してください。

データベースに定義されているワーク・アクション・セット内のワーク・アクションがこれらのアクションのいずれでもない場合、SQL4720N が返されます。

サービス・スーパークラスにワーク・アクション・セットを定義する場合、ワーク・アクション・セット内のワーク・アクションは以下のいずれかのアクションでなければなりません。

- マッピング・アクション

  アクティビティーは、デフォルトのサービス・サブクラスを除き、サービス・スーパークラス内のどのサービス・サブクラスにでもマップできます。 MAP ACTIVITY TO SERVICE CLASS キーワードを使用して、アクティビティーのマップ先となるサービス・サブクラスを指定します。ワーク・アクション・セット内の 1 つのマップ・ワーク・アクションだけを、同じワーク・クラスに適用することができます。

- PREVENT EXECUTION

  振る舞いはデータベース・ワーク・アクションと同じです。

- COLLECT ACTIVITY DATA

  振る舞いはデータベース・ワーク・アクションと同じです。

- COLLECT AGGREGATE ACTIVITY DATA

このアクションが指定された場合、このワーク・アクションの定義対象のワーク・クラスに対応する集約データベース・アクティビティー・データが収集されます。

- COUNT ACTIVITY

  振る舞いはデータベース・ワーク・アクションと同じです。

サービス・スーパークラスに定義されているワーク・アクション・セット内のワーク・アクションがこれらのアクションのいずれでもない場合、SQL4720N が返されます。

ワーク・アクション・セットがワークロードに定義される場合、ワーク・アクション・セット内のワーク・アクションは以下のいずれかのアクションでなければなりません。

- しきい値

  一致するワーク・クラスのアクティビティーそれぞれに以下のしきい値が適用されます。

  – ACTIVITYTOTALTIME
  – CPUTIME
  – ESTIMATEDSQLCOST
  – SQLROWSREAD
  – SQLROWSRETURNED
  – SQLTEMPSPACE

  グループとして一致するワーク・クラス内のすべてのアクティビティーに以下のしきい値が適用されます。

  – CONCURRENTDBCOORDACTIVITIES

  実際のしきい値は `WHEN` *`threshold-type`* 節により指定されます。複数のしきい値のワーク・アクションは、すべてのしきい値が異なるタイプである場合は単一のワーク・クラスに適用できます。このアクションが指定される場合、しきい値はワーク・クラスに関連付けられたすべてのデータベース・アクティビティーに適用されます。

- PREVENT EXECUTION

  振る舞いはデータベース・ワーク・アクションと同じです。

- COLLECT ACTIVITY DATA

  振る舞いはデータベース・ワーク・アクションと同じです。

- COLLECT AGGREGATE ACTIVITY DATA

  振る舞いはサービス・スーパークラス・ワーク・アクションと同じです。

- COUNT ACTIVITY

  振る舞いはデータベース・ワーク・アクションと同じです。

ワークロードに定義されているワーク・アクション・セット内のワーク・アクションがこれらのアクションのいずれでもない場合、SQL4720N が返されます。

例: ワーク・クラス・アクティビティーの適用

以下の図は、LARGE ACTIVITIES と呼ばれるワーク・クラス・セット内のワーク・クラスが、データベースとサービス・スーパークラスの両方にどのように適用されるかの例を示しています。この目標に達するために、2 つのワーク・アクション・セット (「Database large activities」と「Service class large activities」) が作成されています。

この例では示されていませんが、LARGE ACTIVITIES ワーク・クラス・セット内のクラスをワークロードに適用することもできます。これは、ワークロードに関連付けられたワーク・アクション・セットを作成し、ワーク・アクション・セットを LARGE ACTIVITIES ワーク・クラス・セットに関連付けることによって行えます。

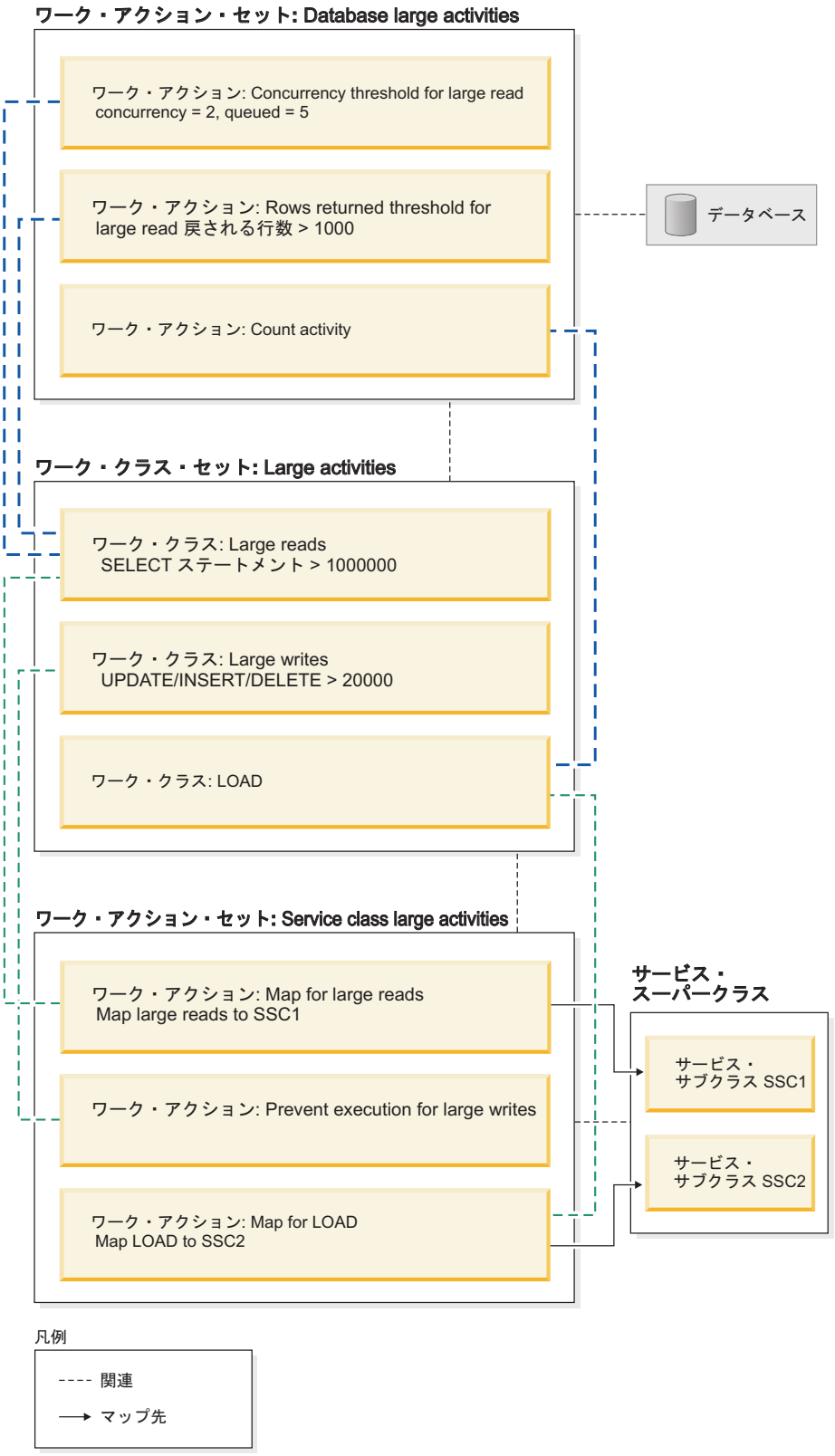


図 17. ワーク・アクション、ワーク・アクション・セット、ワーク・クラス、およびワーク・クラス・セットの例

ワーク・アクション・セットは以下のとおりです。

- Database large activities には以下が含まれています。
  - Concurrency threshold for large reads。これにより 2 つの大規模な読み取りを同時に実行し、5 つの大規模読み取りをキューに入れることができます。
  - Rows returned threshold for large reads。これにより大規模読み取りが 1000 を超える行を返すことはなくなります。
  - Count activity for load。これはロード・ユーティリティーがデータベースに対して実行される回数をカウントします。
- Service class large activities には以下が含まれています。
  - Map for large reads。これは大規模読み取りをサービス・サブクラス 1 にマップします。
  - Map for large writes。これにより大規模書き込みが実行されることはなくなります。
  - Map for LOAD。これはロードをサービス・サブクラス 2 にマップします。

ワーク・アクション・セットには、ワーク・アクション・セットが適用されるワーク・クラス・セット内のすべてのワーク・クラスに対してアクションが含まれている必要はありません。加えて、アクション・タイプが異なっていれば、複数のワーク・アクションをワーク・クラスに適用することができます。しきい値タイプが異なっていれば、複数のしきい値ワーク・アクションをワーク・クラスに適用することができます。

ワーク・アクションで使用できるしきい値

データベースに定義されたワーク・アクション・セットには、しきい値を指定するワーク・アクションを含めることができます。

以下のしきい値がサポートされています。

- 集約しきい値:
  - CONCURRENTDBCOORDACTIVITIES
  - CONCURRENTWORKLOADACTIVITIES
- アクティビティーしきい値:
  - SQLTEMPSPACE
  - SQLROWSRETURNED
  - ACTIVITYTOTALTIME
  - ESTIMATEDSQLCOST
  - CPUTIME
  - SQLROWSREAD

データベース・アクティビティーに対するワーク・アクションの適用

各データベース、各サービス・スーパークラス、または各ワークロードに対して、ワーク・アクション・セットを 1 つだけ適用することができます。

データ・サーバーにサブミットされると、作業はワークロード (ユーザー定義のワークロードまたはデフォルトのワークロードのいずれか) に関連付けられ、続いてサービス・クラスにマップされます。

以下の図は、ワーク・アクションをアクティビティーに適用する方法のプロセスについて示しています。

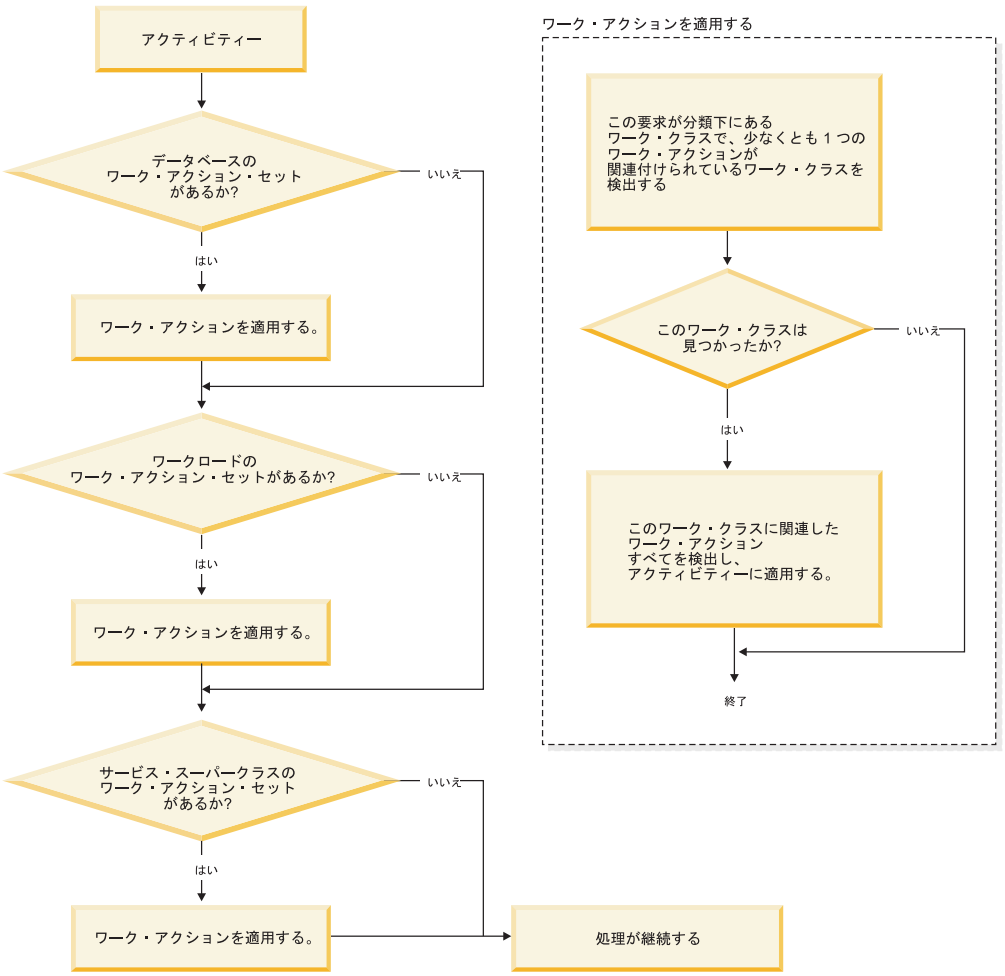


図 18. アクティビティーに対するワーク・アクションの適用

ワーク・アクションは、以下のようにアクティビティーに割り当てられます。

1. アクティビティーがサービス・スーパークラスまたはサービス・サブクラスにマップされると、データ・サーバーは使用可能なデータベース・レベルのワーク・アクション・セットが存在するかどうかを調べます。
2. 使用可能なデータベース・レベルのワーク・アクション・セットが存在する場合、データ・サーバーは、データベース・レベルのワーク・アクション・セットが関連付けられているワーク・クラス・セットのいずれかのワーク・クラスにアクティビティーが該当しているかどうかを調べます。
3. アクティビティーがワーク・クラスに該当し、そのワーク・クラスにワーク・アクションが適用されている場合は、それらのワーク・アクションがアクティビティーに適用されます。

4. データ・サーバーは、使用可能なワークロード・レベルのワーク・アクション・セットが存在するかどうかを調べます。存在する場合、データ・サーバーは、ワークロード・レベルのワーク・アクション・セットが関連付けられているワーク・クラス・セットのいずれかのワーク・クラスにアクティビティーが該当しているかどうかを調べます。
5. アクティビティーがワーク・クラスに該当し、そのワーク・クラスにワーク・アクションが適用されている場合は、それらのワーク・アクションがアクティビティーに適用されます。
6. 次に、アクティビティーがワークロードによってサービス・スーパークラスにマップされる場合、データ・サーバーはワーク・アクション・セットがサービス・スーパークラスに適用されるかどうかを調べます。
7. ワーク・アクション・セットがサービス・スーパークラスに適用される場合、データ・サーバーは、サービス・スーパークラス・レベルのワーク・アクション・セットが関連付けられているワーク・クラス・セットのいずれかのワーク・クラスにアクティビティーが該当しているかどうかを調べます。
8. アクティビティーがワーク・クラスに該当し、そのワーク・クラスにワーク・アクションが適用されている場合は、それらのワーク・アクションがアクティビティーに適用されます。

マッピング・ワーク・アクションがストアード・プロシージャーに適用されると、ワーク・アクション定義で WITH NESTED 節または WITHOUT NESTED 節のどちらが指定されているかに応じて、ストアード・プロシージャーの子アクティビティーは、親アクティビティーと同じサービス・サブクラスまたは別のサービス・サブクラスで実行できることに注意してください。

以下の場合、アクティビティーはワーク・アクション・セットによる影響を受けません。

- アクティビティーがデフォルト・システム (SYSDEFAULTSYSTEMCLASS) およびデフォルト保守 (SYSDEFAULTMAINTENANCECLASS) サービス・クラスに該当する場合。
- アクティビティーがデフォルトの管理ワークロード SYSDEFAULTADMWORKLOAD に割り当てられている場合。
- アクティビティーがロード操作の内部にある場合。ロード操作自体は、ワーク・アクション・セットの評価を受けます。
- システムのストアード・プロシージャーの子アクティビティーの場合。唯一の例外は、SYSPROC.ADMIN_CMD ストアード・プロシージャーです。SYSPROC.ADMIN_CMD の子アクティビティーは、ワーク・アクション・セットの評価を受けます。
- ワーク・アクション・セットが使用不可になっている場合。
- ワークロードがアクティビティーをサービス・サブクラスに直接マップする場合。

ワーク・アクション・セットを使用したワークロード・レベルにおける並行性の制御

ワークロード・レベルにおいて、ワークロードで定義されているワーク・アクション・セットで適用済みの並行性しきい値を使用し、入ってくる作業の並行性を制御します。ワークロード・レベルの並行性制御を、サービス・クラス・レベルの優先度変更と組み合わせることができます。

図 1 は、データベース内のすべての作業を優先度変更を使用して制御しつつ、入ってくる作業の並行性を接続のソースに基づいて制御するワーク・アクション・セットの使用法を示すシナリオ例です。

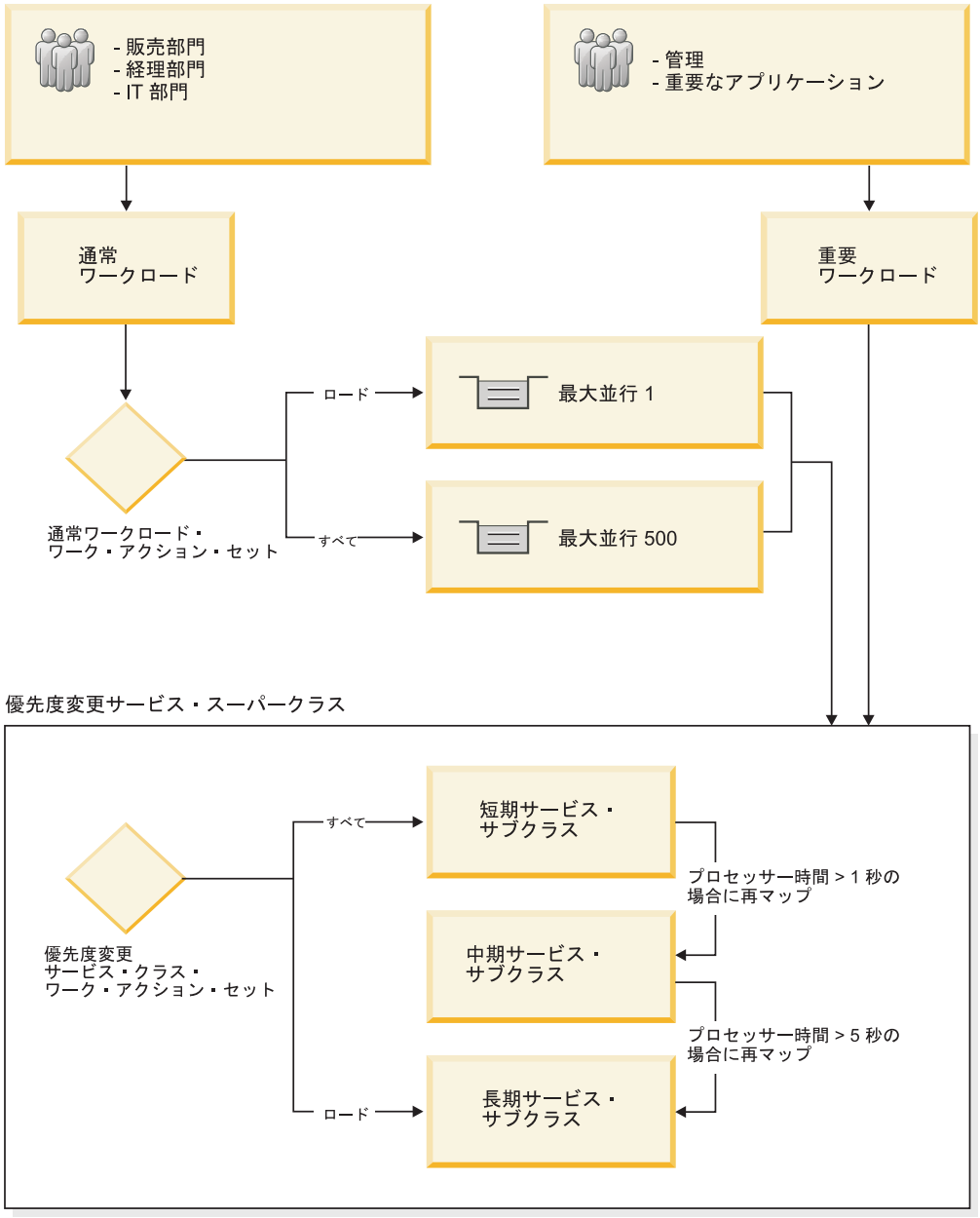


図 19. ワーク・アクション・セットを使用したワークロード・レベルにおける並行性の制御

このシナリオ例の場合、異なるソースに由来する作業を識別して区別するために、2 つのワークロードが作成されます。販売部門、経理部門、および IT 部門からのデータベースへの接続は、通常ワークロードにマップされます。管理アプリケーションと重要なアプリケーションからのデータベースへの接続は、重要ワークロードにマップされます。重要ワークロードの作業にはより高い優先度があり、最短の時間内に完了させる必要があります。重要ワークロード内の作業に関してデータベースが確実に十分な能力を有しているようにするため、通常ワークロード内の作業に並行性しきい値が設定されます。通常ワークロード・レベル・ワーク・アクション・セットというワークロード・レベルのワーク・アクション・セットが通常ワークロードに作成され、ワーク・クラス・セットに適用されます。このワーク・クラス・セットには 2 つのワーク・クラスがあります。ロード・アクティビティーが 1 つのワーク・クラスにマップされ、他のすべてのアクティビティーはもう一方のワーク・クラスにマップされます。通常ワークロード・レベル・ワーク・アクション・セットのワーク・アクションとして CONCURRENTDBCOORDACTIVITIES しきい値が作成され、他のロード・アクティビティーがキューに入れられる一方で、一度に 1 つのロード・アクティビティーだけがシステム内で許可されることになります。また、通常ワークロード・レベル・ワーク・アクション・セットのワーク・アクションとして別の CONCURRENTDBCOORDACTIVITIES しきい値が作成され、最大で 500 の並行アクティビティーが許可され、この最大数を超えるアクティビティーはキューに入れられます。

注: キューイングしきい値に対して CONTINUE しきい値アクションを指定した場合は、キューのサイズとして指定されている強制値に関係なくキューのサイズが事実上無制限になります。

データベースに対する通常ワークロードと重要ワークロードのどちらからの接続も、優先度変更サービス・スーパークラスにマップされます。このサービス・スーパークラスは、短期アクティビティーを優遇する優先度変更を実装するために作成されます。優先度変更サービス・スーパークラスに優先度変更サービス・クラス・ワーク・アクション・セットが作成され、すべての短時間実行アクティビティーと長時間実行ロード・アクティビティーを分離します。ロード・アクティビティー以外のすべてのアクティビティーは、短期サービス・サブクラスにマップされます。短期サービス・サブクラスは、エージェント、プリフェッチ、およびバッファー・プールに関して最も高い優先度を持つように構成されます。短期サービス・サブクラスに CPUTIMEINSC しきい値が作成され、短期サービス・サブクラスのアクティビティーに 1 秒より長くプロセッサー時間がかかると、そのアクティビティーは中期サービス・サブクラスに再マップされます。中期サービス・サブクラスには、エージェント、プリフェッチ、およびバッファー・プールに関して中位の優先度があります。中期サービス・サブクラスに CPUTIMEINSC しきい値が作成され、中期サービス・サブクラスのアクティビティーに 5 秒より長くプロセッサー時間がかかると、そのアクティビティーは長期サービス・サブクラスに再マップされます。長期サービス・サブクラスには、エージェント、プリフェッチ、およびバッファー・プールに関して最も低い優先度があります。ロード・アクティビティーは、優先度変更サービス・クラス・ワーク・アクション・セットによって長期サービス・サブクラスに直接マップされます。ロード・アクティビティーは長期実行され、リソースを集中的に消費する可能性があるものの、完了までのスピードはさほど重視されないためです。

ワークロードとワーク・アクション・セットの比較

データベース・アクティビティーに対してどのようなタイプの制御を保持したいかに応じて、ワークロードのみを使用して、あるいはワークロードとワーク・クラス (ワーク・アクションと併用する場合) の両方を使用して、アクティビティーをサービス・クラスにマップすることができます。

ワークロードを使用した場合、要求の識別とサービス・クラスへの割り当ては接続属性に基づいて行われます。ワークロードは、作業を特定の DB2 サービス・クラスにルーティングして実行するための主な方法です。要求を識別する方法をさらに詳細化する必要がある場合は、ワーク・クラスを使用して、そのタイプおよび他のアクティビティー属性に基づいてアクティビティーを分類することができます。例えば、READ アクティビティー、WRITE アクティビティー、および LOAD アクティビティーを異なるワーク・クラスに分類し、アクティビティー・タイプごとに扱いを別にすることができます。

ワーク・クラス (ワーク・クラス・セットにグループ化された) を使用する場合、ワーク・アクションを使用してさまざまなタイプのアクティビティーを制御することができます。例えば、サービス・スーパークラスで定義されているワーク・アクション・セットの中の 1 つのワーク・アクションを使用して、ある特別なタイプのアクティビティーを、あるサービス・サブクラスにマップすることができます。データベースまたはワークロードで定義されているワーク・アクション・セットでは、ワーク・アクションを定義して、同じタイプのアクティビティーが特定の条件を超えないよう、しきい値という制御を適用することができます。

ワーク・アクションはワーク・アクション・セットにグループ化されます。単一のワーク・アクション・セットは、データベース内のアクティビティー、サービス・スーパークラス内のアクティビティー、またはワークロード内のアクティビティーに適用できます。ただし、同じワーク・アクション・セットを複数のオブジェクトに適用することはできません。ワーク・クラス・セットとワーク・アクション・セットは一緒に動作します。つまり、アクティビティーを特定のタイプの作業として分類するためのワーク・クラスがあらかじめ存在していなければ、ワーク・アクションをワーク・クラスに適用することはできません。 1 つのワーク・クラス・セットは複数のワーク・アクション・セットに関連付けることができますが、1 つのワーク・アクション・セットは 1 つのワーク・クラス・セットにだけ関連付けることができます。

図 1 は、ワークロードおよびワーク・アクション・セットを使用する DB2 ワークロード・マネージャーのインプリメンテーションの例を示しています。この図では、要求をサブミットした接続の接続属性に基づいて、要求がワークロード WL_A に割り当てられると仮定します。ワークロード WL_A は、要求がサービス・スーパークラス SC_A で実行されることを指定します。ワーク・クラス・セット WCS_1 のワーク・クラスは、ワークロード WL_A に関連した要求が実行する作業タイプと一致すると仮定します。

ここで、カタログを更新しないアクティビティー (READ アクティビティー) がシステムに入ってくると仮定します。データベース・レベルのワーク・アクション・セット WAS_1 (ワーク・クラス・セット WCS_1 と関連付けられている) には、READ ワーク・クラスに適用されるワーク・アクションが含まれています。要求はサービス・スーパークラス SC_A に (ワークロード WL_A によって) マップされ

ます。ここで要求は、サービス・スーパークラス・レベルのワーク・アクション・セット WAS_2 を検出します。これもまたワーク・クラス・セット WCS_1 に関連付けられており、サービス・スーパークラス SC_A 内のアクティビティーに適用されます。このワーク・アクション・セットには、マッピング・ワーク・アクションが含まれており、これも READ ワーク・クラスに適用され、これによってすべての READ アクティビティーは、サービス・スーパークラス SC_A 内のサービス・サブクラス SSC_1a にマップされます。

これにやや似た状態が、ワークロード WL_B に関連付けられた要求 (この場合もその接続属性に基づいて関連付けられる) でも発生します。ワークロード WL_B はアクティビティーをサービス・スーパークラス SC_B にマップします。この要求は LOAD アクティビティーに関するものであり、ワーク・クラス・セット WCS_2 には LOAD アクティビティーに当てはまるワーク・クラスが含まれていると仮定します。ワーク・クラス・セット WCS_2 はサービス・スーパークラス・レベルのワーク・アクション・セット WAS_3 に関連付けられ、これはサービス・スーパークラス SC_B のアクティビティーに適用されます。ワーク・アクション・セット WAS_3 に、LOAD ワーク・クラスに適用されているマッピング・ワーク・アクションが含まれていると仮定すると、LOAD アクティビティーがワークロード WL_B によってサービス・スーパークラス SC_B にマップされる場合、それはワーク・アクションによって実行のためにサービス・サブクラス SSC_1b にマップされます。

この例におけるワークロード WL_C の目的は、サービス・スーパークラス・レベルのワーク・アクション・セット WAS_3 とそのマッピング・ワーク・アクションとは関わりなく、着信要求を直接にサービス・サブクラス SSC_1b にマップすることです。着信要求が LOAD アクティビティーであるワークロード WL_C に関連付けられる場合、この要求は実行のためにサービス・サブクラス SSC_1b にも直接にマップされ、LOAD ワーク・クラスに適用されるマッピング・ワーク・アクションの影響を受けません。

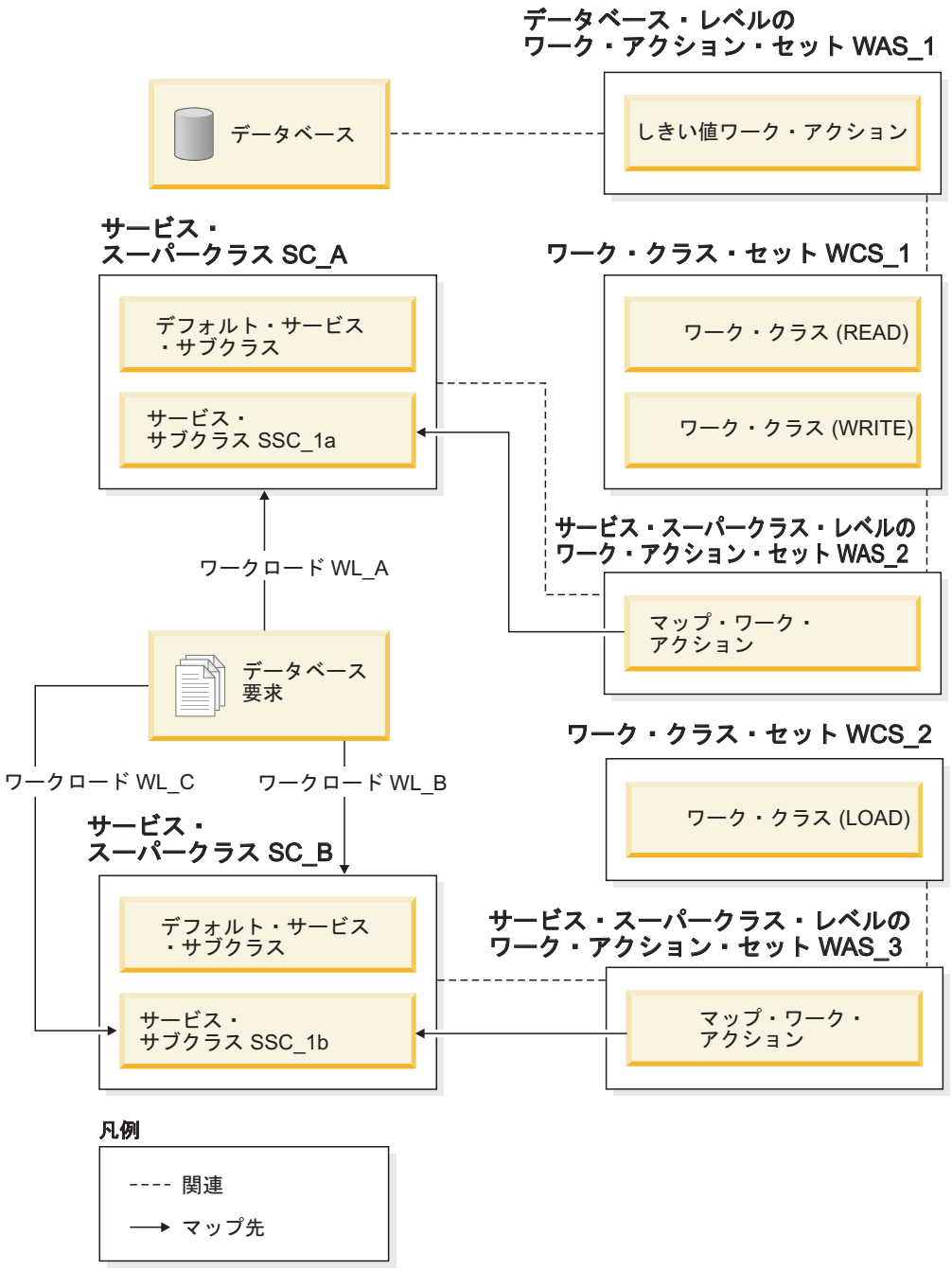


図 20. ワークロードとワーク・アクション・セット

例: データベース・ワーク・アクション・セットおよびデータベースしきい値の使用

以下の例では、DB2 アクティビティーによって消費されるリソースを制御する、ワーク・アクション・セットおよびしきい値のさまざまな使用方法を示します。DB2 ワークロード管理オブジェクトを作成する前に、それらの使用方法について理解しておく必要があります。

ALLSQL というワーク・クラス・セットがあり、それに以下のワーク・クラスが以下に示す順序で含まれているとします。

1. SMALLDML (見積コストが 1 000 timeron より小さいすべての DML タイプ SQL ステートメント用)
2. MEDDML (見積コストが 1 000 timeron から 20 000 timeron までの、すべての DML タイプ SQL ステートメント用)
3. LARGEDML (見積コストが 20 000 timeron より大きいすべての DML タイプ SQL ステートメント用)
4. ALLDDL (すべての DDL タイプ SQL ステートメント用)
5. ALLACTIVITY (すべてのデータベース・アクティビティー用)

以下の SQL ステートメントは、ワーク・クラス・セットおよびワーク・クラスを作成します。

```
CREATE WORK CLASS SET ALLSQL
  (WORK CLASS SMALLDML WORK TYPE DML FOR TIMERONCOST FROM 0 TO 1000,
   WORK CLASS MEDDML WORK TYPE DML FOR TIMERONCOST FROM 1001 TO 20000,
   WORK CLASS LARGEDML WORK TYPE DML FOR TIMERONCOST FROM 20001 TO UNBOUNDED,
   WORK CLASS ALLDDL WORK TYPE DDL,
   WORK CLASS ALLACTIVITY WORK TYPE ALL)
```

これらのワーク・クラスには既に、COUNT ACTIVITY、COLLECT、およびしきい値 (ACTIVITYTOTALTIME しきい値ではない) などのワーク・アクションが適用されています。

大きな DML アクティビティーが 5 時間を超えて実行することを禁止したいものとします。他のすべての SQL は 30 分を超えて実行することができません。以下の 2 つの例は、この目標を達成するために取り得る方法を示しています。

方法 1

1 つの方法は、各ワーク・クラスに ACTIVITYTOTALTIME しきい値を指定するワーク・アクションが入ったワーク・アクション・セットを、データベース・レベルでセットアップすることです。以下のようにします。

表 33. 各ワーク・クラスに指定された ACTIVITYTOTALTIME しきい値

ワーク・アクション	適用される ワーク・クラス	しきい値タイプおよび値	アクション
SMALLDMLTIMEALLOWED	SMALLDML	ACTIVITYTOTALTIME > 30 MINUTES	• 実行の停止 • アクティビティー・データの収集
MEDDMLTIMEALLOWED	MEDDML	ACTIVITYTOTALTIME > 30 MINUTES	• 実行の停止 • アクティビティー・データの収集
LARGEDMLTIMEALLOWED	LARGEDML	ACTIVITYTOTALTIME > 5 HOURS	• 実行の停止 • アクティビティー・データの収集

表 33. 各ワーク・クラスに指定された *ACTIVITYTOTALTIME* しきい値 (続き)

ワーク・アクション	適用される ワーク・クラス	しきい値タイプおよび値	アクション
ALLDDLTIMEALLOWED	ALLDDL	ACTIVITYTOTALTIME > 30 minutes	• 実行の停止 • アクティビティー・データの収集
ALLACTIVITYTIMEALLOWED	ALLACTIVITY	ACTIVITYTOTALTIME > 30 minutes	• 実行の停止 • アクティビティー・データの収集

この方法での SQL ステートメントは、以下のとおりです。

```
CREATE WORK ACTION SET WASNICK FOR DATABASE USING WORK CLASS SET WCSNICK
  (WORK ACTION SMALLDMLTIMEALLOWED ON WORK CLASS SMALLDML
     WHEN ACTIVITYTOTALTIME > 30 MINUTES COLLECT ACTIVITY DATA STOP EXECUTION,
   WORK ACTION MEDDMLTIMEALLOWED ON WORK CLASS MEDDML
      WHEN ACTIVITYTOTALTIME > 30 MINUTES COLLECT ACTIVITY DATA STOP EXECUTION,
   WORK ACTION LARGEDMLTIMEALLOWED ON WORK CLASS LARGEDML
      WHEN ACTIVITYTOTALTIME > 5 HOURS COLLECT ACTIVITY DATA STOP EXECUTION,
   WORK ACTION ALLDDLTIMETIMEALLOWED ON WORK CLASS ALLDDL
      WHEN ACTIVITYTOTALTIME > 30 MINUTES COLLECT ACTIVITY DATA STOP EXECUTION,
   WORK ACTION ALLACTIVITYTIMEALLOWED ON WORK CLASS ALLACTIVITY
      WHEN ACTIVITYTOTALTIME > 30 MINUTES COLLECT ACTIVITY DATA STOP EXECUTION)
```

方法 2

もう 1 つの方法として考えられるのは、LARGEDML というワーク・クラスを 1 つだけ使用し、次いでこのワーク・クラスに適用される LARGEDMLTIMEALLOWED という 1 つのワーク・アクションを持ったワーク・アクション・セットをデータベースのために作成するというものです。

表 34. *LARGEDML* ワーク・クラスに適用される *LARGEDMLTIMEALLOWED* ワーク・アクション

ワーク・アクション	適用される ワーク・クラス	しきい値タイプおよび値	アクション
LARGEDMLTIMEALLOWED	LARGEDML	ACTIVITYTOTALTIME < 5 HOURS	• 実行の停止 • アクティビティー・データの収集

次に、31 MINUTES 未満という ACTIVITYTOTALTIME しきい値をデータベースに適用します。この方法を使用した場合、LARGEDML ワーク・クラスに対応するアクティビティーにだけ 5 時間のしきい値が適用されます。他のアクティビティーには、31 分未満という ACTIVITYTOTALTIME データベース時間しきい値が適用されることになります。

この方法での SQL ステートメントは、以下のとおりです。

```
CREATE WORK ACTION SET WASNICK FOR DATABASE USING WORK CLASS SET WCSNICK
  (WORK ACTION LARGEDMLTIMEALLOWED ON WORK CLASS LARGEDML
     WHEN ACTIVITYTOTALTIME > 5 HOURS COLLECT ACTIVITY DATA STOP EXECUTION)

CREATE THRESHOLD THTEST FOR DATABASE ACTIVITIES ENFORCEMENT DATABASE
   WHEN ACTIVITYTOTAL TIME > 30 MINUTES COLLECT ACTIVITY DATA STOP EXECUTION
```

例: ワーク・アクション・セットを使用して実行中の作業のタイプを判別する

ワーク・クラス・セット、ワーク・クラス、ワーク・アクション・セット、ワーク・アクション、および DB2 ワークロード・マネージャー・モニター・フィーチャーの一部を使用して、システム上で実行されているさまざまな種類の作業、およびその作業の分布を判別することができます。

ここでは、このタスクを実現する 1 つの方法が説明されています。まず対象となるさまざまな作業タイプのためのワーク・クラスの入った、ワーク・クラス・セットを作成します。例えば、システムで実行中の READ アクティビティー、WRITE アクティビティー、DDL アクティビティー、および LOAD アクティビティーがいくつあるかを知りたい場合、以下の例にあるようにワーク・クラス・セット、ACTIVITYTYPES を作成します。

```
CREATE WORK CLASS SET ACTIVITYTYPES
(WORK CLASS READWC WORK TYPE READ,
WORK CLASS WRITEWC WORK TYPE WRITE,
WORK CLASS DDLWC WORK TYPE DDL,
WORK CLASS LOADWC WORK TYPE LOAD)
```

次に、データベース・レベルのワーク・アクション・セット COUNTACTIONS を作成して、ACTIVITYTYPES ワーク・クラス・セットに適用します。そのワーク・アクション・セットには、以下の例にあるように、ACTIVITYTYPES ワーク・クラス・セットにあるそれぞれのワーク・クラスに対する COUNT ACTIVITY ワーク・アクションが含まれます。

```
CREATE WORK ACTION SET COUNTACTIONS FOR DATABASE USING WORK CLASS SET ACTIVITYTYPES
(WORK ACTION COUNTREAD ON WORK CLASSREADWC COUNT ACTIVITY,
WORK ACTION COUNTWRITE ON WORK CLASS WRITEWC COUNT ACTIVITY,
WORK ACTION COUNTDDL ON WORK CLASS DDLWC COUNT ACTIVITY,
WORK ACTION COUNTLOAD ON WORK CLASS LOADWC COUNT ACTIVITY)
```

十分な時間が経過した後、WLM_GET_WORK_ACTION_SET_STATS 表関数を使用して実行されているタイプごとのアクティビティーの数を判別することができます。

```
SELECT SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
LAST_RESET,
SUBSTR(WORK_CLASS_NAME,1,15) AS WORK_CLASS_NAME,
SUBSTR(CHAR(ACT_TOTAL),1,14) AS TOTAL_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS(CAST(NULL AS VARCHAR(128)), -2))
AS WASSTATS WHERE WORK_ACTION_SET_NAME = 'COUNTACTIONS'
ORDER BY WORK_CLASS_NAME, MEMB
```

しきい値による作業の制御

異常な動作をする作業を特定することによって、システムの安定度を維持するために、しきい値を使用できます。予測影響に基づいて、作業の実行を開始する前に、異常な動作を予測的に特定できます。また、作業の実行中にリソースが消費されていく際に発生する異常な動作を敏感に感知し、特定することもできます。

例えば、大量のプロセッサー時間を消費するためにシステム上の他のすべての作業の実行を犠牲にしている照会は、しきい値を使用して制御できる作業の一例です。

このような照会に対する制御は、照会が実行を開始する前でも見積コストに基づいて行えます。または、照会が実行を開始して許容量を超えるリソースを消費しているときに行えます。

しきい値のタイプ

接続しきい値

接続の最大アイドル時間を制限する場合、接続しきい値を使用します。これらのしきい値は、アイドル状態が長すぎる接続の検出に使用できます。

表 35. 接続しきい値

しきい値	説明
CONNECTIONIDLETIME	接続がアイドル状態でユーザー要求のための動作をしていない時間の長さを制御します。このしきい値は、データ・サーバーのリソースを効率的に使用していない状態と、アプリケーションの待ち状態を検出するために使用します。

作業単位しきい値

作業単位の実行時間を制限する場合、作業単位しきい値を使用します。これらのしきい値は、作業単位が DB2 エンジンで費やすことができる最大時間を制限します。これは、完了までの時間が長すぎる作業単位を検出するために使用できます。

表 36. 作業単位しきい値

しきい値	説明
UOWTOTALTIME	作業単位の実行にかかる時間の長さを制御します。

アクティビティーしきい値

データ・サーバーの実行に特定のアクティビティーが与える影響を制限する場合は、アクティビティーしきい値を手段の 1 つとして使用できます。実行時間が長すぎる、異常に大量のデータが返される、または異常に大量のリソースが消費されるという現象はすべて、潜在的問題を含んだアクティビティーが過剰のリソースを消費している可能性があるという警告標識の例です。こうしたアクティビティーは、アクティビティーしきい値で制御できます。

表 37. アクティビティーしきい値

しきい値	説明
ACTIVITYTOTALTIME	特定のアクティビティーがサブミットされてから完了までに費やすことができる時間 (実行時間とキュー時間) の長さを制御します。このしきい値は、完了までに異常に長い時間がかかっているジョブを検出するために使用します。
CPUTIME	アクティビティーが実行中に特定のメンバーで消費できるユーザー・プロセッサー時間とシステム・プロセッサー時間の合計最大時間を制御します。このしきい値は、過度のプロセッサー・リソースを消費しているアクティビティーを検出および制御するために使用します。
CPUTIMEINSC	アクティビティーが特定のサービス・サブクラスで実行中に特定のメンバーで消費できるユーザー・プロセッサー時間とシステム・プロセッサー時間の合計最大時間を制御します。このしきい値は、過度のプロセッサー・リソースを消費している現行サービス・クラス内のアクティビティーを検出および制御するために使用します。

表 37. アクティビティーしきい値 (続き)

しきい値	説明
DATATAGINSC	特定のサービス・サブクラスを実行しているときに、アクティビティーによって触ることのできるデータ、または触ることができないデータを制御します。
ESTIMATEDSQLCOST	照会オプティマイザーによって見積コストが大きいと判断される DML アクティビティーを制御します。このしきい値は、リソースを大量に消費する可能性がある SQL をシステムでの実行開始前に予測し、書き方が不適切な SQL を識別するために使用します。
SQLROWSREAD	アクティビティーがどのメンバーにおいても読み取ることができる行の最大数を制御します。このしきい値は、過度の行数を読み取っているアクティビティーを検出および制御するために使用します。
SQLROWSREADINSC	アクティビティーが特定のサービス・サブクラスで実行中に特定のメンバーで読み取ることができる行の最大数を制御します。このしきい値は、過度の行数を読み取っている現行サービス・クラス内のアクティビティーを検出および制御するために使用します。
SQLROWSRETURNED	SQL を実行する時に返される行数を制御します。このしきい値は、データ量が妥当なボリュームを超えたときを識別するために使用します。
SQLTEMPSPACE	特定のアクティビティーがメンバー上で消費できる TEMPORARY 表スペースの量を制御します。このしきい値は、特定の SQL ステートメントが過度の量の一時スペースを使い果たして他の作業の進行を妨げることがないようにするために使用します。

データ・サーバーは、REORGCHK、IMPORT、および EXPORT などのユーティリティーからの要求を、ユーザー・ロジックと見なします。したがってこれらの要求は、定義済みしきい値に制約されます。

集約しきい値

データ・サーバーに特定のアクティビティー、ワークロード、または接続が与える集合的な影響を制御する場合は、集約しきい値を定義できます。集約しきい値は多くの場合 (常にというわけでない)、同時に実行される特定のアクティビティーの数を制限する必要がある場合に並行性制御を実施します。いくつかの集約しきい値は組み込みキューを持っており、キューイングしきい値とも呼ばれます。

表 38. 集約しきい値

しきい値	説明
AGGSQLTEMPSPACE	サービス・サブクラス内のすべてのアクティビティーで消費できる SYSTEM TEMPORARY 表スペースの合計最大量を制御します。このしきい値は、すべてのアクティビティーが全体として過度の量の SYSTEM TEMPORARY 表スペースを消費しているサービス・サブクラスに属するアクティビティーを検出および制御するために使用します。
CONCURRENTWORKLOADOCCURRENCES	コーディネーター・メンバーで同時に実行できるワークロードのアクティブなオカレンスの数を制御します。特定のソースからの接続の広がりを制御するために使用します。
CONCURRENTWORKLOADACTIVITIES	1 つのワークロード・オカレンスの中で実行できる個々のアクティビティーの数を制御します。個々のワークロード・オカレンスの中での作業を制限するために使用します。

表 38. 集約しきい値 (続き)

しきい値	説明
CONCURRENTDBCOORDACTIVITIES	しきい値が関連しているドメイン内の並行アクティビティー (データベース、ワーク・アクション、サービス・スーパークラス、またはサービス・サブクラス) の数を制御します。
TOTALMEMBERCONNECTIONS	特定のメンバーに対して同時に確立できるデータベース接続の数を制御します。特定のメンバーが過負荷とならないようにするために使用します。
TOTALSCMEMBERCONNECTIONS	特定のサービス・クラスの中で実行される作業用の特定のメンバーに対して同時に確立できるデータベース接続の数を制御します。メンバー接続の合計数に似ていますが、接続がサービス・クラスとリンクされているため、細分性は高くなります。

これをサポートする集約しきい値の場合、実行「チケット」のシステムにより並行性の制御が提供されます。各着信アクティビティーは、実行を開始する前に、該当する並行性しきい値にチケットを請求しなければなりません。チケットをすべて使い尽くすと、しきい値を定義した方法に応じて、追加のアクティビティーはチケットが使用可能になるまでキューに入れられるか、エラーが戻されます。並行性しきい値でキューイングが使用可能になっている場合は、実行を完了したアクティビティーからキューに入れられている別のアクティビティーにチケットが渡されます。その後、そのアクティビティーはキューから出て実行を開始できます。並行性しきい値当たりの使用可能なチケットの数は、しきい値を定義した方法に応じて異なります。例えば、CONCURRENTDBCOORDACTIVITIES しきい値を定義して、データベース・アクティビティーの並行実行数を 10 に制限すると、10 個の実行チケットが使用可能になります。

ストアード・プロシージャーの場合、アクティビティーしきい値は、ストアード・プロシージャー自体とその子アクティビティーに適用されます。並行性しきい値は、CALL ステートメント自体にではなく、ストアード・プロシージャーの子アクティビティーにのみ適用されます。

しきい値違反が発生した場合のアクション

しきい値違反が発生した場合に動的に取るアクションは、そのしきい値がどのように定義されているかによって異なります。

実行の停止 (STOP EXECUTION)

しきい値の違反が発生した場合に取る一般的なアクションは、アクティビティーの実行を停止することです。この場合、サブミットしたアプリケーションに対して、しきい値に違反したことを示すエラー・コードが返されます。TOTALMEMBERCONNECTIONS および TOTALSCMEMBERCONNECTIONS しきい値の場合、STOP EXECUTION アクションを使用すると接続が確立されなくなります。CONNECTIONIDLETIME しきい値の場合は、接続が閉じられます。CONCURRENTWORKLOADOCCURRENCES の場合は、新規のワークロード・オカレンスが作成されなくなります。アクティビティー関連のすべてのしきい値については、アクティビティーが実行を続行しなくなります。

THRESHOLDVIOLATIONS イベント・モニターがアクティブな場合は、しきい値に違反したことを示すレコードがイベント・モニターに書き込まれます。

実行の続行 (CONTINUE)

状況によっては、アクティビティーの実行を停止するのは、対応として行き過ぎである場合もあります。望ましい対応は、アクティビティーには実行を続行させ、この状態が再発しないようにする方法を判別するための分析を将来行えるように、管理者用の関連データを収集することです。この状況では、サブミットしたアプリケーションにエラー・コードは返されません。アクションを続行する場合、ユーザーはしきい値に違反したという通知は受け取りません。THRESHOLDVIOLATIONS イベント・モニターがアクティブな場合は、レコードがイベント・モニターに書き込まれます。キューイングしきい値に対して CONTINUE しきい値アクションを指定した場合は、存在する強制値に関係なくキューのサイズが事実上無制限になります。

アプリケーションの強制終了 (FORCE APPLICATION)

UOWTOTALTIME しきい値に違反する場合、ローカルまたはリモートのユーザーまたはアプリケーションをシステムから強制的に切り離すように指定することができます。これは、サーバー・リソースを求めて競合している他のアプリケーションにとってのメリットとなります。

アクティビティーの再マップ (REMAP ACTIVITY TO)

アクティビティーが特定の制限に違反した場合、そのアクティビティーに別のリソース制御を割り当ててアクティビティーの実行はそのまま続行させることもできます。こうした対応にすると、アクティビティーがその存続期間中に消費できるリソースの量を、動的に増やしたり減らしたりできます。この場合、既に実行中のアクティビティーは、しきい値に違反したことがユーザーやアプリケーションに示されることなく続行できます。ただし、そのアクティビティーは、使用可能な別のリソースを使用して実行されるようになります。再マップは、CPUTIMEINSC、SQLROWSREADINSC、DATATAGINSC のようなサービス・クラス内のしきい値であれば、どれに関しても使用可能です。

データの収集 (COLLECT ACTIVITY DATA)

一部のしきい値では、違反が生じるとデータが収集されます。デフォルトでは、アクティビティーしきい値に違反したという事実が、アクティブにされたしきい値違反イベント・モニターに記録されます。しきい値に違反したアクティビティーに関するより詳細な情報が必要な場合は、COLLECT ACTIVITY DATA 節を使用することで、そのアクティビティーに関する情報をアクティビティー用のアクティブなイベント・モニターに書き込むよう要求できます。この書き込みは、そのアクティビティーの実行完了時に行われます。

しきい値のドメインと適用範囲

各しきい値は、ドメインに対して作用します。しきい値の影響を受ける可能性があるのは、そのしきい値のドメイン内で起こるアクティビティーのみです。

以下のしきい値のドメインが存在します。

- データベース

- サービス・スーパークラス
- サービス・サブクラス
- ワーク・アクション
- ワークロード
- ステートメント

これらの各しきい値ドメイン内のしきい値には有効範囲があり、それを超えるとしきい値が強制可能になります。例えば、単一のワークロード・オカレンス、1 つのメンバー、特定のステートメントの実行、またはすべてのメンバーなどです。これは、しきい値の*適用範囲* として知られています。例: サービス・クラスの集約しきい値では、2 つの適用範囲 (データベースおよびメンバー) のうちの 1 つが可能です。メンバー・レベルでのみ適用される集約しきい値の例は、メンバー上のサービス・スーパークラスに対する同時接続の最大数です (TOTALSCMEMBERCONNECTIONS)。同様に、以下の表は、データベース、スーパークラス、サブクラス、ワーク・アクション、ワークロード・ドメインでプロセッサー時間のしきい値 (CPUTIME) を指定することができ、そのしきい値がメンバーごとに強制されることを示しています。つまり、上限は、アクティビティーが使用できるメンバーごとのユーザーおよびシステム・プロセッサー時間の上限を指定します。

表 39. しきい値のドメインと適用範囲

しきい値のドメイン	適用範囲: データベース	適用範囲: メンバー	適用範囲: ワークロード・オカレンス
データベース	• 155 ページの『ACTIVITYTOTALTIME しきい値』 • 166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』[1] • 153 ページの『CONNECTIONIDLETIME しきい値』 • 160 ページの『ESTIMATEDSQLCOST しきい値』 • 163 ページの『SQLROWSRETURNED しきい値』 • 173 ページの『UOWTOTALTIME しきい値』	• 166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』(データベース適用範囲は、DB2 pureScale® 環境以外の環境のみで許可されています。 DB2 pureScale 環境では、メンバー適用範囲のみが許可されています。) • 156 ページの『CPUTIME しきい値』 • 160 ページの『SQLROWSREAD しきい値』 • 164 ページの『SQLTEMPSPACE しきい値』 • 171 ページの『TOTALMEMBERCONNECTIONS しきい値』	適用されない
ワーク・アクション	• 155 ページの『ACTIVITYTOTALTIME しきい値』 • 166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』[1] • 160 ページの『ESTIMATEDSQLCOST しきい値』 • 163 ページの『SQLROWSRETURNED しきい値』	• 166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』[1] • 156 ページの『CPUTIME しきい値』 • 160 ページの『SQLROWSREAD しきい値』 • 164 ページの『SQLTEMPSPACE しきい値』	適用されない
サービス・スーパークラス	• 155 ページの『ACTIVITYTOTALTIME しきい値』 • 166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』[1] • 153 ページの『CONNECTIONIDLETIME しきい値』 • 160 ページの『ESTIMATEDSQLCOST しきい値』 • 163 ページの『SQLROWSRETURNED しきい値』 • 173 ページの『UOWTOTALTIME しきい値』	• 166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』[1] • 156 ページの『CPUTIME しきい値』 • 160 ページの『SQLROWSREAD しきい値』 • 164 ページの『SQLTEMPSPACE しきい値』 • 172 ページの『TOTALSCMEMBERCONNECTIONS しきい値』	適用されない
サービス・サブクラス	• 155 ページの『ACTIVITYTOTALTIME しきい値』 • 166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』[1] • 160 ページの『ESTIMATEDSQLCOST しきい値』 • 163 ページの『SQLROWSRETURNED しきい値』	• 165 ページの『AGGSQLTEMPSPACE しきい値』 • 166 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』[1] • 156 ページの『CPUTIME しきい値』 • 157 ページの『CPUTIMEINSC しきい値』 • 158 ページの『DATATAGINSC しきい値』 • 160 ページの『SQLROWSREAD しきい値』 • 162 ページの『SQLROWSREADINSC しきい値』 • 164 ページの『SQLTEMPSPACE しきい値』	適用されない

表 39. しきい値のドメインと適用範囲 (続き)

しきい値のドメイン	適用範囲: データベース	適用範囲: メンバー	適用範囲: ワークロード・オカレンス
ワークロード	• 155 ページの『ACTIVITYTOTALTIME しきい値』 • 160 ページの『ESTIMATEDSQLCOST しきい値』 • 163 ページの『SQLROWSRETURNED しきい値』 • 173 ページの『UOWTOTALTIME しきい値』	• 170 ページの『CONCURRENTWORKLOADOCCURRENCES しきい値』 • 156 ページの『CPUTIME しきい値』 • 160 ページの『SQLROWSREAD しきい値』 • 164 ページの『SQLTEMPSPACE しきい値』	• 168 ページの『CONCURRENTWORKLOADACTIVITIES しきい値』
ステートメント	• 155 ページの『ACTIVITYTOTALTIME しきい値』 • 160 ページの『ESTIMATEDSQLCOST しきい値』 • 163 ページの『SQLROWSRETURNED しきい値』	• 156 ページの『CPUTIME しきい値』 • 160 ページの『SQLROWSREAD しきい値』 • 164 ページの『SQLTEMPSPACE しきい値』	適用されない

しきい値の評価順序

1 つのデータベースに関して定義されたしきい値は、特定の順序で評価されます。

以下のしきい値は、他のすべてのしきい値より先に評価されます。

- TOTALMEMBERCONNECTIONS。このしきい値は、データベースに新規の接続が行われたときに評価されます。
- CONCURRENTWORKLOADOCCURRENCES。このしきい値は、このしきい値の適用対象であるワークロード定義で新規ワークロード・オカレンスが開始したときに評価されます。
- TOTALSCMEMBERCONNECTIONS。このしきい値は、接続がサービス・クラスに割り当てられるとき (新規の接続、またはワークロードの再割り当ての結果としてのサービス・クラス間の転送のどちらか) に、評価されます。

他のすべてのしきい値は、SQL ステートメントの結果またはロード・ユーティリティーなどのユーティリティーの実行結果による認識されたアクティビティーに基づいており、次の順序で評価されます。

1. 『予測しきい値』
2. 148 ページの『反応的しきい値』

予測しきい値

予測しきい値は、反応的しきい値よりも前に検査されます。予測しきい値は、データベース・アクティビティーが実行を開始できるかどうかに影響するからです。

予測しきい値が評価される順序は以下のとおりです。特定のしきい値を定義していない場合、このステップは省略されます。パフォーマンス上の理由で、実行時に、説明されているステップがまとめられる可能性もあります。

1. CONCURRENTWORKLOADACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。当てはまる場合は、次のステップへ進みます。
2. ESTIMATEDSQLCOST しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。このしきい値を複数のドメイン内に定義すると、しきい値は有効範囲の解決の規則に従って解決されます (詳細については、 154 ページの『アクティビティーしきい値のドメイン優先順位』を参照して

ください)。この操作の結果は、そのアクティビティーに適用可能な ESTIMATEDSQLCOST 値です。しきい値に違反している場合は、対応するアクションが実行されます。

3. ワークロード・ワーク・アクション・セットのしきい値のドメインに関して CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。
4. データベース・ワーク・アクション・セットのしきい値のドメインに関して CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。
5. サービス・サブクラスのしきい値のドメインに関して CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。
6. サービス・スーパークラスのしきい値のドメインに関して CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。
7. データベースのしきい値のドメインに関して CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。

並行性しきい値に関する考慮事項: 並行性しきい値の評価順序は、アクティビティーしきい値の解決に使用される階層に従いません。アクティビティーは、定義された各並行性しきい値を通過することで実行を許可されます。

並行性しきい値の場合、ワークロード・レベルのワーク・アクション・セットのしきい値が最初に検査され、データベース・レベルのワーク・アクション・セットが 2 番目に検査されます。ワーク・アクション・セットのしきい値が最初に検査されるのは、特定タイプの作業のワーク・アクション・セットのしきい値が他のタイプの作業をブロックして並行性に影響を与えるのを回避するためです。例えば、データベース・レベルのワーク・アクション・セットの並行性しきい値を最初に検査することにより、次のような状態が回避されます。

以下のしきい値が定義されていると仮定します。

- LOAD アクティビティーに対するワーク・アクションの並行性しきい値が含まれているデータベースのワーク・アクション・セットが、1 の値で定義されています。
- サービス・スーパークラス S1 の並行性限度が 10 に設定されています。

また、1 つの LOAD アクティビティーがデータベース内 (任意のサービス・スーパークラス下) で既に実行中で、そして 9 つのアクティビティーが既にサービス・スーパークラス S1 で実行されているとします。 2 番目の新規 LOAD アクティビティーが 10 番目のアクティビティーとして入ってきます。しきい値の評価時にアクティビティーしきい値の有効範囲解決階層が使用されると、着信 LOAD アクティビティーはサービス・クラスのしきい値に違反しないことになり、並行性が 10 に増

加します。次に LOAD アクティビティーはデータベース・レベルのワーク・アクションのしきい値の並行性限度に照らして評価されますが、これは違反になります。データベースで LOAD アクティビティーが既に実行中であり、ワーク・アクションのしきい値の並行性の値は 1 に限られているからです。こうして 2 番目の LOAD アクティビティーはキューに入れられます。

サービス・スーパークラス S1 に到着する新規アクティビティーはどれもキューに入れられます (サービス・クラスの並行性限度に既に達したため)。ワーク・アクションのしきい値のキューはサービス・クラスに影響を与えますが、サービス・クラス内で実行しようとしているアクティビティーは必ずしもワーク・アクションのしきい値の条件と関係するわけではないので (例えば、サービス・スーパークラス S1 で実行しようとしている挿入操作は、データベース・レベルのワーク・アクションのしきい値の条件のためにキューに入れられた LOAD アクティビティーを待つ必要はありません)、それは望ましいことではありません。それで、こうした種類の状態を回避するために、サービス・サブクラス、サービス・スーパークラス、およびデータベースの並行性しきい値の前に、データベース・レベルのワーク・アクションの並行性しきい値が検査されます。データベース・レベルのワーク・アクション・セットの並行性しきい値が最初に検査されるので、サービス・クラス内の 10 番目のアクティビティー (ここでは LOAD アクティビティー) が、サービス・スーパークラス S1 内の 1 つの場所を占めてしまう前に、データベースのワーク・アクションのしきい値レベルでブロックされます。

反応的しきい値

反応的しきい値はアクティビティーの実行中に個々に評価され、特定の評価順序はありません。以下の反応的しきい値が使用可能です。

- ACTIVITYTOTALTIME
- AGGSQLTEMPSPACE
- CONNECTIONIDLETIME
- CPUTIME
- CPUTIMEINSC
- DATATAGINSC
- SQLTEMPSPACE
- SQLROWSREAD
- SQLROWSREADINSC
- SQLROWSRETURNED
- UOWTOTALTIME

しきい値の作成

DDL ステートメント CREATE THRESHOLD (または CREATE WORK ACTION SET ステートメント) を使用してしきい値を作成します。しきい値は、リソースの使用量に制限を設けるために作成します。

始める前に

しきい値を作成するためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』
- 命名規則

ワーク・アクション・セットのしきい値を作成するには、CREATE WORK ACTION SET ステートメントまたは ALTER WORK ACTION SET ステートメントを ADD WORK ACTION キーワードとともに使用します。詳しくは、CREATE WORK ACTION SET ステートメントまたは ALTER WORK ACTION SET ステートメントを参照してください。

手順

しきい値を作成するには、次のようにします。

1. CREATE THRESHOLD ステートメントを発行し、以下に挙げるしきい値のプロパティーを 1 つ以上指定します。
   - しきい値の名前。
   - しきい値のドメイン。しきい値のドメインは、しきい値が付加されて、しきい値が作用するデータベース・オブジェクトです。適用されるドメインは、しきい値のタイプによって異なります。詳しくは、 144 ページの『しきい値のドメインと適用範囲』を参照してください。
   - しきい値の適用範囲。しきい値の有効範囲は、ドメイン内でのしきい値の強制範囲です。適用される適用範囲は、しきい値のタイプによって異なります。詳しくは、 144 ページの『しきい値のドメインと適用範囲』を参照してください。
   - オプション: しきい値の作成時にそのしきい値を使用不可にします。デフォルトでは、しきい値は使用可能として作成されます。しきい値を使用不可として作成し、後で使用可能にする場合は、ALTER THRESHOLD ステートメントを使用します。
   - しきい値のタイプおよび許可されている最大値を指定するしきい値述部。最大値に違反すると、しきい値に指定されたアクションが強制的に施行されます。使用可能なしきい値について詳しくは、 153 ページの『接続しきい値』、 154 ページの『アクティビティーしきい値』、 164 ページの『集約しきい値』、および 173 ページの『作業単位しきい値』を参照してください。
   - しきい値の最大値を超えたときに実行するアクション。アクションには、アクティビティーの実行に影響を与える必須アクション (STOP EXECUTION、CONTINUE、FORCE APPLICATION、または REMAP ACTIVITY TO) と、オプションのアクティビティーの収集アクション (COLLECT ACTIVITY DATA) があります。しきい値の境界を違反する原因となったアクティビティーについてどの情報を収集するかは、アクティビティーの収集アクションに指定するオプションによって決まります。
2. 変更をコミットします。 変更をコミットすると、しきい値が SYSCAT.THRESHOLDS ビューに追加されます。

しきい値の変更

ALTER THRESHOLD ステートメントを使用してしきい値を変更します。しきい値を変更して、特定のリソースに課されている限界を変更することもできます。

始める前に

しきい値を変更するためには、SQLADM、WLMADM、または DBADM 権限が必要です。 COLLECT 節以外の節を指定するには、許可 ID に WLMADM または DBADM 権限が組み込まれている必要があります。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

ワーク・アクション・セットのしきい値を変更するには、ALTER WORK ACTION SET ステートメントを ADD WORK ACTION キーワードとともに使用します。

制約事項

ALTER THRESHOLD ステートメントを使用してしきい値タイプを変更することはできません。例えば、MEMBERCONNECTIONS しきい値を TOTALSCMEMBERCONNECTIONS しきい値に変更することはできません。別のしきい値タイプが必要な場合は、既存のしきい値をドロップした後、新規のしきい値を作成します。

手順

しきい値を変更するには、次のようにします。

1. ALTER THRESHOLD ステートメントで、以下に挙げるしきい値のプロパティーを 1 つ以上指定します。変更できるのは以下の各プロパティーです。
   - しきい値述部の境界。
   - しきい値の境界に違反したときに実行するアクション。
   - しきい値が使用可能か使用不可か。
2. 変更をコミットします。 変更をコミットすると、しきい値が SYSCAT.THRESHOLDS ビューで更新されます。

しきい値のドロップ

DDL ステートメント DROP THRESHOLD を使用して、必要でなくなったしきい値をドロップします。

始める前に

しきい値をドロップするためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

ワーク・アクション・セット内のしきい値をドロップする場合は、ALTER WORK ACTION SET ステートメントを使用してください。 DROP ステートメントを使用して WORK ACTION SET 全体をドロップすることによってしきい値をドロップすることもできます。

手順

しきい値をドロップするには、次のようにします。

1. 以下のステップのいずれかを実行します。
   - しきい値がキューイングしきい値である場合は、ALTER THRESHOLD ステートメントを使用して、しきい値を使用不可にします。
   - ALTER THRESHOLD ステートメントを使用してキューイングしきい値を使用不可にしたなら、COMMIT ステートメントを発行して変更をコミットします。
2. DROP THRESHOLD ステートメントを使用して、しきい値をドロップします。
3. 変更をコミットします。 変更をコミットすると、しきい値が SYSCAT.THRESHOLDS ビューから除去されます。

例: しきい値の使用

しきい値をさまざまな目的に使用することができます。このシナリオでは、アプリケーション別に異なる実行時間を許可し、開発中のアプリケーションの動作を制御する目的で、しきい値を使用して大規模なジョブの実行数を制御します。

DB2 ワークロード管理ソリューションを使用することで、企業のさまざまな部署のためにデータベース・リソースを分割し、管理することができます。例えば、営業部門が 2 つのメイン・レポートを管理しており、そこには月間および年間の売り上げが含まれていると想定します。また、人事部門が 1 週間おきに給与計算のアプリケーションを実行し、開発チームは管理チームの要請で新しいタイプのレポートを作成中であると想定します。これらの部門の異なる実行環境を定義するには、次のようにサービス・クラスを作成します。

```
CREATE SERVICE CLASS SALES
CREATE SERVICE CLASS HUMANRESOURCES
CREATE SERVICE CLASS DEVELOPMENT
```

この状態で、これらのアプリケーション 1 つ 1 つに対してワークロード定義を作成し、そのアプリケーションを適用可能なサービス・スーパークラスへマップします。

```
CREATE WORKLOAD MONTHLYSALES APPLNAME('monthlyrpt.exe') SERVICE CLASS SALES
CREATE WORKLOAD YEARLYSALES APPLNAME('yearlyrpt.exe') SERVICE CLASS SALES
CREATE WORKLOAD PAYROLL APPLNAME('payroll.exe') SERVICE CLASS HUMANRESOURCES
CREATE WORKLOAD NEWREPORT APPLNAME('dev.exe') SERVICE CLASS DEVELOPMENT
```

その結果、データベース・カタログには以下のワークロード定義が含まれます。

- MonthlySales (サービス・スーパークラス Sales へマッピングする)
- YearlySales (サービス・スーパークラス Sales へマッピングする)
- Payroll (サービス・スーパークラス Human Resources へマッピングする)
- NewReport (サービス・スーパークラス Development へマッピングする)

大規模なジョブの数のしきい値

YearlySales レポートは非常に大規模なので、データベース内でこのアプリケーションの複数のオカレンスが実行されている、という状況を常に避けたいと思います。そこで、しきい値を作成して、このワークロードの並行オカレンスの最大数を 1 に設定します。

```
CREATE THRESHOLD SINGLEYEARLYSALESRPT FOR WORKLOAD YEARLYSALES ACTIVITIES
   ENFORCEMENT MEMBER
   WHEN CONCURRENTWORKLOADOCCURRENCES > 1
   STOP EXECUTION
```

類似の解決策は、YearlySales アプリケーションをサービス・サブクラス YearlySalesReports (Sales サービス・スーパークラスの下にある) に関連付け、またサービス・サブクラスの最大の並行性しきい値を値 1 に設定することによって実現できます。

```
CREATE SERVICE CLASS YEARLYSALESREPORTS UNDER SALES

ALTER WORKLOAD YEARLYSALES SERVICE CLASS YEARLYSALESREPORTS UNDER SALES

CREATE THRESHOLD SINGLEYEARLYSALESREPORT FOR SERVICE CLASS YEARLYSALESREPORTS
   UNDER SALES ACTIVITIES ENFORCEMENT DATABASE
   WHEN CONCURRENTDBCOORDACTIVITIES > 1
   STOP EXECUTION
```

どちらの状態でも、しきい値のアクションを STOP EXECUTION に設定して、ワークロードの複数のオカレンスが実行されないようにすることができます。しきい値に違反するときの条件についての追加情報を知りたい場合は、アクティビティー情報を収集することもできます。

アクティビティーの存続時間のしきい値

すべてのアプリケーションは 1 時間以内に完了することになっているので、データベース・ドメインを使用してしきい値を作成し、どのアクティビティーも 1 時間を超えて実行されることがないようにします。この規則の唯一の例外は、完了するのに最大 5 時間かかる年間のレポートだけです。したがって、YearlySales ワークロードにはアクティビティー合計時間しきい値として 5 時間を関連付けます。これによって、年間売り上げレポートに適用されるアクティビティー合計時間しきい値がオーバーライドされるので、時間制約が緩和されます。次のように、データベースの他の部分にはグローバル値の 1 時間が適用されますが、YearlySales ワークロードには 5 時間という新しい値が適用されるようになります。

```
CREATE THRESHOLD MAXDBACTIVITYTIME FOR DATABASE ACTIVITIES
   ENFORCEMENT DATABASE
   WHEN ACTIVITYTOTALTIME > 1 HOUR
   STOP EXECUTION

CREATE THRESHOLD MAXYRPTACTIVITYTIME FOR WORKLOAD YEARLYSALES
   ACTIVITIES ENFORCEMENT DATABASE
   WHEN ACTIVITYTOTALTIME > 5 HOURS
   STOP EXECUTION
```

コーディネーターの数およびネストされたアクティビティーのしきい値

NewReport アプリケーションはストアード・プロシージャーおよびユーザー定義関数を頻繁に使用し、まだ完全にデバッグされていないため、システムの残りの部分に影響を与えるアクティビティーを多数生成する傾向があります。開発者に相談すると、この新規のレポートは合計して 20 を超えるアクティビティーを生成しないことになっている、ということが分かります。それで、NewReport ワークロード上にワークロード・アクティビティー・タイプのしきい値を定義して、それを 20 に設定します。最初は、しきい値のアクションを STOP EXECUTION および

COLLECT ALL に設定して、意に反するアプリケーションの副次作用で多数のアクティビティーが開始しないようにするとともに、開発者が問題を識別しやすくなるようにします。

```
CREATE THRESHOLD MAXDEVACTIVITIES FOR SERVICE CLASS DEVELOPMENT ACTIVITIES
   ENFORCEMENT DATABASE
   WHEN CONCURRENTDBCOORDACTIVITIES > 20
   COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
   STOP EXECUTION
```

アプリケーションがより安定したなら、今度は最適化のフェーズに入ります。このフェーズ中に、開発者はアプリケーションが生成するアクティビティーの数を、15 から 20 という数から、15 に削減しようとします。この時、しきい値を変更して、上限の値を 15 に、しきい値のアクションを CONTINUE にします。このしきい値の定義は、生成されたアクティビティーの数が 15 を超えた状態を識別して、それに対応するのに役立ちますが、アプリケーションの安定度が増したため、実行を停止する必要はありません。

```
ALTER THRESHOLD MAXDEVACTIVITIES
   WHEN CONCURRENTDBCOORDACTIVITIES > 15
   COLLECT ACTIVITY DATA ON COORDINATOR WITH DETAILS AND VALUES
   CONTINUE
```

作業単位の実行時間を制限するしきい値

アプリケーション LongUOW は、ときどき実行時間が所定の 10 分を超えることがあるトランザクションを発行します。この結果、ロックが保持される時間が過度に長くなり、より重要なアプリケーションの続行が妨げられてしまいます。このような場合には、アプリケーションが他の作業を中断することは避け、アプリケーションを強制終了します。このアプリケーションのトランザクションの実行時間を、UOWTOTALTIME しきい値を使って管理者が定義した時間に制限することができます。

まず、LongUOW アプリケーションのワークロードを次のように作成します。

```
CREATE WORKLOAD LONG_UOW APPLNAME('LONGUOW') SERVICE CLASS SYSDEFAULTUSERCLASS
```

次に、アプリケーションのトランザクションの完了に 10 分より長くかかった場合に LongUOW アプリケーションを強制終了するこのワークロードのしきい値を次のように作成します。

```
CREATE THRESHOLD FORCELONGUOW FOR WORKLOAD LONG_UOW ACTIVITIES ENFORCEMENT DATABASE
 WHEN UOWTOTALTIME > 10 MINUTES FORCE APPLICATION
```

サービス・サブクラス・レベルまたはデータベース・レベルでこのしきい値を適用することもできます。

接続しきい値

接続しきい値は、個々のデータベース接続に制御を適用します。接続しきい値を使用して、データベースへの同時接続の総数および接続の最大アイドル時間を制限できます。

CONNECTIONIDLETIME しきい値

CONNECTIONIDLETIME しきい値は、接続をアイドル (つまり、ユーザー要求を処理していない状態) にしておくことのできる時間の長さの上限を指定します。

タイプ 接続

定義ドメイン
データベースまたはサービス・スーパークラス

適用範囲
データベース

トラッキングされる作業
ユーザー接続

キューイング
不可

単位 分、時間、または日数で表現される時間の長さ

予測的か反応的か
反応的

しきい値で指定された時間よりも長く接続がアイドル状態のままになっており、しきい値アクションが STOP EXECUTION である場合、接続は閉じられます。

アクティビティーしきい値

アクティビティーしきい値は、個別のアクティビティーに適用されます。個別のアクティビティーのリソースの使用量が、それをトラッキングしているしきい値の上限を超えると、対応するアクションが起動され、そのアクティビティーに 1 回適用されます。

一度適用されると、しきい値はそのアクティビティーに対して非活動状態になり、再び適用されることはありません。

例えば、経過時間 5 分で CONTINUE アクションを起動する時間ベースのしきい値を定義したとします。アクティビティーがこのしきい値に違反した場合、アクションは 1 回適用されるだけで、5 分おきに再適用されることはありません。

アクティビティーしきい値のドメイン優先順位

アクティビティーしきい値は個々のアクティビティーに適用されます。実行中の同一アクティビティーに複数のしきい値が適用される場合は、どのしきい値を実施するかについて決定する必要があります。

集約しきい値は影響を受けません。例えば並行性しきい値と同じように、同一アクティビティーが複数のアクティビティー集約に同時に寄与できるためです。

実行中のアクティビティーに適用するアクティビティーしきい値についての解決は、ローカル側のドメイン内で定義された値は、より広域またはよりグローバルなドメインからのあらゆる値をオーバーライドする、という規則に従います。ドメインの階層は次のとおりです。最もローカルなものから最もグローバルなものへの順に示しています。

1. ステートメント
2. ワーク・アクション (ワークロード・レベル)
3. ワークロード
4. サービス・サブクラス

5. サービス・スーパークラス
6. ワーク・アクション (データベース・レベル)
7. データベース

以下の例は、しきい値のオーバーライド方法を示しています。

例

以下の例は、しきい値のオーバーライド方法を示しています。

- データベース・ドメインで定義されたすべてのデータベース照会の最大実行時間として 1 時間を定義するしきい値は、大きな照会を扱うようセットアップされたサービス・スーパークラスの最大実行時間として 5 時間を定義するしきい値によってオーバーライドされます。
- その同じサービス・スーパークラスしきい値は、サービス・サブクラスの最大実行時間として 10 時間を定義する、非常に大きい照会に関するしきい値によってオーバーライドされます。
- データベース・ドメインで定義された最大実行時間 1 時間は、より短い重要な照会を迅速に完了できるようにする別のサービス・スーパークラスでの 10 分という値によってオーバーライド可能です。
- ステートメント・ドメインでしきい値として指定されたテキストに一致するテキストを含んだステートメントを実行すると、しきい値違反が発生し、他のすべてのしきい値がオーバーライドされます。

ACTIVITYTOTALTIME しきい値

ACTIVITYTOTALTIME しきい値は、データ・サーバーがアクティビティーの処理に費やすことができる時間の上限を指定します。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション、ワークロード、およびステートメント

適用範囲
データベース

トラッキングされる作業
認識されているコーディネーター・アクティビティーおよびネストされたアクティビティー (17 ページの『アクティビティー』を参照)

キューイング
不可

単位 秒、分、時間、または日で表される時刻期間。指定する時間単位が秒の場合、その値は 10 の倍数でなければなりません。

予測的か反応的か
反応的

この時間は、ワークロード管理キューでの待機時間すべて、およびアクティビティーの実行中に生じたその他すべての待機時間 (ロック待機時間など) を含めたアクティビティーの全存続時間を表します。カーソルが開かれている場合は、そのカーソ

ルに関連付けられているアクティビティーはカーソルが閉じられるまで続きます。このしきい値が適用されるアクティビティーには、コンパイル時間を除く SQL ステートメントの実行、およびロード・ユーティリティーの実行が含まれます。

時間のしきい値がストアード・プロシージャーに適用される際は、そのストアード・プロシージャーの内部で行われている作業にもしきい値が適用されます。したがって、ストアード・プロシージャーの時間のしきい値が限度に達したときは、そのストアード・プロシージャーの内部で行われているすべての作業が停止します。最も深いネスト・レベルのアクティビティー実行に適用される時間しきい値の階層は、ストアード・プロシージャーの呼び出しの階層から導き出すことができます。常に、その階層内で最も制限の高い時間しきい値 (つまり、期限が最も近い時間しきい値) が適用されます。

データ・サーバーは、IMPORT、EXPORT、およびその他の各 CLP コマンドをユーザー・ロジックとみなします。 IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

CPUTIME しきい値

CPUTIME しきい値は、アクティビティーの実行中にそのアクティビティーが特定のメンバーで使用できるユーザーおよびシステム・プロセッサー時間の合計の上限を指定します。このしきい値を使用して、プロセッサー・リソースを過剰に使用しているアクティビティーを検出し、制御します。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション、ワークロード、およびステートメント

適用範囲
メンバー

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 時間

予測的か反応的か
反応的

アクティビティーの実行に費やされるプロセッサー時間の長さは、しきい値によるキューイングの後、アクティビティーがメンバーで実行を開始した時点から、アクティビティーが実行を終了した時点までの時間から測定されます。

このしきい値では、以下のアクティビティーがトラッキングされます。

- すべての DML アクティビティー。
- CALL アクティビティー。子アクティビティーのプロセッサー時間は CALL アクティビティーのプロセッサー時間に含まれません。 fenced プロセスに費やされるプロセッサー時間も、CALL アクティビティーの合計プロセッサー時間に含まれません。

ユーティリティーまたはプロシージャーを使用してデータベース・マネージャーによって開始されるアクティビティーは、この条件に含まれません (ADMIN_CMD プロシージャーは例外)。データ・サーバーは、IMPORT、EXPORT、およびその他の各 CLP コマンドをユーザー・ロジックとみなします。 IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。 LOAD コマンドの子アクティビティーは、このしきい値によってトラッキングされません。

例

以下の例は、メンバーの適用範囲を持つデータベース・ドメインの CPUTIME しきい値 TH1 を作成します。このしきい値は、アクティビティーの実行時間が 30 秒を超えた場合にそのアクティビティーを停止します。これは 5 秒間隔で検査されます。このしきい値を使用して、システム上の照会がプロセッサー時間を必要以上に費やさないようにすることができます。これが制御されないと、システム上で実行されている他の作業に悪影響が及ぶ可能性があります。

```
CREATE THRESHOLD TH1 FOR DATABASE ACTIVITIES
  ENFORCEMENT MEMBER
  WHEN CPUTIME > 30 SECONDS CHECKING EVERY 5 SECONDS
  STOP EXECUTION;
```

CPUTIMEINSC しきい値

サービス・クラス内 CPUTIMEINSC しきい値は、特定のサービス・サブクラスでの実行中にアクティビティーが特定のメンバーで使用できるユーザーおよびシステム・プロセッサー時間の合計の上限を指定します。このしきい値を使用して、プロセッサー・リソースを過剰に使用しているアクティビティーを検出し、制御します。

クラス アクティビティー

定義ドメイン
サービス・サブクラス

適用範囲
メンバー

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 時間

予測的か反応的か
反応的

アクティビティーの実行に費やされるプロセッサー時間は、アクティビティーが現行サービス・サブクラスに入った時点から、アクティビティーがそのサービス・サブクラスから出るかまたは実行を終了した時点までの時間から測定されます。

このしきい値は、アクティビティーの存続時間中に使用される合計プロセッサー時間ではなく、特定のサービス・サブクラスで使用されるプロセッサー時間のみを制御するという点で CPUTIME しきい値と異なります。

このしきい値では、以下のアクティビティーがトラッキングされます。

- すべての DML アクティビティー。
- CALL アクティビティー。子アクティビティーのプロセッサー時間は CALL アクティビティーのプロセッサー時間に含まれません。 fenced プロセスに費やされるプロセッサー時間も、CALL アクティビティーの合計プロセッサー時間に含まれません。

ユーティリティーまたはプロシージャーを使用してデータベース・マネージャーによって開始されるアクティビティーは、この条件に含まれません (ADMIN_CMD プロシージャーは例外)。データ・サーバーは、IMPORT、EXPORT、およびその他の各 CLP コマンドをユーザー・ロジックとみなします。 IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。 LOAD コマンドの子アクティビティーは、このしきい値によってトラッキングされません。

REMAP ACTIVITY アクションを使用して、異なるリソース割り当てによってアクティビティーをサービス・サブクラスに再マップすることでアクティビティーを制御することができます。

例

以下の例は、スーパークラス A の下、A1、A2 という 2 つのサービス・サブクラスを作成します。これとともに単一のサービス・クラス内 CPUTIMEINSC しきい値が作成されます。このしきい値は、照会の評価の際に、サービス・サブクラス A1 でプロセッサー時間が 1 分間使用された後、別のサブクラスにアクティビティーを再マップします。イベント・モニターのレコードに記録されます。

```
CREATE SERVICE CLASS A;
CREATE SERVICE CLASS A1 UNDER A;
CREATE SERVICE CLASS A2 UNDER A;

CREATE THRESHOLD T1 FOR SERVICE CLASS A1 UNDER A
  ACTIVITIES ENFORCEMENT MEMBER
  WHEN CPUTIMEINSC > 1 MINUTE CHECKING EVERY 30 SECONDS
  REMAP ACTIVITY TO A2 LOG EVENT MONITOR RECORD;
```

DATATAGINSC しきい値

サービス・クラス内しきい値 DATATAGINSC は、アクティビティーによってアクセスされる、表スペースまたはストレージ・グループのデータ・タグの値を確認します。このしきい値を使用して、どのデータがアクセスされているかに基づき、アクティビティーを別の DB2 サービス・サブクラスに動的にマップします。

タイプ アクティビティー

定義ドメイン
サービス・サブクラス

適用範囲
メンバー

トラッキングされる作業
認識されるコーディネーター

キューイング
不可

単位　Data タグ

予測的か反応的か
反応的

表スペースやストレージ・グループを作成または変更しているときに、DATA TAG 属性を指定することによって、データを数値でタグ付けできます。スキャンが最初に表で開かれたとき、および挿入が表に対して実行されたときに、データ・タグしきい値が評価されます。再マップ操作の結果としてスキャンが開かれた後で、アクティビティーによって選出された新しいデータ・タグしきい値は、そのスキャンに適用されません。

照会でアクセスされているデータ表スペースに対するデータ・タグのみが、しきい値によって考慮されます。索引のデータ・タグの値、または LONG 表スペースは、しきい値の評価中に考慮されません。例えば、照会 `SELECT COUNT(*) FROM T1` が、表スペース内にデータ・タグ 3 で配置された索引および表スペースにデータ・タグ 1 で配置された表データへの索引専用アクセスである場合、照会が実行されるときのしきい値の評価はデータ・タグ 1 (データ表スペースに対するデータ・タグ) を使用します。データ・タグしきい値の動作は、照会のアクセス・プランで選択されたアクセス方式 (索引または直接スキャン) から独立しています。

このしきい値で追跡されるアクティビティーは、以下のとおりです。

- DML タイプのコーディネーター・アクティビティーと、それに対応するサブエージェント作業 (サブセクション実行)。
- ユーザー・アプリケーションから派生するネストされた DML アクティビティー。結果として、ユーティリティー、SYSPROC プロシージャー (SYSPROC.ADMIN_CMD を除く)、または内部 SQL ステートメントなどのデータベース・マネージャーによって発行される DML アクティビティーはこのしきい値の影響を受けません。

以下の使用法のシナリオでは、ユーザーは 3 つの表スペースを作成します。TBHIGH には、優先順位の高いデータが含まれ、データ・タグ 1 を持ち、TBMED には、中間の優先順位のデータが含まれ、データ・タグ 4 を持ち、TBLOW には、優先順位の低いデータが含まれ、データ・タグ 9 を持ちます。アクティビティーは、最初に優先順位の高いサービス・サブクラスにマップされます。アクティビティーが TBHIGH 表スペース以外の表スペースのデータをタッチする場合、ユーザーはそれらのアクティビティーの優先順位を落とそうとします。以下の例では、3 つのサービス・サブクラス SCHIGH、SCMED、および SCLOW (それぞれ優先順位が高い作業、中間の作業、低い作業の実行のため) を含むサービス・スーパークラス MAINSC が既に作成されています。

優先順位の高いサービス・サブクラスで DATATAGINSC しきい値を作成し、データ・タグが 1 以外のデータにアクティビティーがタッチする場合は、中間の優先順位のサービス・サブクラスにマップします。

```
CREATE THRESHOLD MAPTOMED FOR SERVICE CLASS SCHIGH UNDER MAINSC ACTIVITIES
  ENFORCEMENT MEMBER WHEN DATATAGINSC NOT IN (1) REMAP ACTIVITY TO SCMED
```

中間の優先順位のサービス・サブクラスで SQLDATATAGINSC しきい値を作成し、データ・タグが 9 のデータにアクティビティーがタッチする場合は、優先順位が低いサービス・サブクラスにマップします。

```
CREATE THRESHOLD MAPTOLOW FOR SERVICE CLASS SCMED UNDER MAINSC ACTIVITIES
  ENFORCEMENT MEMBER WHEN DATATAGINSC IN (9) REMAP ACTIVITY TO SCLOW
```

データ・タグが 9 の表スペースにアクティビティーがタッチする場合、そのアクティビティーは SCMED に再マップされ、その後すぐに SCLOW にもう一度再マップされます (SCMED サービス・サブクラスのしきい値のため)。この場合、ユーザーには、2 つのしきい値の違反が表示されます。

ESTIMATEDSQLCOST しきい値

ESTIMATEDSQLCOST しきい値は、DML アクティビティーで許可される見積コストの最大値を指定します。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション、ワークロード、およびステートメント

適用範囲
データベース

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 timeron で表現される見積 SQL コスト

予測的か反応的か
予測的

このしきい値は、以下のアクティビティーを追跡します。

- コーディネーター・メンバーで発行される DML アクティビティー。
- ユーザー・アプリケーションから呼び出されるネストされた DML アクティビティー。したがって、DB2 ユーティリティー、SYSPROC ストアード・プロシージャー、および内部 SQL の中から発行される DML アクティビティーなど、データ・サーバーによって内部的に発行される DML アクティビティーは、このしきい値の影響を受けません。ただし、アクティビティーのコストが親アクティビティーの見積もりに含まれている場合は例外です。この場合は、アクティビティーは間接的にトラッキングされます。間接的にトラッキングされるアクティビティーの例としては、トリガーが挙げられます。IMPORT、EXPORT、およびその他の各 CLP コマンドは、ユーザー・ロジックと見なされます。
  IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。 DML 作業タイプをもつワーク・クラスに分類されるアクティビティーについては、56 ページの『タイプ、コスト、またはワーク・クラスでアクセスされるデータによる作業の識別』を参照してください。

SQLROWSREAD しきい値

SQLROWSREAD しきい値は、DML アクティビティーがメンバーで読み取ることのできる行の最大数を指定します。このしきい値を使用して、過剰な数の行を読み取っているアクティビティーを検出し、制御します。

クラス アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション、ワークロード、およびステートメント

適用範囲
メンバー

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 行の数

予測的か反応的か
反応的

このしきい値は、データ・サーバーからクライアント・アプリケーションに戻される行の数ではなく、照会を評価する際に読み取られる行の最大数を制御するという点で SQLROWSRETURNED しきい値と異なります。

索引アクセスは、合計読み取り行数に含まれません。アクセス・プランが照会の評価の際に索引だけを使用する場合、SQLROWSREAD しきい値の違反は発生しません。

このしきい値は、ユーザーが構成できる時間間隔で評価されます。読み取り行数を超えるまでにかかる時間よりその間隔が長いと、違反が検出される前にメンバーのアクティビティーの読み取り行数がしきい値の境界を超える可能性があります。

このしきい値では、以下のアクティビティーがトラッキングされます。

- DML タイプのコーディネーター・アクティビティーとそれに対応するサブエージェントの作業 (サブセクション実行など)。
- ユーザー・アプリケーションから派生するネストされた DML アクティビティー。
  DB2 ロジック (ユーティリティー、SYSPROC プロシージャー、または内部 SQL ステートメント) によって発行される DML アクティビティーはこのしきい値の影響を受けません。 IMPORT、EXPORT、およびその他の各 CLP コマンドは、ユーザー・ロジックと見なされます。そのため、IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

例

以下の例は、メンバーの適用範囲を持つデータベース・ドメインの SQLROWSREAD しきい値 TH1 を作成します。このしきい値は、照会の評価の際に読み取られる行数が 5 000 000 を超える場合にそのアクティビティーの実行を停止します。しきい値はこれを 10 秒間隔で検査します。このしきい値を使用して、システム上の照会が必要以上の行数を読み取らないようにすることができます。これが制御されないと、システム上で実行されている他の作業に悪影響が及ぶ可能性があります。

```
CREATE THRESHOLD TH1 FOR DATABASE ACTIVITIES
  ENFORCEMENT MEMBER
  WHEN SQLROWSREAD > 5000000 CHECKING EVERY 10 SECONDS
  STOP EXECUTION;
```

SQLROWSREADINSC しきい値

サービス・クラス内 SQLROWSREADINSC しきい値は、特定のサービス・サブクラスでの実行中に DML アクティビティーが特定のメンバーで読み取ることのできる行の最大数を指定します。このしきい値を使用して、過剰な数の行を読み取っているアクティビティーを検出し、制御します。

クラス アクティビティー

定義ドメイン
サービス・サブクラス

適用範囲
メンバー

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 行の数

予測的か反応的か
反応的

このしきい値は、アクティビティーの存続時間中の合計読み取り行数ではなく、アクティビティーが特定のサービス・サブクラスに入った時点以降の読み取り行数のみを制御するという点で SQLROWSREAD しきい値と異なります。また、このしきい値は、データ・サーバーからクライアント・アプリケーションに戻される行の数ではなく、現行サービス・サブクラスの照会の評価の際に読み取られる行の最大数を制御するという点でも SQLROWSRETURNED しきい値と異なります。

索引アクセスは、合計読み取り行数に含まれません。アクセス・プランが照会の評価の際に索引だけを使用する場合、SQLROWSREADINSC しきい値の違反は発生しません。

このしきい値は、ユーザーが構成できる時間間隔で評価されます。読み取り行数を超えるまでにかかる時間よりその間隔が長いと、違反が検出される前にメンバーのアクティビティーの読み取り行数がしきい値の境界を超える可能性があります。

このしきい値では、以下のアクティビティーがトラッキングされます。

- DML タイプのコーディネーター・アクティビティーとそれに対応するサブエージェントの作業 (サブセクション実行など)。
- ユーザー・アプリケーションから派生するネストされた DML アクティビティー。
  DB2 ロジック (ユーティリティー、SYSPROC プロシージャー、または内部 SQL ステートメント) によって発行される DML アクティビティーはこのしきい値の影響を受けません。 IMPORT、EXPORT、およびその他の各 CLP コマンドは、

ユーザー・ロジックと見なされます。そのため、IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

REMAP ACTIVITY アクションを使用して、異なるリソース割り当てによってアクティビティーをサービス・サブクラスに再マップすることでアクティビティーを制御することができます。

例

以下の例は、スーパークラス A の下、A1、A2 という 2 つのサービス・サブクラスを作成します。これとともに単一のサービス・クラス内 SQLROWSREADINSC しきい値が作成されます。このしきい値は、照会の評価の際に、サービス・サブクラス A1 で 10 000 行が読み取られた後、別のサブクラスにアクティビティーを再マップします。イベント・モニターのレコードに記録されます。

```
CREATE SERVICE CLASS A;
CREATE SERVICE CLASS A1 UNDER A;
CREATE SERVICE CLASS A2 UNDER A;

CREATE THRESHOLD T1 FOR SERVICE CLASS A1 UNDER A
  ACTIVITIES ENFORCEMENT MEMBER
  WHEN SQLROWSREADINSC > 10000 REMAP ACTIVITY TO A2
  LOG EVENT MONITOR RECORD;
```

SQLROWSRETURNED しきい値

SQLROWSRETURNED しきい値は、データ・サーバーがクライアントに戻すことができる行の最大数を指定します。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション、ワークロード、およびステートメント

適用範囲
データベース

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 行の数

予測的か反応的か
反応的

CALL ステートメントから複数の結果セットが戻される場合、しきい値は、全結果セットから戻される行の総数の集約に対してではなく、結果セットごとに別個に適用されます。例えば、しきい値を 20 行として定義している場合に、CALL ステートメントからそれぞれ 15 行と 19 行を戻す 2 つの結果セットが戻されたとしても、しきい値はトリガーされません。

このしきい値では、以下のアクティビティーがトラッキングされます。

- コーディネーター・メンバーで発行される DML アクティビティー。

- ユーザー・アプリケーションから呼び出されるネストされた DML アクティビティー。したがって、DB2 ユーティリティー、SYSPROC ストアード・プロシージャー、および内部 SQL の中から発行される DML アクティビティーなど、データ・サーバーによって内部的に発行される DML アクティビティーは、このしきい値の影響を受けません。

SQLTEMPSPACE しきい値

SQLTEMPSPACE しきい値は、あらゆるメンバーにおいて DML アクティビティーが消費できる SYSTEM TEMPORARY 表スペースの最大量を指定します。 DML アクティビティーでは、しばしば、ソートや中間結果セットの処理などの操作に TEMPORARY 表スペースが使用されます。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション、ワークロード、およびステートメント

適用範囲
メンバー

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 キロバイト (KB)、メガバイト (MB)、またはギガバイト (GB) で表現される TEMPORARY 表スペースの量

予測的か反応的か
反応的

このしきい値では、以下のアクティビティーがトラッキングされます。

- コーディネーター・メンバーで発行される DML アクティビティー。
- ユーザー・アプリケーションから派生するネストされた DML アクティビティー。
  DB2 ロジック (ユーティリティー、SYSPROC プロシージャー、または内部 SQL) によって発行される DML アクティビティーはこのしきい値の影響を受けません。

データ・サーバーは、IMPORT、EXPORT、およびその他の各 CLP コマンドをユーザー・ロジックとみなします。 IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

集約しきい値

集約しきい値は、データベース内の作業の複数のエレメントに対して、集合的な制御を行います。集約しきい値を使用して定義する境界は、合計値として機能し、しきい値によってトラッキングされるあらゆる作業の合計値となります。

新しくインスタンス化された作業が原因で上限に違反した場合、対応するアクションが起動します。上限を違反する原因となった作業だけが、トリガー・アクションの影響を受けます。

アクティビティーのキューイング

しきい値によってはキューが組み込まれています。これらのしきい値では、並行性限度に達した後、キューの限度設定を超えるまで、すべての追加アクティビティーをキューに入れることによって、並行実行できるアクティビティーの数を強制することができます。

アクティビティーの数がキューイングしきい値の並行性限度を超えると、新しい要求は、先入れ先出し法で自動的にキューに入れられます。これはキューがしきい値定義のキューイング境界によって指定されたサイズに達するまで続きます。キューがいっぱいになると、上限に達します。次の要求はしきい値に違反したと見なされ、しきい値に指定されているアクションがこれに適用されることになります。例えば、STOP EXECUTION のアクションでは、新しく到着する作業が拒否されます。

キューイング上限を無制限と定義することも可能です。この場合、キューのサイズに上限はありません。この場合、新しく到着する作業がキューに追加された場合、キューがどれほど大きくなったかにかかわらず、しきい値に違反したとは見なされません。キューの上限にハード・リミットを定義し、CONTINUE をしきい値違反のアクションとして定義すると、ハード・リミットを超えて新しく到着する作業はすべてしきい値境界に違反したと見なされ、しきい値違反は作成されますが、新しい作業は引き続きキューに追加されます。

AGGSQLTEMPSPACE しきい値

AGGSQLTEMPSPACE しきい値は、サービス・サブクラス内で同時に実行されている DML アクティビティー全体で使用できる SYSTEM TEMPORARY 表スペースの最大量を指定します。 DML アクティビティーでは、しばしば、ソートや中間結果セットの処理などの操作に TEMPORARY 表スペースが使用されます。

クラス 集約

定義ドメイン
サービス・サブクラス

適用範囲
メンバー

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 キロバイト、メガバイト、またはギガバイト

予測的か反応的か
反応的

このしきい値では、以下のアクティビティーがトラッキングされます。

- コーディネーター・メンバーで発行される DML アクティビティー。
- ユーザー・アプリケーションから派生するネストされた DML アクティビティー。
  DB2 ロジック (ユーティリティー、SYSPROC プロシージャー、または内部 SQL ステートメント) によって発行される DML アクティビティーはこのしきい値の

影響を受けません。 IMPORT、EXPORT、およびその他の各 CLP コマンドは、ユーザー・ロジックと見なされます。そのため、IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

CONCURRENTDBCOORDACTIVITIES しきい値

CONCURRENTDBCOORDACTIVITIES しきい値は、指定された定義ドメインおよび適用範囲内で同時に実行できる、認識されているコーディネーター・アクティビティーの最大数を指定します。

このタイプのしきい値の使用は、一度に複数のアクティビティーを実行しないアプリケーションに最適です。アプリケーションが複数のアクティビティーを同時に開始する場合 (例えば、カーソルが開いている間に UPDATE SQL ステートメントを発行するなど)、しきい値によって許容される並行性のレベルや、関係する他のアプリケーションの動作によっては、特定のキュー競合シナリオが発生することがときどきあります。アプリケーションが複数のアクティビティーを同時に実行できる、またはアプリケーションの動作が不明なシナリオにこのしきい値が存在する場合は、それらのアクティビティーに ACTIVITYTOTALTIME しきい値を定義することをお勧めします。これは、潜在的なキュー競合シナリオを自動的に解決する上で役立ちます。

タイプ 集約

定義ドメイン
データベース、ワーク・アクション、サービス・スーパークラス、サービス・サブクラス

適用範囲
DB2 pureScale 環境以外の環境のデータベース

DB2 pureScale 環境のメンバー

トラッキングされる作業
認識されているコーディネーター・アクティビティーおよびネストされたアクティビティー (56 ページの『タイプ、コスト、またはワーク・クラスでアクセスされるデータによる作業の識別』を参照)

キューイング
可

単位 並行データベース・アクティビティーの数

予測的か反応的か
予測的

このしきい値は、CONCURRENTWORKLOADACTIVITIES しきい値の汎用化です。CONCURRENTWORKLOADACTIVITIES はワークロード・ドメインで実行されるアクティビティーにのみ適用されますが、CONCURRENTDBCOORDACTIVITIES しきい値は、データベース全体から単一のワーク・アクションに及ぶさまざまなドメインに適用することができます。CONCURRENTDBCOORDACTIVITIES しきい値は、非 CALL ステートメント用のコーディネーター・アクティビティーと、CALL ステートメントによって生成されるすべてのネストされたアクティビティーをトラッキングします。 CONCURRENTWORKLOADACTIVITIES しきい値と異なり、CONCURRENTDBCOORDACTIVITIES しきい値はキューイングしきい値です。

注: キューイングしきい値に対して CONTINUE しきい値アクションを指定した場合は、キューのサイズとして指定されている強制値に関係なくキューのサイズが事実上無制限になります。**concurrent_act_top** モニター・エレメントを使用して、収集された時間間隔にサービス・サブクラス用のメンバーで到達したアクティビティー（ネストされたアクティビティーを含む）の並行性の最大数を判別することができます。

CONCURRENTDBCOORDACTIVITIES タイプのキューイングしきい値を作成する場合は、キューによる解決できない競合が発生しかねない構成に注意してください。以下に例を示します。

1. タイプ CONCURRENTDBCOORDACTIVITIES の並行性しきい値が作成されます。最大並行値は 1、キュー・サイズは 2 以上です。
2. アプリケーションは、DB2 データ・サーバーがアクティビティー A1 と認識するカーソルを開きます。これによって、しきい値に使用できるユニークなチケットが消費されます。
3. カーソルがオープンである間に、アプリケーションは UPDATE ステートメントを発行します。これをデータ・サーバーはアクティビティー A2 と認識します。このアクティビティーも、並行性しきい値の影響を受けます。A1 アクティビティーが既に実行中であるため、新規のアクティビティー A2 はキューに入れられます。

   アプリケーションは、解決できないキュー競合の状態になっています。アプリケーションは A2 が実行するまで待機しますが、A2 は A1 の実行が終了するまで待機しています。この状況は、ACTIVITYTOTALTIME しきい値もそれらのアクティビティーに対して定義されていない限り、外部からの介入なしには解決しません。 ACTIVITYTOTALTIME しきい値が定義されている場合、データ・サーバーがアクティビティー A1 の処理に費やす時間の上限を超えることによって A1 がそのしきい値に違反したときに、この状況は解決されます。

この例は、複数のアプリケーションとキューの場合にも一般化して考えられます。この状況は、並行値を大きくするか、または並行値が適正に設定されている場合は一部のアクティビティーをキャンセルすることによって解決します。

解決できないキュー競合のシナリオを作成してしまう可能性を減らすために、CONCURRENTDBCOORDACTIVITIES しきい値は以下のようにさまざまなタイプのアクティビティーに影響を与えます。

- CALL ステートメントはしきい値の管理を受けませんが、ネストされたすべての子アクティビティーはしきい値の管理下にあります。無名ブロックと自律型ルーチンは、CALL ステートメントに分類されることに注意してください。
- ユーザー定義関数 (UDF) はしきい値の制御を受けますが、UDF 内にネストされた子アクティビティーは制御されません。自律型ルーチンがユーザー定義関数から呼び出される場合、自律型ルーチンと、その自律型ルーチンの子アクティビティーはどちらもしきい値の制御下に置かれません。
- CALL ステートメントを呼び出すトリガー・アクション、およびそれらの CALL ステートメントの子アクティビティーは、しきい値の管理下にありません。トリガーをアクティブ化する可能性がある INSERT、UPDATE、および DELETE ステートメント自体はしきい値の制御を受けることに注意してください。

CONCURRENTWORKLOADACTIVITIES しきい値

CONCURRENTWORKLOADACTIVITIES しきい値は、1 つのワークロード・オカレンス内で同時に実行できるコーディネーター・アクティビティーおよびネストされたアクティビティーの最大数を指定します。

タイプ 集約

定義ドメイン
ワークロード

適用範囲
ワークロード・オカレンス

トラッキングされる作業
認識されているコーディネーター・アクティビティーおよびネストされたアクティビティー (17 ページの『アクティビティー』を参照)

キューイング
不可

単位 並行ワークロード・アクティビティーの数

予測的か反応的か
予測的

このしきい値は、単一のワークロード・オカレンスに適用されます。同時に実行されるワークロードのオカレンスが複数存在する場合、しきい値は、各ワークロード・オカレンスに別個に適用されます。トラッキングされるアクティビティーには、認識されているコーディネーター・アクティビティーすべてと、コーディネーター・アクティビティーの実行の結果として生成されたすべてのネストされたアクティビティーが含まれます。例えば、CONCURRENTDBCOORDACTIVITIES しきい値とは異なり、あるストアード・プロシージャーが呼び出されてそのストアード・プロシージャーが何らかの SQL を実行した場合は、CALL ステートメント (コーディネーター・アクティビティー) と、ストアード・プロシージャーが実行した SQL ステートメント (ネストされたアクティビティー) の両方が、しきい値の合計のカウントに含められます。

COMMIT、ROLLBACK、および ROLLBACK to SAVEPOINT ステートメントは、このしきい値の影響を受けません。

ネストされたアクティビティーに関する考慮事項

このしきい値によってトラッキングされるネストされたアクティビティーは、以下の基準を満たしている必要があります。

- 認識されているコーディネーター・アクティビティーであること。 56 ページの『タイプ、コスト、またはワーク・クラスでアクセスされるデータによる作業の識別』に記述されている認識済みのタイプのものではない、ネストされたコーディネーター・アクティビティーは、カウントされません。
- SQL を発行するユーザー作成のストアード・プロシージャーなどのユーザー・ロジックや、SYSPROC.ADMIN_CMD ストアード・プロシージャーから直接呼び出されること。DB2 ユーティリティーの呼び出しや、SYSIBM、SYSFUN、または

SYSPROC スキーマ内の他の何らかのコードの呼び出しによって開始された、ネストされたコーディネーター・アクティビティーは、このしきい値で指定された上限へのカウントに含まれません。

例

この例では、CONCURRENTWORKLOADACTIVITIES しきい値の最大値が 5 に設定されており、ユーザー・ロジックによってワークロード・オカレンスで以下の一連の操作が行われます。

1. **load** コマンドを発行する: 現在のワークロード・アクティビティーの数は 1 です。
   - **load** コマンドが内部で何らかの SQL を発行する: 現在のワークロード・アクティビティーの数は 1 です (ユーティリティーによって生成された SQL は、CONCURRENTWORKLOADACTIVITIES しきい値のカウントに含まれません)。
   - **load** コマンドが終了する: 現在のワークロード・アクティビティーの数は 0 です。
2. SYSPROC.SP1 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 1 です。
   - SYSPROC.SP1 ストアード・プロシージャーが何らかの SQL を生成する: 現在のワークロード・アクティビティーの数は 1 です (ユーティリティーによって生成された SQL は、CONCURRENTWORKLOADACTIVITIES しきい値のカウントに含まれません)。
   - SYSPROC.SP1 ストアード・プロシージャーが終了する: 現在のワークロード・アクティビティーの数は 0 です。
3. カーソル C1 を開く: 現在のワークロード・アクティビティーの数は 1 です。
4. **runstats** コマンドを発行する: 現在のワークロード・アクティビティーの数は 1 です。
   - **runstats** コマンドが何らかの SQL を生成する: 現在のワークロード・アクティビティーの数は 1 です。
   - **runstats** コマンドが終了する: 現在のワークロード・アクティビティーの数は 1 です。
5. カーソル C1 を閉じる: 現在のワークロード・アクティビティーの数は 0 です。
6. BOB.SP1 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 1 です。
   - BOB.SP1 ストアード・プロシージャーが 3 つのカーソルを開く: 現在のワークロード・アクティビティーの数は 4 です。
   - BOB.SP1 ストアード・プロシージャーが SYSPROC.SP2 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 5 です。
     - SYSPROC.SP2 ストアード・プロシージャーが何らかの SQL を発行する: 現在のワークロード・アクティビティーの数は 5 です。
     - SYSPROC.SP2 ストアード・プロシージャーが終了する: 現在のワークロード・アクティビティーの数は 4 です。

- BOB.SP1 ストアード・プロシージャーが BOB.SP2 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 5 です。
  - BOB.SP2 ストアード・プロシージャーが何らかの SQL を発行する: この時点で、しきい値がトリガーされます。
  - BOB.SP2 ストアード・プロシージャーが終了する: 現在のワークロード・アクティビティーの数は 4 です。
- BOB.SP1 ストアード・プロシージャーが終了する: 現在のワークロード・アクティビティーの数は 0 です。

7. カーソル C2 を開く: 現在のワークロード・アクティビティーの数は 1 です。
8. BOB.SP2 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 2 です。

CONCURRENTWORKLOADOCCURRENCES しきい値

CONCURRENTWORKLOADOCCURRENCES しきい値は、コーディネーター・メンバー上で同時に実行できるワークロード・オカレンスの最大数を指定する集約しきい値です。

タイプ 集約

定義ドメイン
ワークロード

適用範囲
メンバー

トラッキングされる作業
ワークロード・オカレンス

キューイング
不可

単位 並行ワークロード・オカレンスの数

予測的か反応的か
予測的

ワークロード・オカレンスの開始時に、それが生成する作業が非コーディネーター・メンバーに送信された場合、それらのメンバー上の作業は、コーディネーター・メンバー上の並行性しきい値の合計に対してカウントされません。例えば、CONCURRENTWORKLOADOCCURRENCES しきい値が、あるメンバー上のワークロード A の 1 つのオカレンスのみを許可するように定義されていると仮定します。さらに、あるアプリケーションがメンバー 1 に接続し、その結果ワークロード A のオカレンスが開始され、このワークロードが原因で作業がデータベース・メンバー 1、2、および 3 に送信されると仮定します。この状況では、ワークロード A のオカレンスの合計数は、メンバー 1 では 1、データベース・メンバー 2 および 3 では 0 です。そのため、別のアプリケーションがメンバー 1 に接続し、メンバー 1 でワークロード A の別のオカレンスが開始されると、そのワークロードは拒否されます。ただしワークロード A の新規オカレンスについては、データベース・メンバー 2 および 3 で開始できます。

TOTALMEMBERCONNECTIONS しきい値

TOTALMEMBERCONNECTIONS しきい値は、データベースのコーディネーター・メンバーでの並行データベース接続の最大数を指定します。つまり、このしきい値は、各データベース・メンバーでデータベースに接続できるクライアントの最大数を制御します。

このしきい値は、DBADM 権限および WLMADM 権限を持つユーザーには強制されません。

タイプ 集約

定義ドメイン
データベース

適用範囲
メンバー

トラッキングされる作業
接続

キューイング
可 (0 で強制)

単位 同時接続の数

予測的か反応的か
予測的

例えば、TOTALMEMBERCONNECTIONS しきい値を 10 に設定していて、データベースに 5 つのメンバーがある場合、各メンバーでは 10 (データベース全体では合計で 50) までクライアントを同時接続させることができます。

TOTALMEMBERCONNECTIONS しきい値で制御されるのはコーディネーター接続だけです。サブエージェントによる接続はしきい値のカウントに含まれません。

このしきい値は、同一インスタンス内で複数のデータベースを使用する場合に役立ちます。メンバーに TOTALMEMBERCONNECTIONS しきい値を設定することにより、1 つのデータベースからのクライアント接続がメンバーで使用可能な接続をすべて使用してしまうことのないようにすることができます。

max_connections データベース・マネージャー構成パラメーターは、データベース全体で使用する接続の最大数に対応できる大きさに設定してください。データベースに TOTALMEMBERCONNECTIONS しきい値を設定する場合は、**max_connections** をしきい値以上の値に設定する必要があります。同一インスタンス上で複数のデータベースを実行させる場合は、**max_connections** を、全データベースの接続の最大数に対応できる大きさに設定するようにしてください。データ・サーバーでは、同時にアクティブにされるデータベースの数をあらかじめ知ることができないため、この条件は検査されません。

注: TOTALMEMBERCONNECTIONS しきい値がある場合、キューのサイズは通常 0 に強制されますが、キューイングしきい値に対して CONTINUE しきい値アクションを指定した場合は、キューのサイズとして指定されている強制値に関係なくキューのサイズが事実上無制限になります。

TOTALSCMEMBERCONNECTIONS しきい値

TOTALSCMEMBERCONNECTIONS しきい値は、サービス・スーパークラスのコーディネーター・メンバーでの並行データベース接続の最大数を指定します。

タイプ 集約

定義ドメイン
サービス・スーパークラス

適用範囲
メンバー

トラッキングされる作業
接続

キューイング
可

単位 サービス・クラス内での同時接続の数

予測的か反応的か
予測的

サービス・クラス内で接続数が TOTALSCMEMBERCONNECTIONS しきい値に達すると、それよりも後にサービス・スーパークラスに加わるコーディネーター接続はキューに入れられます。これはキューが指定されたキュー・サイズに達するまで続きます。デフォルトのキュー・サイズはゼロで、これは接続がキューに入らないことを意味します。接続が TOTALSCMEMBERCONNECTIONS しきい値のキューに加わった場合、その接続は*過渡* 状態にあると見なされます。

注: キューイングしきい値に対して CONTINUE しきい値アクションを指定した場合は、キューのサイズとして指定されている強制値に関係なくキューのサイズが事実上無制限になります。

トラッキングされる接続には、新しいクライアント接続と、別のサービス・クラスからそのサービス・クラスに切り替わる既存のクライアント接続の両方が含まれます。接続のサービス・クラスの切り替えは、別のサービス・クラスにマップされている別のワークロード定義に関連付けることによって行われます。ワークロードの再評価はトランザクション境界でのみ行われるため、接続のサービス・クラスの切り替えはトランザクション境界でのみ行うことができます。ただし、WITH HOLD カーソルに関連付けられているリソースはトランザクション境界を越えて維持されるため、開かれている WITH HOLD カーソルがある接続では、サービス・スーパークラスの切り替えができません。接続コンセントレーターがオンになっている場合、切り替えられたアプリケーションはすべてサービス・クラスを離れます。. 後続のステートメントでそのアプリケーションへの切り替えがあるときは、アプリケーションは再度サービス・クラスに加わり、そこでしきい値を渡す必要があります。

キュー・サイズがしきい値に達すると、しきい値アクションがトリガーされます。TOTALSCMEMBERCONNECTIONS しきい値で制御されるのはコーディネーター接続だけです。サブエージェントによる接続はしきい値のカウントに含まれません。

TOTALMEMBERCONNECTIONS のしきい値を設定する際は、TOTALSCMEMBERCONNECTIONS に指定したしきい値に対応できる大きさの値を設定してください。例えば、データベースに 5 つのサービス・スーパークラスを定

義していて、それぞれのサービス・スーパークラスで TOTALSCMEMBERCONNECTIONS しきい値を 10 に設定している場合は、TOTALMEMBERCONNECTIONS しきい値を最低でも 50 以上に設定してください。

作業単位しきい値

作業単位しきい値は、個別の作業単位に制御を適用します。個別の作業単位のリソースの使用量が、それをトラッキングしているしきい値の上限を超えると、対応するアクションが起動され、アプリケーション (アプリケーションの強制終了の場合) または作業単位 (ロールバックの場合) のいずれかに 1 回適用されます。

UOWTOTALTIME しきい値

UOWTOTALTIME しきい値は、作業単位が DB2 エンジンで使用することができる最大時間を指定します。

タイプ 作業単位

定義ドメイン
データベース、ワークロード、サービス・スーパークラス

適用範囲
データベース

トラッキングされる作業
このトピックで後述される情報を参照。

キューイング
不可

単位 秒、分、時間、または日で表される時刻期間。指定する時間単位が秒の場合、その値は 10 の倍数でなければなりません。

予測的か反応的か
反応的

UOWTOTALTIME しきい値の STOP EXECUTION アクションは、作業単位をロールバックします。 FORCE APPLICATION アクションは、作業単位が属するアプリケーションを強制終了します。このしきい値に COLLECT ACTIVITY DATA オプションを指定することは可能ですが、無視されます。

ワークロード・ドメインに対して定義される UOWTOTALTIME しきい値は、サービス・スーパークラス・ドメインに対して定義される UOWTOTALTIME しきい値をすべてオーバーライドします。サービス・スーパークラス・ドメインに対して定義される UOWTOTALTIME しきい値は、データベース・ドメインに対して定義される UOWTOTALTIME しきい値をすべてオーバーライドします。

例

以下の例は、作業単位の実行時間が 10 分を超えたときに FORCE APPLICATION コマンドを発行するしきい値 FORCE10MINUTEUOW を作成します。

```
CREATE THRESHOLD FORCE10MINUTEUOW FOR DATABASE ACTIVITIES
   ENFORCEMENT DATABASE
   WHEN UOWTOTALTIME > 10 MINUTES
   FORCE APPLICATION;
```

継続中の作業の優先度変更

優先度変更は、進行中のアクティビティーの優先度が時間の経過とともに自動的に変更される、ワークロード管理に対するアプローチです。

アクティビティーの実行時間が長くなればなるほど、その優先度は低くなり、アクティビティーが受け取るリソースは少なくなります。優先度変更を使用して、長期実行アクティビティーを制御することにより、短期実行アクティビティーのスループットを改善できます。優先度変更のアプローチは、リソース制御がサービス・クラス間での作業の移動に対応する場合に機能します。つまり、既に処理中の作業のサービス・クラスが変わると、その作業 (および新規サービス・クラスの他の作業) が受け取るリソースにおいて、移動が実際に反映されます。CPU ディスパッチャーによって、あるいはオペレーティング・システムのワークロード管理製品との統合によって提供される明示的な CPU 制御を使用する場合に、このアプローチをインプリメントするのが最善です。

再マップによるアクティビティー優先順位の変更

システム・リソースは、サービス・クラスを使用することによって割り振られ、制御されます。優先度変更を使用する場合、アクティビティーの優先順位は、アクティビティーをあるサービス・クラスから別のサービス・クラスに移動することによって変更できます。新しいサービス・クラスのリソースがそれまでより増えた場合は優先順位が上がり、新しいサービス・クラスのリソースがそれまでより減った場合は優先順位が下がります。アクティビティーの移動は、プロセッサー時間や読み取り行数など、特定のリソースに関してあらかじめ決められた最大の使用状況に基づく、REMAP ACTIVITY アクションが定義されたしきい値に対して違反が発生すると行われます。新しいサービス・クラスにマップされたアクティビティーは、新しいリソース制約が適用されて実行を続けます。

短い照会の実行の高速化に役立つ単純アプローチでは、リソース優先順位のレベルが段階的に低くなる一連のサービス・クラスと、アクティビティーをサービス・サブクラス間で移動するしきい値アクションを定義します。このセットアップを使用すると、長期実行作業の優先順位を下げていく、つまりエージングを行うことができるので、短期実行作業の応答時間をおそらく改善できます。データ・サーバーで実行されていアクティビティーに関する詳しい知識は必要ありません。

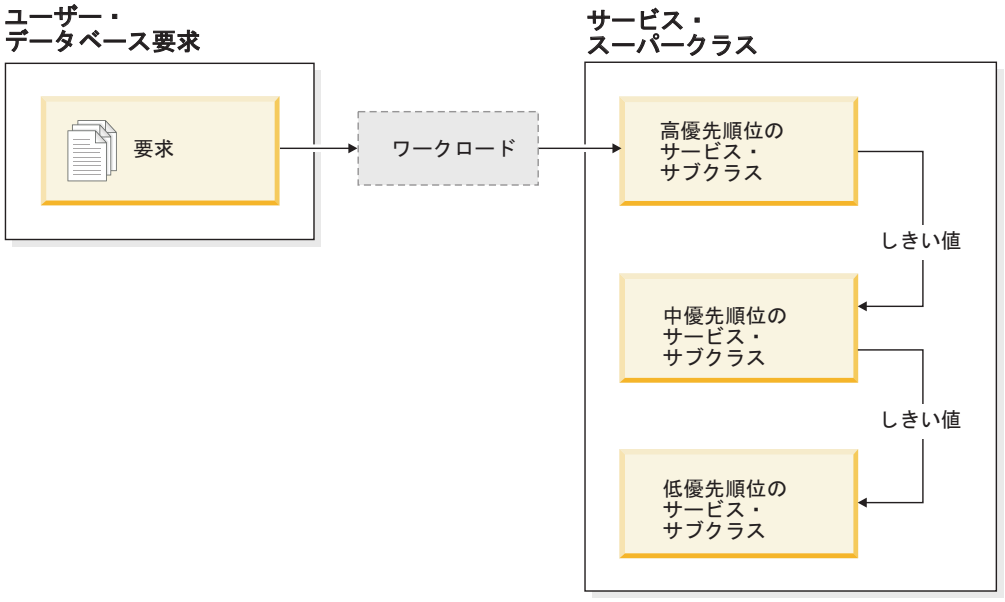


図 21. 優先順位が段階的に低くなる 3 つのサービス・クラスによる単純層構造セットアップ

このセットアップは、適用できるすべてのリソースについて、1 つのサービス・クラスに高優先順位を、2 番目のサービス・クラスには中優先順位を、3 番目のサービス・クラスには低優先順位をそれぞれ割り当てることによって作成できます。システムに入れられた作業は、最初のサービス・クラスに自動的に配置され、そのサービス・クラスの高優先順位の設定を使用して実行を開始します。サービス・クラスのそれぞれに実行中の使用時間または使用リソースを制限するしきい値も定義した場合、あるサービス・クラスのしきい値に対する違反が発生すると、作業は優先順位が 1 つ低いクラスに動的に再割り当てされます。この動的リソース制御は、作業が完了するか最低優先順位のクラスに置かれるまで繰り返し適用されます。最低優先順位のクラスでは、実行が完了するか強制的に停止させられるまで作業は続行します。

サービス・クラス内しきい値

アクティビティーの再マップは、どのサービス・クラス内しきい値についても使用可能です。サービス・クラス内しきい値は、アクティビティーが特定のサービス・サブクラスで実行中に使用できるリソースの量を制御します。リソースの例として、使用されるプロセッサー時間 (CPUTIMEINSC しきい値)、アプリケーションの 1 つのアクティビティーが 1 つのメンバーで読み取る行数 (SQLROWSREADINSC しきい値) があります。これらのしきい値は、他のアクティビティーしきい値とは異なります。他のアクティビティーしきい値は、アクティビティーの全存続期間中に使用されるリソースを制御します。

サービス・クラス内しきい値による制御はサービス・サブクラスを対象とするので、サービス・サブクラス・ドメインのみに対してサービス・クラス内しきい値を定義できます。サービス・クラス内しきい値は、DB2 ガバナーの規則 (モニター・エレメントであるプロセッサー時間および読み取り行数に従って行われる) と似た制御を提供します。

サービス・クラス内しきい値を REMAP ACTIVITY アクションと関連付けると、しきい値違反が発生したかどうかを、アクティビティーのために動作するエージェントがメンバーごとに定期的に検査します。エージェントはメンバー上のしきい値違反を検出すると、そのメンバー上のそのアクティビティー用の REMAP ACTIVITY アクションを起動し、自身をターゲット・サービス・サブクラスに再マップします。同一メンバー上のそのアクティビティーのために動作する他のエージェントもすべて、アクティビティーが再マップされたことを検出すると、自身をターゲット・サービス・サブクラスに再マップします。しきい値違反を検出してアクティビティーを再マップするエージェントは 1 つのみであり、そのエージェントがしきい値違反を検出して再マップを実行した後は、アクティビティーは再マップされたと見なされます。

サービス・サブクラス内のアクティビティー再マップに関する情報は、2 つのモニター・エレメントによって提供されます。 **act_remapped_in** モニター・エレメントはカウンターを備えています。このカウンターはサービス・サブクラスに再マップされたアクティビティーの数を記録し、アクティビティーがターゲット・サービス・サブクラスに再マップされるたびに増分されます。同様に、**act_remapped_out** モニター・エレメントのカウンターは、アクティビティーがソース・サービス・サブクラスから再マップされるたびに増分されます。さらに、モニター・エレメント **num_remaps** は、アクティビティーがサービス・サブクラス間で再マップされた合計回数をカウントします。

アクティビティーは、異なるサービス・サブクラスに複数回再マップできます。また、別のサービス・サブクラスに再マップされた後で元のサービス・サブクラスに戻ることができます。

サービス・クラス内しきい値は、調整されることなく、各メンバーのアクティビティーごとに個別に評価されます。メンバー間の調整が行われないため、あるメンバーで再マップされたアクティビティーと同じアクティビティーが、異なるメンバーの異なるサービス・サブクラスに同時に存在する可能性があります。

リモート・メンバーでアクティビティーのためのサブエージェント作業が完了した後、同じアクティビティーのための作業が同じメンバーにさらに送られると、その要求をメンバーに送信したエージェントと同じサービス・サブクラスでアクティビティーが再始動します。このサービス・サブクラスに対してサービス・クラス内しきい値が定義されている場合、リモート・メンバー上のアクティビティー用のタイマーまたはカウンターは、ゼロから再開します。

アクティビティーがネストされている場合、親アクティビティーと子アクティビティーは別々にトラッキングされます。したがって、子アクティビティーが過度の量のリソースを使用している場合、しきい値に違反しているのはそのアクティビティーのみであり、その親アクティビティーや兄弟アクティビティーではありません。

サービス・クラス内しきい値の使用

アクティビティーが獲得のために競合しなければならない主なリソースがプロセッサー時間であるデータ・サーバーでは、制御の第 1 手段として CPUTIMEINSC しきい値を使用します。表の多数の行を読み取る照会が主に入出力競合につながるデータ・サーバーでは、SQLROWSREADINSC を使用します。プロセッサー・アクテ

ィビティーと入出力アクティビティーがどちらも重いという組み合わせのシステムでは、CPUTIMEINSC しきい値と SQLROWSREADINSC しきい値の組み合わせを使用します。

サービス・サブクラスの相対的なエージェント優先順位を設定して、データ・サーバーがビジネス優先度に応じてアクティビティーを扱えるようにする必要があります。デフォルトのシステム・クラスのエージェント優先順位は、ユーザーが作成するどのユーザー定義サービス・クラスよりも常に高くする必要があります。パフォーマンスに対する悪影響を回避するためです。デフォルトの保守クラスのエージェント優先順位は、ユーザー定義サービス・クラスよりも低く設定できます。

アクティビティーを別のサービス・サブクラスに再マップする基準となる、アクティビティーがサービス・サブクラスで消費できる所定のリソースの量の設定は、ご使用の特定の環境に大きく依存します。各しきい値条件の最適値を見つけるには、データ・サーバー上のアクティビティーの処理をモニターする必要があります。使用可能な最大プロセッサー時間またはサービス・クラス内で読み取れる最大行数の設定が大きすぎると、各アクティビティーが必要とするリソース量に関係なく同じサービス・サブクラスでアクティビティーが開始して終了するという不適切な状態になります。最大プロセッサー時間または最大読み取り行数の設定が小さすぎると、最初にマップされたサービス・クラスで終了するアクティビティーがなくなり、最終的にはどのアクティビティーも、ビジネス優先度に関係なく別のサービス・クラスに再マップされてしまいます。どちらの場合も、層構造の構成がデータ・サーバー上の全体スループットの向上に役立っておらず、アクティビティーは事実上それぞれのビジネス優先度に従って扱われているわけではありません。

アクティビティーが消費できる所定のリソースの量を決定することに加えて、一部のしきい値では、データ・サーバーがしきい値違反を検査する頻度としての検査時間間隔を定義できます。この機能は、制御されるリソースの単位が消費されるたびにしきい値を検査するとコストがかかりすぎるしきい値のためのものであり、こうしたしきい値に対する違反を検出するための待ち時間を決定します。しきい値 CPUTIME および SQLROWSREAD と、サービス・クラス内でそれらに対応する CPUTIMEINSC および SQLROWSREADINSC は、検査時間間隔をサポートします。シリアル・データベース・インスタンスの場合、検査時間間隔は、しきい値違反検査の間隔として経過すべき実時間の長さと等しくなります。マルチメンバー・データベース環境または SMP インスタンスの場合は、アクティビティー用のエージェントが複数存在するためにプロセッサー時間が同時に累積する可能性があることを考慮に入れて、検査時間間隔を実経過時間より小さい値に設定する必要があります。マルチメンバー・データベース環境または SMP インスタンスの場合の検査時間間隔を概算するには、検査間の実経過時間を、アクティビティーの並列処理の度合いで除算します。この結果値を CHECKING EVERY 節に使用します。

例: 単一メンバー・データベースで、プロセッサー時間として 30 秒が消費されると CPUTIMEINSC しきい値によって REMAP ACTIVITY アクションが起動されるようにするとします。この場合は、検査時間間隔を 30 秒に設定し、消費されたプロセッサー時間が 30 秒を超えるまでにこのしきい値アクションが起動されるようにします (使用プロセッサー時間が実経過時間を上回ることはありません)。マルチメンバー・データベース環境で、設定が 5 秒で検査時間間隔が 5 秒の CPUTIMEINSC を定義するとします。アクティビティーのために動作するエージェントとして、コーディネーター・メンバー・エージェントが 1 つとサブエージェン

トが 4 つあります。この場合、実時間は 1 秒でも、アクティビティーは CPU 時間として 5 秒を消費する可能性があります。エージェントが 5 つ存在するためにプロセッサー時間の各 1 秒が同時に累積するからです。アクティビティーがプロセッサー時間として 5 秒の倍数を消費しないようにするには、この場合は検査時間間隔を 1 秒に設定する必要があります。

しきい値の使用法に関する追加情報は、サンプル層構造スクリプトおよび優先度変更のシナリオを参照してください。

再マップのしきい値への影響

REMAP ACTIVITY アクションを介した再マップ後にどのしきい値が引き続き適用されるかは、しきい値が特定のサービス・サブクラスにのみ適用されるのかアクティビティーの存続期間中適用されるのかに依存します。

アクティビティーを新しいサービス・サブクラスに再マップすると、CPUTIMEINSC や SQLROWSREADINSC などのサービス・クラス内しきい値のみが変わります。これらのサービス・クラス内しきい値は、ソース・サービス・サブクラスから離れたアクティビティーには影響を与えなくなり、ターゲット・サブクラスの対応するしきい値に置き換えられます (それらのしきい値が定義されている場合)。アクティビティーが最初にマップされたサービス・サブクラスでの他のアクティビティーしきい値はすべて、変わりません。また、適用できるしきい値タイマーおよびしきい値カウンターはリセットされません。ターゲット・サービス・サブクラスに定義した他のしきい値に対するアクティビティーの再評価は行われません。

例えば、しきい値を持つ 2 つのサービス・サブクラスが以下のように定義されているとします。

- サービス・サブクラス A。次のしきい値が定義されています。
  - ACTIVITYTOTALTIME 存続期間しきい値 TH1。30 分経過後に STOP EXECUTION アクションが起動します。
  - SQLROWSREADINSC サービス・クラス内しきい値 TH2。2000 を超える行が読み取られるとサービス・サブクラス B への REMAP ACTIVITY アクションが起動します。
- サービス・サブクラス B。次のしきい値が定義されています。
  - ACTIVITYTOTALTIME 存続期間しきい値 TH3。5 分経過後に STOP EXECUTION アクションが起動します。
  - SQLROWSREADINSC しきい値 TH4。1000 を超える行が読み取られると STOP EXECUTION アクションが起動します。

アクティビティーがシステムのサービス・サブクラス A に入れられると、そのアクティビティーに TH1 と TH2 の両方のしきい値が適用されます。照会の評価中にアクティビティーが 2000 を超える行を読み取ると、アクティビティーはサービス・サブクラス B に動的に再マップされます。アクティビティーがサブクラス B に再マップされたため、適用できるサービス・クラス内しきい値が変わり、TH2 ではなく TH4 がアクティビティーに適用されるようになります。両方のしきい値用のカウンターがゼロにリセットされます。アクティビティーが元のサービス・サブクラスで 2000 を超える行を読み取っていても、TH4 用のカウンターはゼロから再開します。アクティビティーがサービス・サブクラス B で実行中に 1000 を超える行を

読み取ると、しきい値 TH4 に違反します。しきい値 TH1 はアクティビティーの存続期間中適用されるので、アクティビティーが別のサブクラスで実行されるようになっても、引き続き適用されます。しきい値 TH3 は、再マップされたアクティビティーに対する制御は一切行いません。アクティビティーが最初に入れられたサービス・サブクラスには、アクティビティーの実行開始時にこのしきい値が適用されていないからです。

優先度変更のサンプル・スクリプト

提供されているサンプル・スクリプトを使用して、層構造のサービス・クラス構成をデータ・サーバー上に短時間で作成します。層構造の構成を使用すると、実行時間の長い照会の優先順位を時間の経過とともに下げること (優先度変更とも呼ばれる) によって、特定のパフォーマンス目標に取り組むことができます。スクリプトを環境に適合させるときに、独自のビジネス優先順位に応じてスクリプトを変更することもできます。

wlmtiersdefault.db2 と wlmtierstimerons.db2 の 2 つは、優先度変更をデータ・サーバーで使用して、全体のスループットを向上させる方法を示すためのサンプル・スクリプトです。DB2 ワークロード・マネージャーを使用すると、データ・サーバー全体のスループットの向上に貢献する制御を行うことができます。しかし、スクリプトおよび優先度変更から十分な益を得るには、一般に、長時間にわたってデータ・サーバーを稼働させ、その処理が実行される方法をモニターした後、それに応じてサービス・クラスとしきい値の設定を調整する必要があります。

スクリプトは、インストール・ディレクトリー下の samples/admin_scripts ディレクトリーにあります。

wlmtiersdefault.db2 および wlmtierstimerons.db2 スクリプトは、3 つのサービス・サブクラスが共通のスーパークラスの下に作成されます。それらのサービス・サブクラスでは、高から低へ段階的に低くなるリソースの設定と、プロセッサー時間の消費に応じてアクティビティーを移動または再マップする CPUTIMEINSC しきい値が設定されています。これらのスクリプトは、アクティビティーが最初にデータ・サーバーに入ったときにサービス・クラスにマップされる方法が異なります。3 つ目のスクリプト wlmtiersdrop.db2 は、他の 2 つのサンプル層構造スクリプトによって作成された WLM オブジェクトをドロップします。

wlmtiersdefault.db2

データ・サーバーに入るアクティビティーはすべて、作業のタイプに関係なく、高優先順位のサービス・サブクラス WLM_SHORT にマップされます。アクティビティーが費やすことのできる最大プロセッサー時間を超えない限り、最も高い優先順位がそのアクティビティーに割り当てられている間、そのアクティビティーは高優先順位のサービス・サブクラスで完了します。過剰なプロセッサー時間を費やすアクティビティーはまず、しきい値 REMAP ACTIVITY アクションによって中優先順位のサービス・サブクラス WLM_MEDIUM に再マップされます。それでもまだ、そのアクティビティーが完了するまでに、割り当てられたプロセッサー時間を超えてしまう場合は、低優先順位のサービス・サブクラス WLM_LONG に再マップされます。この優先順位で、完了するまでアクティビティーの処理が続きます。

CPUTIMEINSC しきい値によって再マップできないアクティビティーは、サービス・サブクラス WLM_MEDIUM に直接マップされ、その後、そのサービス・サブクラスに残ります。

wlmtierstimerons.db2

データ・サーバーに入る DML アクティビティーは、その見積コストに応じて評価され、3 つのサービス・サブクラスのうちの 1 つにマップされます。短いと評価される DML アクティビティーは高優先順位のサービス・サブクラス WLM_SHORT にマップされ、中程度の長さと評価される DML アクティビティーは中優先順位を受け取る WLM_MEDIUM サービス・サブクラスにマップされ、長い DML アクティビティーは最も低い優先順位を受け取る WLM_LONG サービス・サブクラスにマップされます。非 DML アクティビティーは、最も高い優先順位のサービス・サブクラスに入ります。処理が行われるにつれ、アクティビティーによって費やされるプロセッサー時間が、サービス・サブクラスに割り当てられているプロセッサー時間を超えると、しきい値 REMAP ACTIVITY アクションによって、次に最も低い優先順位のサービス・サブクラスに連続的に再マップされます。これは、最も低い優先順位のサービス・サブクラスに再マップされるまで続きます。その優先順位で、完了するまでアクティビティーの処理が続きます。CPUTIMEINSC しきい値によって再マップできないアクティビティーは、サービス・サブクラス WLM_MEDIUM に直接マップされ、その後、そのサービス・サブクラスに残ります。

wlmtiersdrop.db2

このスクリプトは、スクリプト wlmtiersdefault.db2 および wlmtierstimerons.db2 によって作成されたすべての DB2 ワークロード・マネージャー・サービス・クラス、しきい値、ワークロード、ワーク・クラス・セット、およびワーク・アクション・セットをドロップします。

デフォルトでは、wlmtiersdefault.db2 および wlmtierstimerons.db2 スクリプトは、以下のサービス・クラスおよびしきい値の定義を使用します。

表 40. スクリプトによって作成される CPU シェアおよびプリフェッチ優先順位の設定を持つサービス・クラス

サービス・クラス	CPU シェア	プリフェッチ優先順位
WLM_SHORT (高優先順位)	6000	高
WLM_MEDIUM (中優先順位)	3000	中
WLM_LONG (低優先順位)	1000	低
デフォルト・システム・クラス	デフォルト	高
デフォルト保守クラス	デフォルト	低

表 41. スクリプトによって作成されるしきい値

しきい値	再マップする前にサービス・クラスで使用できる最大プロセッサー時間
WLM_TIERS_REMAP_SHORT_TO_MEDIUM	30 秒
WLM_TIERS_REMAP_MEDIUM_TO_LONG	30 秒

wlmtiersdefault.db2 サンプル・スクリプトは、以下のワーク・アクション・クラス・セットの作成も行います。以下のワーク・アクション・クラス・セットは、CPUTIMEINSC しきい値によって再マップできないアクティビティーを直接 WLM_MEDIUM サービス・サブクラスにマップするために使用されます。実行期間中、これらのアクティビティーは WLM_MEDIUM サービス・サブクラスに残ります。

表 42. wlmtiersdefault.db2 サンプル・スクリプトによって作成されるワーク・クラス・セット

ワーク・クラス	ワーク・アクション
WLM_DML_WC	DML アクティビティーについては、最初、サービス・クラス WLM_SHORT にマップされます。これらのアクティビティーは、CPUTIMEINSC しきい値によって再マップすることができます。
WLM_CALL_WC	CALL アクティビティーについては、最初、サービス・クラス WLM_SHORT にマップされます。これらのアクティビティーは、CPUTIMEINSC しきい値によって再マップすることができます。
WLM_OTHER_WC	CPUTIMEINSC しきい値によって再マップできないアクティビティーについては、サービス・クラス WLM_MEDIUM にマップされます。これらのアクティビティーは WLM_MEDIUM サービス・サブクラスに残ります。

wlmtierstimerons.db2 サンプル・スクリプトは、以下のワーク・アクション・セットおよびワーク・クラス・セットの作成も行います。以下のワーク・クラス・セットは、その見積コストに応じてアクティビティーをマップするために使用されます。

表 43. wlmtierstimerons.db2 サンプル・スクリプトによって作成されるワーク・クラス・セット

ワーク・クラス	見積コストの範囲 (timeron 単位) および作業アクション
WLM_SHORT_DML_WC	見積コストが 0 から 999 timeron の DML アクティビティーについては、最初、サービス・クラス WLM_SHORT にマップされます。これらのアクティビティーは、CPUTIMEINSC しきい値によって再マップすることができます。
WLM_MEDIUM_DML_WC	見積コストが 1000 から 99 999 timeron の DML アクティビティーについては、最初、サービス・クラス WLM_MEDIUM にマップされます。これらのアクティビティーは、CPUTIMEINSC しきい値によって再マップすることができます。

表 43. wlmtierstimerons.db2 サンプル・スクリプトによって作成されるワーク・クラス・セット (続き)

ワーク・クラス	見積コストの範囲 (timeron 単位) および作業アクション
WLM_LONG_DML_WC	見積コストが 100 000 から無限大 timeron の DML アクティビティーについては、サービス・クラス WLM_LONG にマップされます。
WLM_CALL_WC	CALL アクティビティーについては、最初、サービス・クラス WLM_SHORT にマップされます。これらのアクティビティーは、CPUTIMEINSC しきい値によって再マップすることができます。
WLM_OTHER_WC	再マップできないアクティビティーについては、サービス・クラス WLM_MEDIUM にマップされます。

環境に合わせたスクリプトの変更

環境に合わせてサンプル・スクリプトを変更するときに考慮すべき最も重要な設定は、各サービス・クラスで使用できる最大プロセッサー時間です。各サービス・サブクラスでアクティビティーが費やすことのできるプロセッサー時間は、ご使用の環境によって大きく異なります。最適な値を見つけるには、データ・サーバーでアクティビティーが処理される方法をモニターする必要があります。デフォルトでは、wlmtiersdefault.db2 スクリプトと wlmtierstimerons.db2 スクリプトのどちらも、どちらか一方が活動化されると、イベント・モニター・レコードをしきい値違反イベント・モニターに記録します。その際、アクティビティー・イベント・モニターをオンにして使用可能にし、アクティビティー・データを収集するためのオプションが使用されています (ただし、追加のオーバーヘッドが発生するというコストが生じます)。wlmtiersdefault.db2 では、各サービス・クラスで使用できる最大プロセッサー時間の値が大きすぎる場合、アクティビティーのほとんどが、実際に必要とするプロセッサー時間に関係なく、常に高優先順位のクラスで開始および終了するようになります。設定された最大プロセッサー時間の値が小さすぎる場合は、アクティビティーが高優先順位のサービス・クラスで終了しなくなります。すべてのアクティビティーは、ビジネス優先順位に関係なく、中または低優先順位のサービス・クラスに再マップされることになります。どちらの場合も、スクリプトはデータ・サーバー全体のスループットを向上させることができず、アクティビティーは事実上そのビジネス優先順位に応じて扱われません。 wlmtierstimerons.db2 においても軽度ですが、同様の問題が生じます。このスクリプトでは、最初にアクティビティーを見積コストに応じてサービス・サブクラスにマップして区分します。各サービス・クラスで使用できる最大プロセッサー時間の設定が正しくないと、アクティビティーによって費やされるプロセッサー時間が長すぎる場合にそのアクティビティーをより適切なサービス・サブクラスに再マップできなかったり、より高いビジネス優先順位があるにもかかわらず性急に再マップされたりします。

スクリプトによって作成される特定の DB2 ワークロード・マネージャー・オブジェクトについて、またその実行方法について詳しくは、スクリプトを参照してください。

サンプル・シナリオ

資料には、優先度変更を利用するようにデータ・サーバー上のサンプル層構造スクリプトを適合させる方法を示す 2 つの例が含まれています。

シナリオ: 優先度変更によるリソース集中ビジネス・インテリジェンス・レポートの制御

以下のシナリオは、高コストのビジネス・インテリジェンス・レポートの優先順位を動的に下げるようデータ・サーバーを構成する方法を示しています。このビジネス・インテリジェンス・レポートは、他の照会にとってのシステム・パフォーマンスを維持するうえで、実行開始前に識別することができません。

問題: どのエンド・ユーザーも実行できるが非常のコストのかかるビジネス・インテリジェンス・レポートがあります。このレポートが実行されるといつも、システムのパフォーマンスが損なわれます。このレポートの生成に使用されるフロントエンド・ツールは、クライアント情報を一切設定しません。クライアント情報があれば、それを使用してあらかじめこのレポートを識別し、ワークロードを使用して低優先順位サービス・クラスにマップできるところです。

ソリューション: wlmtiersdefault.db2 サンプル層構造スクリプトを使用して、層構造の構成のデータ・サーバーを構成できます。これは、プロセッサー集中アクティビティーの優先順位をその存続期間中に動的に下げる、つまり繰り下げることで、他のすべてのユーザーにとってデータ・サーバーのパフォーマンスが損なわれないようにします。ワークロードが最初にすべての作業を高優先順位サービス・サブクラスにマップした後、CPUTIMEINSC サービス・クラス内しきい値は、消費されるプロセッサー時間に基づいてこの高コスト・レポートを検出します。アクティビティーが最大許容プロセッサー時間を使用して CPUTIMEINSC しきい値に違反すると、REMAP ACTIVITY がそのアクティビティーをより低い優先順位のサービス・サブクラスに移動します。このアクティビティーは、プロセッサー時間使用量に応じて、最低優先順位サービス・サブクラスで実行されるまで再マップしていくことができます。最低優先順位サービス・サブクラスでは、完了するか手動で介入されるまで、アクティビティーが続行します。しきい値を超えない他のアクティビティーは、高優先順位サービス・サブクラスで実行を続行し、そこではより高いエージェント優先順位を受け取ります。

しきい値違反イベント・モニターを作成した場合、アクティビティーが再マップされるたびに、イベント・モニター・レコードがログに記録されます。再マップされたアクティビティーに関する追加情報を収集してさらに詳しく調べる場合は、wlmtiersdefault.db2 スクリプト内の ALTER THRESHOLD ステートメントに COLLECT ACTIVITY DATA 節を追加します。このスクリプトを再実行するだけで変更が有効になります。

ワークロードを一定期間実行したら、次のように WLM_GET_SERVICE_SUBCLASS_STATS 表関数を使用して、サービス・サブクラス間で再マップされたアクティビティーの数を調べることができます。

```
SELECT substr(service_superclass_name,1,21) AS superclass,
       substr(service_subclass_name,1,21) AS subclass,
       substr(char(coord_act_completed_total),1,10) AS completed,
       substr(char(act_remapped_in),1,10) AS remapped_in,
       substr(char(act_remapped_out),1,10) AS remapped_out,
       substr(char(last_reset),1,19) AS last_reset
FROM table( WLM_GET_SERVICE_SUBCLASS_STATS(
               CAST(NULL AS VARCHAR(128)),
```

```
              CAST(NULL AS VARCHAR(128)),
              -2 )
         ) AS TF_subcls_stats@

  SELECT SUBSTR(WORKLOAD_NAME,1,19) AS WL_NAME,
         COORD_ACT_LIFETIME_AVG,
         COORD_ACT_LIFETIME_STDDEV
  FROM TABLE(WLM_GET_WORKLOAD_STATS
        (CAST(NULL AS VARCHAR(128)), -2))
         AS WLSTATS
  ORDER BY WL_NAME@

SUPERCLASS            SUBCLASS              COMPLETED  REMAPPED_IN REMAPPED_OUT LAST_RESET
--------------------- --------------------- ---------- ----------- ------------ --------------------
SYSDEFAULTSYSTEMCLASS SYSDEFAULTSUBCLASS    0          0           0            2008-10-06-20.53.47
SYSDEFAULTMAINTENANCE SYSDEFAULTSUBCLASS    3          0           0            2008-10-06-20.53.47
SYSDEFAULTUSERCLASS   SYSDEFAULTSUBCLASS    0          0           0            2008-10-06-20.53.47
WLM_TIERS             SYSDEFAULTSUBCLASS    0          0           0            2008-10-06-20.53.47
WLM_TIERS             WLM_SHORT             999        0           35           2008-10-06-20.53.47
WLM_TIERS             WLM_MEDIUM            19         35          16           2008-10-06-20.53.47
WLM_TIERS             WLM_LONG              16         16          0            2008-10-06-20.53.47

  7 record(s) selected.
```

より低い優先順位のサービス・サブクラスに再マップされているアクティビティーがまったくないかごくわずかしかないことが分かった場合は、スクリプト内の ALTER THRESHOLD ステートメントで使用される CPUTIMEINSC しきい値および検査時間間隔を小さくし、サービス・クラス層全体のアクティビティーのマッピングをビジネス優先順位に従って改善してください。大半またはほとんどすべてのアクティビティーがより低い優先順位のサービス・サブクラスに再マップされている場合は、ALTER THRESHOLD ステートメントの CPUTIMEINSC しきい値および検査時間間隔を大きくし、より多くのアクティビティーがより高い優先順位で完了できるようにしてください。変更を完了したら、wlmtiersdefault.db2 スクリプトを再実行して変更を有効にします。

シナリオ: 誤ってマップされた照会を優先度変更により再マップする

以下のシナリオは、当初見積もりを超えたプロセッサー時間を消費しているアクティビティーを、他の照会にとってのシステム・パフォーマンスを維持するために動的に再マップする、つまり優先順位をエージングによって下げるようにデータ・サーバーを構成する方法を示しています。

問題: 高コスト・アクティビティーについては、見積もり SQL コストに基づいて低めの優先順位のサービス・サブクラスにマップし、こうしたアクティビティーがより低コストで短いアクティビティーのパフォーマンスに影響を与えないようにします。この処置は既に行われているかもしれません。このようなマッピングは、サービス・スーパークラス・レベルのワーク・アクション・セットを定義することによって行えます。しかし、例えば統計が最新でないために見積もり SQL コストが正しくない場合は、高コスト・アクティビティーが誤って高優先順位サービス・サブクラスにマップされるかもしれません。その結果、高コスト・アクティビティーは過度の量のリソースを消費し始め、他のすべての高優先順位アクティビティーが犠牲になります。

ソリューション: wlmtierstimerons.db2 サンプル層構造スクリプトを使用して、層構造の構成のデータ・サーバーを構成できます。このデータ・サーバーは、着信アクティビティーをその見積コストに基づいて評価し、エージェント優先順位が異なる 3 つのサービス・サブクラスのうちの 1 つにマップします。アクティビティーがプロセッサー時間を消費しすぎると、データ・サーバーはアクティビティーをパフォーマンス層間で再マップすることにより、アクティビティーの優先順位をその存続期間中に動的に下げます。このように優先順位を下げるためにアクティビティーを再マップする動的なプロセスは、優先度変更とも呼ばれます。

アクティビティーがその初期サービス・クラスにマップされて実行を開始すると、スクリプトは CPUTIMEINSC サービス・クラス内しきい値を使用して、アクティビティーが消費できるプロセッサー時間を制御します。アクティビティーが最大許容プロセッサー時間を使用してしきい値に違反すると、REMAP ACTIVITY アクションが起動され、アクティビティーはより低いエージェント優先順位のサービス・サブクラスに移されます。このアクティビティーは、プロセッサー時間使用量に応じて、最低優先順位サービス・サブクラスで実行されるまで再マップしていくことができます。最低優先順位サービス・サブクラスでは、完了するか手動で介入されるまで、アクティビティーが続行します。

アクティビティーが再マップされるたびに、イベント・モニター・レコードがログに記録されます。再マップされたアクティビティーに関する追加情報を収集してさらに詳しく調べる場合は、wlmtiersdefault.db2 スクリプト内の ALTER THRESHOLD ステートメントに COLLECT ACTIVITY DATA 節を追加します。このスクリプトを再実行するだけで変更が有効になります。

ワークロードを一定期間実行したら、次のように WLM_GET_SERVICE_SUBCLASS_STATS 表関数を使用して、サービス・サブクラス間で再マップされたアクティビティーの数を調べることができます。

```
SELECT substr(service_superclass_name,1,21) AS superclass,
       substr(service_subclass_name,1,21) AS subclass,
       substr(char(coord_act_completed_total),1,10) AS completed,
       substr(char(act_remapped_in),1,10) AS remapped_in,
       substr(char(act_remapped_out),1,10) AS remapped_out,
       substr(char(last_reset),1,19) AS last_reset
FROM table( WLM_GET_SERVICE_SUBCLASS_STATS(
                CAST(NULL AS VARCHAR(128)),
                CAST(NULL AS VARCHAR(128)),
                -2 )
          ) AS TF_subcls_stats@

  SELECT SUBSTR(WORKLOAD_NAME,1,19) AS WL_NAME,
         COORD_ACT_LIFETIME_AVG,
         COORD_ACT_LIFETIME_STDDEV
  FROM TABLE(WLM_GET_WORKLOAD_STATS
        (CAST(NULL AS VARCHAR(128)), -2))
         AS WLSTATS
  ORDER BY WL_NAME@
```

```
SUPERCLASS           SUBCLASS             COMPLETED  REMAPPED_IN REMAPPED_OUT LAST_RESET
-------------------- -------------------- ---------- ----------- ------------ -------------------
SYSDEFAULTSYSTEMCLASS SYSDEFAULTSUBCLASS    0          0           0            2008-10-06-20.59.27
SYSDEFAULTMAINTENANCE SYSDEFAULTSUBCLASS    3          0           0            2008-10-06-20.59.27
SYSDEFAULTUSERCLASS   SYSDEFAULTSUBCLASS    0          0           0            2008-10-06-20.59.27
WLM_TIERS             SYSDEFAULTSUBCLASS    0          0           0            2008-10-06-20.59.27
WLM_TIERS             WLM_SHORT             651        0           5            2008-10-06-20.59.27
WLM_TIERS             WLM_MEDIUM            36         5           7            2008-10-06-20.59.27
WLM_TIERS             WLM_LONG              16         7           0            2008-10-06-20.59.27

  7 record(s) selected.
```

このシナリオでは、サービス・サブクラス間で再マップされているアクティビティーは比較的少数です。ほとんどの場合、アクティビティーは見積コストに基づいて、適切なサービス・サブクラスに最初にマップされるからです。アクティビティーが通常は WLM_SHORT または WLM_LONG サービス・クラスでのみ完了していることが分かった場合は、スクリプト内の ALTER WORK CLASS SET ステートメントで使用される見積コスト値を調整して、サービス・クラス層全体にわたってアクティビティーのマッピングを改善します。つまり、短めのアクティビティーが WLM_SHORT_DML_WC ワーク・クラスに、長めのアクティビティーが WLM_MEDIUM_DML_WC または WLM_LONG_DML_WC ワーク・クラスにそれ

ぞれマップされるようにします。アクティビティーのほとんどが再マップされていることが分かった場合は、ALTER THRESHOLD ステートメントで使用されるしきい値を大きくして、アクティビティーのサービス・サブクラスへの初期マッピングを改善します。変更を完了したら、wlmtierstimerons.db2 スクリプトを再実行して変更を有効にします。

サービス・サブクラス間のアクティビティーの再マップ

再マップを使用可能にするには、CREATE ステートメントおよび ALTER THRESHOLD ステートメントで REMAP ACTIVITY アクションを指定します。再マップ・アクションがしきい値違反によってトリガーされると、同じスーパークラスの下にある 1 つのサービス・サブクラスから別のサービス・サブクラスにアクティビティーが移動されます。

始める前に

別のサービス・サブクラスへの再マップを可能にするには、アクティビティーのオリジナルのサービス・サブクラスと同じサービス・スーパークラスの下にターゲットのサービス・サブクラスが存在している必要があります。ターゲットまたはオリジナルのいずれかのサービス・サブクラスをスーパークラスのデフォルトのサブクラスにすることができます。 REMAP ACTIVITY アクションは、デフォルトのシステム・クラス、デフォルトの保守クラス、またはデフォルトのユーザー・クラスの下にあるサービス・サブクラスには適用できません。

このタスクについて

REMAP ACTIVITY アクションは、アクティビティーを同じサービス・スーパークラス内の別のサービス・サブクラスに移動します。再マップは、CPUTIMEINSC や SQLROWSREADINSC など、いずれかのサービス・クラス内しきい値で使用できます。このようにアクティビティーを再マップする動的なプロセスは、時間の経過とともに優先順位を下げるために使用します。これは優先度変更とも呼ばれます。時間の経過とともに特定のアクティビティーの優先順位を下げると、システム・リソースを解放でき、そのリソースをビジネス上の重要度が高い他のアクティビティーに適用できます。

DATATAGINSC サービス・クラス内しきい値と共に REMAP ACTIVITY アクションを使用すると、アクセス対象として見積もられるデータに基づいて別の DB2 サービス・サブクラスにアクティビティーをマップすることができます。

アクティビティーに対して作業を行っているエージェントは、メンバー間の調整なしで、各メンバーのしきい値が違反されているかどうかを定期的に検査します。いずれかのエージェントがメンバー上でサービス・クラス内しきい値違反を検出すると、そのエージェントはメンバー上のアクティビティーに対して REMAP ACTIVITY アクションをトリガーし、その後、ターゲットのサービス・サブクラスに再マップされます。これが行われた後、アクティビティーは再マップされたと見なされます。同じメンバー上のアクティビティーに対して作業を行っている他のすべてのエージェントも、アクティビティーが再マップされたことを検出すると、ターゲットのサービス・サブクラスに再マップされます。

制約事項

ターゲットのサービス・サブクラスをオリジナルのサービス・サブクラスと同じにすることはできません。オリジナルのサービス・サブクラスに再マップするにはまず、別のサービス・サブクラスに再マップする必要があります。

使用不可のサービス・サブクラスにアクティビティーが再マップされると、そのアクティビティーは使用不可のサブクラスによって拒否されたかのように扱われ、エラー・メッセージ -4714 がクライアントに戻されます。

手順

1. 優先度変更によって制御するアクティビティーを識別します。層構造のセットアップは、REMAP ACTIVITY しきい値アクションが定義されたサービス・クラス内しきい値を持つ同じサービス・スーパークラスの下のサービス・サブクラスで構成されます。基本的な優先度変更の例および開始点となるサンプル・シナリオについては、以下を参照してください。
   a. 174 ページの『継続中の作業の優先度変更』
   b. 183 ページの『シナリオ: 優先度変更によるリソース集中ビジネス・インテリジェンス・レポートの制御』
   c. 184 ページの『シナリオ: 誤ってマップされた照会を優先度変更により再マップする』
2. アクティビティーがマップされる別のサービス・サブクラスを選択します。これには、アクティビティーの実行が開始されるときに最初にマップされるサービス・サブクラスと、アクティビティーがその存続時間中に再マップされるその他いずれかのサービス・サブクラスまたはクラスの両方が含まれます。サービス・クラスの詳細およびその作成方法については、77 ページの『サービス・クラスでのリソース割り当て』を参照してください。
3. アクティビティーを制御するしきい値を作成または変更します。しきい値について詳しくは、174 ページの『継続中の作業の優先度変更』を参照してください。
   a. サービス・クラス内しきい値を定義し、それに REMAP ACTIVITY アクションが含まれるようにします。このアクションは、しきい値が違反されたときにトリガーされます。サービス・クラス内しきい値は、アクティビティーが関連するサービス・サブクラスにマップされている間のみそのアクティビティーに適用され、またそのアクティビティーの影響を受けます。影響を受けるカウンターおよびタイマーは、再マップの後、リセットされます。アクティビティーが再マップされるたびにしきい値違反レコードがログに記録されるかどうかを検討します。そのレコードは、アクティビティーがどのサービス・クラスの実行に時間を費やしたかに関する情報を提供します。その情報は、パフォーマンスの分析に使用することができます。サービス・サブクラス間のアクティビティーの再マップが頻発すると、ロギングされるしきい値違反レコードによって、大量のディスク・スペースが消費され始める可能性があります。
   b. アクティビティーの存続時間に適用するしきい値を定義することもできます。しかし、アクティビティーの存続時間中ずっと適用され続けるのは、アクティビティーが最初にマップされた最初のサービス・サブクラスのしきい値だけです。後でアクティビティーが再マップされるいずれかのサービス・サブクラスに対してしきい値を定義する場合も、そのしきい値は適用されません。

4. 変更をコミットします。 変更をコミットすると、しきい値が SYSCAT.THRESHOLDS ビューに追加されます。
5. サービス・クラス内しきい値のターゲットとするアクティビティーの実行とその存続時間中の進行のモニターをデータ・サーバーに許可します。アクティビティーがサービス・クラス内しきい値を違反しない限り、実行中はオリジナルのサービス・サブクラスに残ります。アクティビティーの実行中にサービス・クラス内しきい値違反が発生すると、アクティビティーは REMAP ACTIVITY アクションをトリガーします。このアクションは、アクティビティーを動的に別のサービス・サブクラスに再マップします。再マップされた後、アクティビティーは実行を続け、ターゲットのサービス・サブクラスに設定したリソース制約によって制御されるようになります。
6. 必要に応じて、所定のパフォーマンス目標を達成するために、優先度変更に対するアプローチを改善します。

例

以下の例は、時間の経過とともに継続中のアクティビティーの優先順位を下げる、つまりエージングによって下げる単純な 3 層セットアップを作成します。単一のスーパークラス A の下の 3 つのサービス・サブクラスは、照会すべての必須の実行場所となる実行環境を提供します。デフォルトのユーザー・ワークロードが着信した照会をサービス・サブクラス A1 にマップするとします。このサービス・サブクラスは、実行時間の短い照会を短時間で実行するための高優先順位のサブクラスです。中優先順位のサービス・サブクラス A2 は、実行時間の長い照会を実行するためのサービス・サブクラスです。ただし、これにはより厳格なリソース制御があります。サービス・サブクラス A3 は、完了に過剰なプロセッサー時間を必要とする非常に大きな照会を受け入れます。

3 つのしきい値は、進行中の照会のリソースの消費を制御します。 1 分未満のプロセッサー時間で完了する場合のみ、照会は高優先順位のサービス・サブクラス A1 での実行を許可されます。 1 分のプロセッサー時間が経過すると、しきい値 T1 はアクティビティーを自動的にサブクラス A2 に再マップします。費やされるプロセッサー時間が 10 分未満の場合、アクティビティーはこのサブクラスで実行を続けます。 10 分のプロセッサー時間が経過しても照会が完了しない場合、しきい値 T2 は最も低い優先順位のサービス・サブクラス A3 にアクティビティーを再マップします。サブクラス A3 の照会は、無期限に続けることができます。ただし、費やされたプロセッサー時間が 1 時間を超えると、イベント・モニター・レコードはログに記録され、詳細を含むアクティビティー・データが収集されます。

```
CREATE SERVICE CLASS A
CREATE SERVICE CLASS A1 UNDER A
CREATE SERVICE CLASS A2 UNDER A
CREATE SERVICE CLASS A3 UNDER A

CREATE THRESHOLD T1 FOR SERVICE CLASS A1 UNDER A
  ACTIVITIES ENFORCEMENT MEMBER
  WHEN CPUTIMEINSC > 1 MINUTE REMAP ACTIVITY TO A2

CREATE THRESHOLD T2 FOR SERVICE CLASS A2 UNDER A
  ACTIVITIES ENFORCEMENT MEMBER
  WHEN CPUTIMEINSC > 10 MINUTES REMAP ACTIVITY TO A3

CREATE THRESHOLD T3 FOR SERVICE CLASS A3 UNDER A
```

```
ACTIVITIES ENFORCEMENT MEMBER
WHEN CPUTIMEINSC > 1 HOUR LOG EVENT MONITOR RECORD
COLLECT ACTIVITY DATA WITH DETAILS
CONTINUE
```

ワークロード管理ディスパッチャーの概要

DB2 ワークロード管理ディスパッチャーは、組み込みの DB2 テクノロジーで、データベース・サーバーで実行中の作業に CPU リソースを明示的に割り振ることができます。 CPU リソース割り振りは、DB2 サービス・クラスの CPU シェアと CPU リミットの属性を使用して制御できます。

概要

システムの CPU リソースに負担を与えるワークロード条件で特に役立つものとして、ディスパッチャーは、上限のない (ほとんど制限がない) ソフト CPU シェアと、上限のある (制限のある) ハード CPU シェアおよび CPU リミットといった属性を使用して、サービス・クラスの CPU リソース割り振りを効率的に管理できます。上限がないソフト CPU シェアを使用して、優先順位の高い作業のサービス・クラスに、未使用の CPU リソースを与えることができます。また、優先順位の低い作業のサービス・クラスに割り当てられた、上限があるハード CPU シェアまたは CPU リミットを使用することによって、CPU リソース割り振りの制御を強制でき、それにより優先順位の高い作業への影響を抑えることができます。システムの CPU リソースへの負担が少ない条件下では、上限がない属性を割り当てることなく、上限がある属性の 1 つまたは両方を使用して、標準的な、アンダーランの CPU 環境で実行されているサービス・クラス間で、CPU リソース割り振りを効率的に制御できます。

ワークロード管理ディスパッチャーのインフラストラクチャーは、DB2 データベース・マネージャー内で、インスタンス・レベルで動作します。ディスパッチャーは、オペレーティング・システム (OS) にディスパッチされている実行中のエージェントの数、および各エージェントが実行を許可される期間を任意の時点で制限することによって動作します。同時にディスパッチできる実行中のエージェントの数は、ディスパッチ並行性レベルと呼ばれます。

DB2 ワークロード管理ディスパッチャーには、以下のメリットがあります。

- AIX WLM や Linux WLM などの OS WLM の実装と比較して、実装が容易で必要な時間と労力が少なくて済みます。
- 通常の毎日のシステム使用の負荷の高低を通して、柔軟な CPU 割り振りをサポートします。この柔軟性は、常時強制される永続的な割り振り (ハード CPU シェアと CPU リミット)、または要求がキャパシティーを超過したときにのみ強制される動的割り振り (ソフト CPU シェア) の両方をプロビジョニングすることによって実現されます。
- DB2 データベース・マネージャー内で自己完結しているため、CPU リソース割り振りの設定によって、AIX WLM または Linux WLM などの OS WLM から独立していることにより、すべてのプラットフォームに効果があるワークロードの制御が提供されます。
- ワークロードの制御メカニズムとして、OS WLM 製品の使用を継続できますが、実装の余分な複雑さ (例えば、各メンバーで AIX WLM を設定するなど) や、組織の意見の相違 (例えば、システム管理者に OS WLM を実装したり使用

を許可したりする意思がないなど) が妨げとなっている場合は、必須ではありません。その代わりに、OS WLM 製品をモニター目的で使用する一方、DB2 ワークロード管理ディスパッチャーをワークロードの制御のために利用することができます。

その他の情報

このセクションは、次のエリアの重要な管理に関する概念、タスク、および使用法のシナリオに関する情報を提供します。

- DB2 ワークロード管理ディスパッチャー
- ハード CPU シェア
- ソフト CPU シェア
- CPU リミット
- サービス・クラスがアクティブと見なされるための CPU 使用率の最小レベル
- ディスパッチ並行性レベル
- ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング

ワークロード管理ディスパッチャー

ワークロード管理ディスパッチャーは、DB2 データベース・マネージャー内で実行されているワークロードのための CPU リソース割り振りを管理します。ここでは、フィーチャーと機能についての詳細について説明します。

概要

組み込みの DB2 ワークロード管理ディスパッチャーを使用すると、サード・パーティーのワークロード管理ソフトウェアを使用しなくても、CPU リソース割り振りを構成することによって、DB2 ワークロードを管理できます。ディスパッチャーは、CPU シェア・ベースの設定および CPU リミットの設定を使用して、DB2 固有の CPU リソース使用量を制御できます。ソフト CPU シェアは上限がないリソース割り振りを表しており、未使用の CPU リソースが存在する場合に、割り振られているシェアより多くのシェアを優先順位の高い作業が消費できるようにします。ハード CPU シェアと CPU リミットは上限がある CPU 割り振りを表しており、優先順位の低い作業が優先順位の高い作業の実行に影響しないようにするために使用可能です。CPU リミットが、特定のワークロードによって消費される CPU を厳密に制御して、システムで実行される他の作業と分離するために使用できるのに対し、ハード CPU シェアは、優先順位の高いワークロードが存在しない場合に、優先順位の低い作業が未使用の CPU リソースを消費できるようにさらに柔軟性を提供します。このシナリオは、オフピークの営業時間に発生する可能性があります。

以下のセクションでは、ワークロード管理ディスパッチャーのフィーチャーと機能を、より詳細に説明します。シナリオのセクションでは、ワークロード管理ディスパッチャーのフィーチャーと機能、およびその使用例を説明します。

フィーチャーおよび機能

ワークロード管理ディスパッチャーのインフラストラクチャーは、DB2 データベース・マネージャー内で、インスタンス・レベルで作動します。ただし、データベー

ス・レベルでは、CREATE SERVICE CLASS ステートメントおよび ALTER SERVICE CLASS ステートメントを使用して、サービス・クラスに配置できる CPU シェアと CPU リミット属性を使用し、CPU リソース割り振りを指定します。

これまでに説明したように、ハード CPU シェアとソフト CPU シェアの属性では、サービス・クラス、または複数のサービス・クラスによって解放された未使用の CPU リソースを、残りの競合するサービス・クラスが要求できる条件が異なります。ただし、ハード CPU シェアとソフト CPU シェアの属性は両方とも、これらの要求された CPU リソースを与えるときは、同じ動作をします。例えば、サービス・クラスが CPU リソースの完全な共有の一部またはすべてを解放する際、未使用の CPU リソースを要求したそれらのサービス・クラスは、完全な共有を取得しようとしているサービス・クラスが CPU リソースへの要求を増やし始めた時点でそのサービス・クラスに CPU リソースを返す必要があります。これにより、要求が存在する場合に、取得しようとしているサービス・クラスへのユーザー指定の CPU リソース割り振りの完全なリストアが発生します。 DB2 データベース・マネージャーのスケジューラー・スレッドは、毎秒、処理中の作業による CPU リソースの使用率を評価し、時間の経過と共に、ディスパッチャーが構成された CPU シェアと CPU リミットの割り振りを必ず提供しているよう調整します。

ワークロード管理ディスパッチャーは、オペレーティング・システム (OS) に同時にディスパッチされている実行中のエージェントの数と、各エージェントが実行を許可される期間の長さの両方を制限することによって作動します。同時にディスパッチできる実行中のエージェントの数は、ディスパッチ並行性レベルと呼ばれ、データベース・マネージャーの構成パラメーター **wlm_disp_concur** を使用して設定できます。

デフォルトでは、ワークロード管理ディスパッチャーは使用不可になっています。DB2 固有の CPU リソース割り振りを制御するには、最初にワークロード管理ディスパッチャーを使用可能にしておく必要があります。ワークロード管理ディスパッチャーを使用可能にするには、データベース・マネージャーの構成パラメーター **wlm_dispatcher** を YES に設定する必要があります (デフォルトでは、この構成パラメーターは NO に設定されています)。ワークロード管理ディスパッチャーを使用可能にする方法の完全な詳細については、212 ページの『ワークロード管理ディスパッチャーを使用可能にする』を参照してください。

ワークロード管理ディスパッチャーが使用可能な場合、DB2 データベース・マネージャー内のユーザー・サービス・クラスおよび保守サービス・クラスで実行されているすべての作業は、ディスパッチャーの制御下にあります。システム・サービス・クラスで実行されている作業は、CPU リソースの制御に対して構成することはできません。これは、このサービス・クラスで実行されている重要な DB2 サブシステムは、最高の優先順位を与えられており、ワークロード管理ディスパッチャーの制御を受けないためです。デフォルトで、ディスパッチャーをオンにすると、ディスパッチャーは CPU リミットの設定によってのみ CPU リソースを管理できます。ディスパッチャーが CPU シェアと CPU リミットの両方を使用して CPU リソースを管理できるようにするには、データベース・マネージャーの構成パラメーター **wlm_disp_cpu_shares** の値を YES に設定する必要があります。

ワークロード管理ディスパッチャーを初めて使用可能にする前後に、ワークロードをモニターして、ワークロードが消費する相対的な CPU リソースを判別すること

ができます。ワークロードの相対的な優先順位に基づいて、この情報は、作業を割り当てることができるサービス・クラスの作成、特定のサービス・クラスに割り当てる CPU シェアのタイプ (ハードまたはソフト)、各サービス・クラスに割り当てる CPU シェアの相対的な量、および CPU リミットを使用するかどうかに関して決定する上で役に立ちます。

DB2 データベース・マネージャーの動作を制御する最大限の柔軟性を提供する別の考慮事項として、アクティブと見なされるサービス・クラスの CPU リソースの使用率の最小パーセント値を設定するというオプションがあります。データベース・マネージャーの構成パラメーター **wlm_disp_min_util** を設定した後で、最小パーセント値以上の CPU リソースを使用しているサービス・クラスは、ホストまたはロジカル・パーティション (LPAR) 上でアクティブであると見なされ、アクティブなサービス・クラスの CPU シェアで、CPU リソース割り振りが計算されます。

ワークロード管理ディスパッチャーによって、システム上のサービス・クラス間の CPU リソースの割り振りを、サービス・クラスの CPU シェアの属性を経由して制御できます。共有の属性は、各サービス・クラスがシステム上の他のサービス・クラスと比較して受け取る CPU 時間の相対的な割り振り量を表します。あるサービス・クラスにより多くの CPU シェアを割り振り、別のサービス・クラスに少なく割り振ることによって、各サービス・クラスに割り振られる CPU リソースの量を制御でき、あるサービス・クラスに別のサービス・クラスよりも優れたサービス品質を提供することができます。

以前に使用可能にされたワークロード管理ディスパッチャーが CPU シェアと CPU リミットの両方を使用して、負担の大きい CPU リソースを最も適切に管理できるという決定を行った後で、データベース・マネージャーの構成パラメーター **wlm_disp_cpu_shares** を YES に設定して、CPU シェアを使用可能にする必要があります。このパラメーターのデフォルトの設定値は NO です。CREATE SERVICE CLASS ステートメントおよび ALTER SERVICE CLASS ステートメントを使用することによって、CPU シェアと CPU リミットを設定して調整できます。

サービス・クラスに割り当てられている CPU シェアの数に基づいて、ワークロード管理は、各サービス・クラスが使用を許可されている CPU リソースの比率を計算します。各サービス・スーパークラスが許可されている CPU リソースの比率を判別するために、以下の公式を使用して、特定のサービス・スーパークラスの CPU シェアの数を、ワークロード管理ディスパッチャーによって割り振られている CPU リソースのパーセンテージに変換できます。

```
% CPU(スーパークラス) = (スーパークラス共有の数 /
 アクティブ・スーパークラスすべての共有総数) x 100
```

各サービス・サブクラスが許可されている CPU リソースの比率を判別するには、以下の公式を使用して、特定のサービス・サブクラスの CPU シェアの数を、ワークロード管理ディスパッチャーによって割り振られている CPU リソースのパーセンテージに変換できます。

```
% CPU(サブクラス) = % CPU(スーパークラス) x (サブクラス共有の数 /
 スーパークラス内のアクティブ・サブクラスすべての共有総数)
```

注: すべてのアクティブなスーパークラスの (ハードとソフトの両方の) CPU シェアの総数は、ホストまたは LPAR 上のすべてのデータベースおよびすべてのメンバー間でカウントされます。

CPU の比率の計算の図の例は、『使用法のシナリオ』のセクションを参照してください。

CPU リソースのスケジューリング機能の要約

以下の表に、ワークロード管理ディスパッチャーによって管理される DB2 サービス・クラスで利用可能な、さまざまな CPU リソースの制御属性のフィーチャーを要約します。

表 44. ワークロード管理ディスパッチャーの CPU リソースのスケジューリング機能の要約

CPU リソースの制御属性	同じコンテキスト内のアイドル状態の CPU リソースの使用率	CPU リソース割り振りの制限が同じコンテキストの他のアクティブなサービス・クラスと相対的である	CPU リソース割り振りの制限がシステム全体の CPU の容量に基づき固定されている	CPU リソースの高い使用率での CPU 消費の制御	CPU リソースの低い使用率での CPU 消費の制御	使用上の注意
ソフト共有	Y	Y	N	Y	N	ワークロードの進行を最大化するために、優先順位の高い作業が、ホストまたは LPAR で利用可能な予備の CPU リソースを利用するのに最適
ハード共有	N	Y	N	Y	Y	さまざまな量と強度でホストまたは LPAR で実行されている可能性がある優先順位の高い作業を、別の作業が干渉しないようにするために最適

表 44. ワークロード管理ディスパッチャーの CPU リソースのスケジューリング機能の要約 (続き)

CPU リソースの制御属性	同じコンテキスト内のアイドル状態の CPU リソースの使用率	CPU リソース割り振りの制限が同じコンテキストの他のアクティブなサービス・クラスと相対的である	CPU リソース割り振りの制限がシステム全体の CPU の容量に基づき固定されている	CPU リソースの高い使用率での CPU 消費の制御	CPU リソースの低い使用率での CPU 消費の制御	使用上の注意
CPU リミット	N	N	Y	Y	Y	ホストや LPAR の他の作業に関係なく、サービス・クラス間の一貫性のある固定された CPU リソース割り振りのサンドボックス化と強制、またはサービス・クラスの CPU リソースの消費を制限するのに最適

ワークロード管理ディスパッチャーの制御の外部の作業の影響

ワークロード管理ディスパッチャーは、各 DB2 インスタンスに対して存在します。それ自体のインスタンス内でのみ、DB2 の作業を直接検出して管理できます。同じホストまたは LPAR で実行されている他の作業は、ディスパッチャーの直接制御下にありません。 DB2 データベース・マネージャーの外部で実行される作業の CPU リソースの要求がランダムであるということは、次のことを意味します。DB2 内で実行されている作業に対しては CPU リミットと CPU リソースの相対シェアが任意の時点において実施されているとはいえ、ディスパッチャーの管理する作業は、ディスパッチャーの直接制御外であり、かつ DB2 インスタンスの外部で実行される作業と競合します。ワークロード管理ディスパッチャーによって検出も管理もされない作業には、以下のものが含まれます。

- DB2 データベース・マネージャーの外部で作業の一部を実行するアプリケーションまたはミドルウェア製品によって実行される作業
- DB2 システムのサービス・クラスで実行されるエンティティーによって実行される作業
- 他の DB2 インスタンスによって実行される作業
- fenced ストアード・プロシージャーなど、fenced モード・プロセスで実行される DB2 以外の作業
- トラステッド・ルーチンで実行される DB2 以外の作業。トラステッド・プロシージャーおよびユーザー定義関数 (UDF) は、ディスパッチャーの CPU リソースのスケジューリングの原因となりますが、トラステッド・ルーチンが長時間ユー

ザー・コード内で実行された場合、ディスパッチャーは CPU 使用率を制限できません。

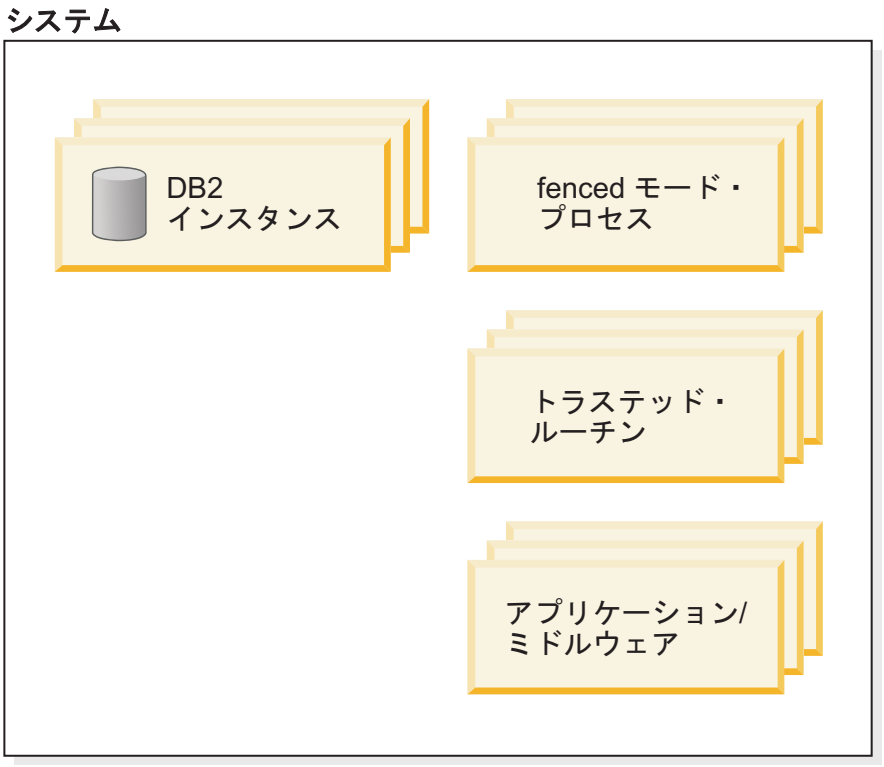


図 22. システム図の例

DB2 データベース・マネージャーのみを実行しているホストまたは LPAR がある場合を考えてみましょう。サービス・クラス A と B のあるデータベースを作成し、CPU リソースの 50% のシェアを各サービス・クラスに割り当てます。ホストまたは LPAR 全体が常に CPU リソースを完全に使用していると仮定すると、一定期間の CPU 使用率 (%) の合計の測定は、 196 ページの図 23 パネル A で示したものと似ています。

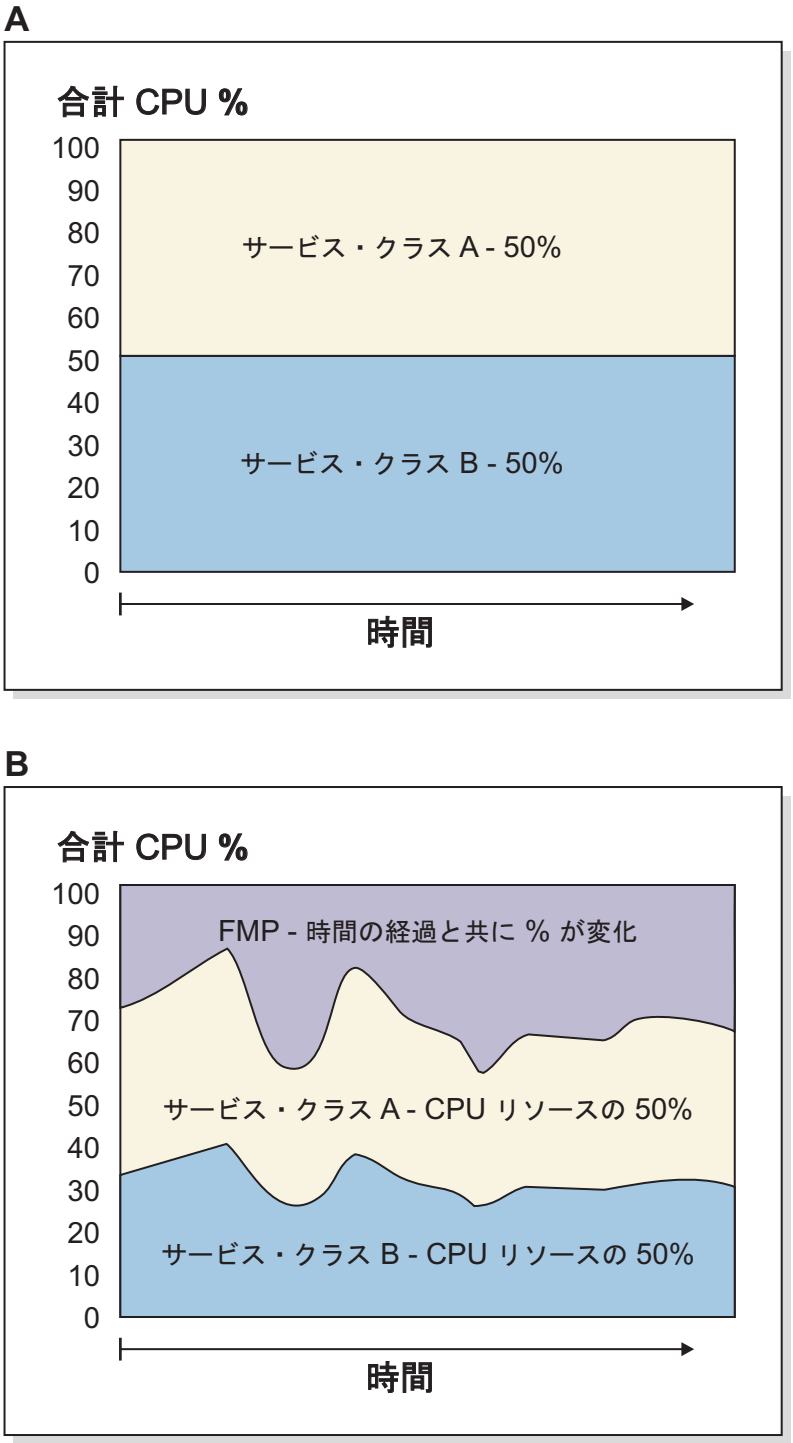


図 23. CPU 使用率の合計 (%)

では、ホストまたは LPAR で fenced モード・プロセス (FMP) が実行されている時点で何が発生しているかを考慮してみましょう。これらのプロセスは、ワークロード管理ディスパッチャーの制御下にないため、使用する CPU リソースの量はランダムに変化します。ただし、パネル B に示すように、ディスパッチャーが管理できる残りの CPU リソースから、サービス・クラス A と B はそれぞれ、残りの CPU リソースを等しく 50% に分割したものを継続して取得します。

ディスパッチャーと他のサービス・クラスの設定の間の相互作用

以下の表は、サービス・クラス (ハード CPU シェアとソフト CPU シェア、および CPU リミット) のワークロード管理ディスパッチャーの設定が、どのように同じサービス・クラスの他の設定に相互作用するかを要約したものです。

表 45. ディスパッチャーと他のサービス・クラスの設定の間の相互作用

ディスパッチャーのサービス・クラスの設定	他のサービス・クラスの設定	相互作用
ハード CPU シェアとソフト CPU シェアおよび CPU リミット	エージェントの優先順位 **注:** 各 DB2 サービス・クラスを、サービス・クラス内のエージェントのオペレーティング・システムの相対的な優先順位を制御するエージェント優先順位に関連付けることは推奨されていません。また、将来のリリースでは除去される可能性があります。CPU 使用量を制御する場合は、エージェントの優先順位の代わりに、ワークロード管理ディスパッチャーを使用してください。	データベース・マネージャーの構成パラメーター **wlm_dispatcher** が YES に設定されている場合、エージェントの優先順位は無視され、CPU リミットがアクティブ化されます。**wlm_dispatcher** パラメーターが NO に設定されている場合、CPU リミットは無視され、エージェントの優先順位をオーバーライドしなくなります。
ハード CPU シェアとソフト CPU シェアおよび CPU リミット	アウトバウンド相関関係子 (OS WLM 統合)	ハード CPU シェアとソフト CPU シェア、CPU リミット、および OS WLM をすべて同時にアクティブにすることができます。
ハード CPU シェアとソフト CPU シェアおよび CPU リミット	バッファー・プール優先順位	ハード CPU シェアとソフト CPU シェア、CPU リミット、およびバッファー・プールの優先順位をすべて同時にアクティブにすることができます。
ハード CPU シェアとソフト CPU シェアおよび CPU リミット	プリフェッチ優先順位	ハード CPU シェアとソフト CPU シェア、CPU リミット、およびプリフェッチの優先順位をすべて同時にアクティブにすることができます。

ディスパッチャーの設定と REMAP アクションを使用したしきい値の間の相互作用

一部のワークロード管理しきい値は、実行中の作業を、あるサービス・サブクラスから、同じサービス・スーパークラス内の別のサービス・サブクラスに移動するための REMAP ACTIVITY アクションをサポートします。作業が移動元のサービス・サブクラスとは異なる CPU シェアまたは CPU リミットが指定されたサービス・サブクラスに移動する際、作業の移動先のサービス・サブクラスの CPU シェアと CPU リミットの設定が適用されます。

ワークロード管理ディスパッチャーとワークロード管理並行性しきい値の間の比較

ワークロードのスケジューリングおよびリソースの消費量の制御は、ワークロード管理ディスパッチャーと並行性しきい値のどちらを使用しても行えますが、それぞれが行うワークロードの制御の程度の違いと利点を認識しておくと、ご自分の環境でどんなときにどちら (または両方) を適用すればよいのか決定するのに役立ちます。以下の表では、CPU リソースの制御機能を比較しています。

表 46. ワークロード管理ディスパッチャーとワークロード管理並行性しきい値との間の CPU リソースの制御機能の比較

ワークロード管理ディスパッチャー	DB2 ワークロード管理並行性しきい値
アクティビティーが CPU リソースの消費を開始した後で、DB2 データベース・マネージャー内の CPU の割り振りを直接制御できます。	同時に実行を許可されている接続またはアクティビティーの数を制限することによって、CPU リソースと他のリソース (メモリーなど) の使用量の両方を間接制御できます。
CPU リソースのソフト共有とハード共有のサービス・クラスへの相対的な割り振り、および単一サービス・クラスによる CPU 消費の絶対的な制限を構成できます。	アクティビティーの実行の並行性や、サービス・クラスに対して開かれた接続に絶対的な制限を課すことができますが、相対的な CPU 割り振りはどのようなタイプでも指定することはできません。

ワークロード管理ディスパッチャーと OS WLM (AIX WLM および Linux WLM) の間の比較

ワークロード管理ディスパッチャーは、ある特定の状況でより大きな制御を提供するために、OS WLM (AIX WLM または Linux WLM) と共に使用することができます。次の例に、ワークロード管理ディスパッチャーと OS WLM の両方が一緒に動作できる状況を示します。

- OS WLM はしばしば、ワークロード管理ディスパッチャーのモニター機能を補完する、OS レベルのモニターを持っていることがあります。
- OS WLM は、DBMS を含むシステムのすべてのアプリケーションを管理できます。ワークロード管理ディスパッチャーは、DB2 固有のディスパッチングを提供できる一方、OS WLM は、システムのその他すべてのアプリケーションを管理します。
- OS WLM とワークロード管理ディスパッチャーの両方を同時に使用して、個々のサービス・クラスへの CPU リソース割り振りを制御しようとしないでください。ワークロード管理ディスパッチャーが使用可能になっているときに、モニターの目的で OS WLM をサービス・クラス・レベルで統合できます。また、外部のアプリケーション、または DB2 インスタンスに割り振られている全体の CPU リソース、あるいはその両方を制御するために、OS WLM を統合することもできます。

表 47. ワークロード管理ディスパッチャーと OS WLM の間の機能の比較

ワークロード管理ディスパッチャー	OS WLM
すべてのプラットフォーム間で動作する単一のソリューション。	OS WLM は、各 OS に固有です。それぞれ独自のユーザー・インターフェースと独自の制限があります。

表 47. ワークロード管理ディスパッチャーと OS WLM の間の機能の比較 (続き)

ワークロード管理ディスパッチャー	OS WLM
OS 特有の統合や設定を必要とせずに、DB2 データベース・マネージャー内の CPU 割り振りを制御できます。DBA は、システムへのルート・アクセスを必要とせずに、これらの設定を管理できます。	OS WLM は構成のためにルート特権を必要とすることがあります。
CPU リソースのスケジューリングを決定する際に、OS レベルの機能では不可能な、データベース固有の知識を活用できます。	OS WLM は、DB2 データベース・マネージャーを別のアプリケーションとして取り扱うため、CPU リソースのスケジューリングを決定するときに、データベース固有の知識を利用できません。
複数メンバー環境では、すべてのメンバーを 1 つのコマンドで構成できます。ただし、その結果、各メンバーで別の構成を持つことができません。	複数メンバー環境では、OS WLM は各メンバーで構成する必要があります。各メンバーで別の構成を持つことができます。
単一の DB2 インスタンスの有効範囲内のユーザー・サービス・クラスおよび保守サービス・クラスで実行されている、DB2 の作業のみによる CPU リソースの使用を制御します。ディスパッチャーは、DB2 インスタンスの外部で実行されているアプリケーションによる CPU リソースの使用を制御できません。	OS WLM はシステム全体の CPU リソースを完全に制御できます。
CPU リソースの制御は、ソフト CPU シェアとハード CPU シェアの両方によって達成されます。	AIX および Linux 上の OS WLM ソリューションは、現時点では、DB2 のハード CPU シェアに相当するものを提供していません。

ワークロード管理ディスパッチャーとエージェントの優先順位の間の比較

エージェントの優先順位を使用したワークロードの制御を使用できます。以下の表では、CPU リソースを制御するエージェントの優先順位の機能と、ワークロード管理ディスパッチャーの機能を比較します。

表 48. ワークロード管理ディスパッチャーとエージェントの優先順位の間の CPU リソースの制御機能の比較

ワークロード管理ディスパッチャー	エージェントの優先順位
サービス・クラス間での CPU リソースの使用について、一貫して正確に制限を割り振る、または強制する方法。	エージェントの優先順位は、サービス・クラスによる CPU リソースの使用に制限を強制することはできません。エージェントの優先順位により、CPU リソース割り振りに関して詳細に制御することなく、サービス・クラス相互の相対優先順位のみを指定できます。

注: 各 DB2 サービス・クラスを、サービス・クラス内のエージェントのオペレーティング・システムの相対的な優先順位を制御するエージェント優先順位に関連付けることは推奨されていません。また、将来のリリースでは除去される可能性があり

ます。CPU 使用量を制御する場合は、エージェントの優先順位の代わりに、ワークロード管理ディスパッチャーを使用してください。

使用法のシナリオ

シナリオ 1: CPU 割り当ての計算

図 24に、図のように構成されたデータ・サーバーを示します。基本的な概念を説明する図では、デフォルトのユーザー、保守、およびシステムのサービス・クラスで実行されている無視できる作業があると仮定します。シナリオの最初の状態として、ユーザー・サービス・スーパークラス A、B、および C で実行されており、それらのサービス・クラスに対して CPU リソース割り振り全体を使用するための作業が十分あるとします。以下のシナリオでは、1 つのデータベースを持つ 1 つの DB2 インスタンスしかなく、このホストまたは LPAR に 1 つのロジカル・メンバーしかありません。

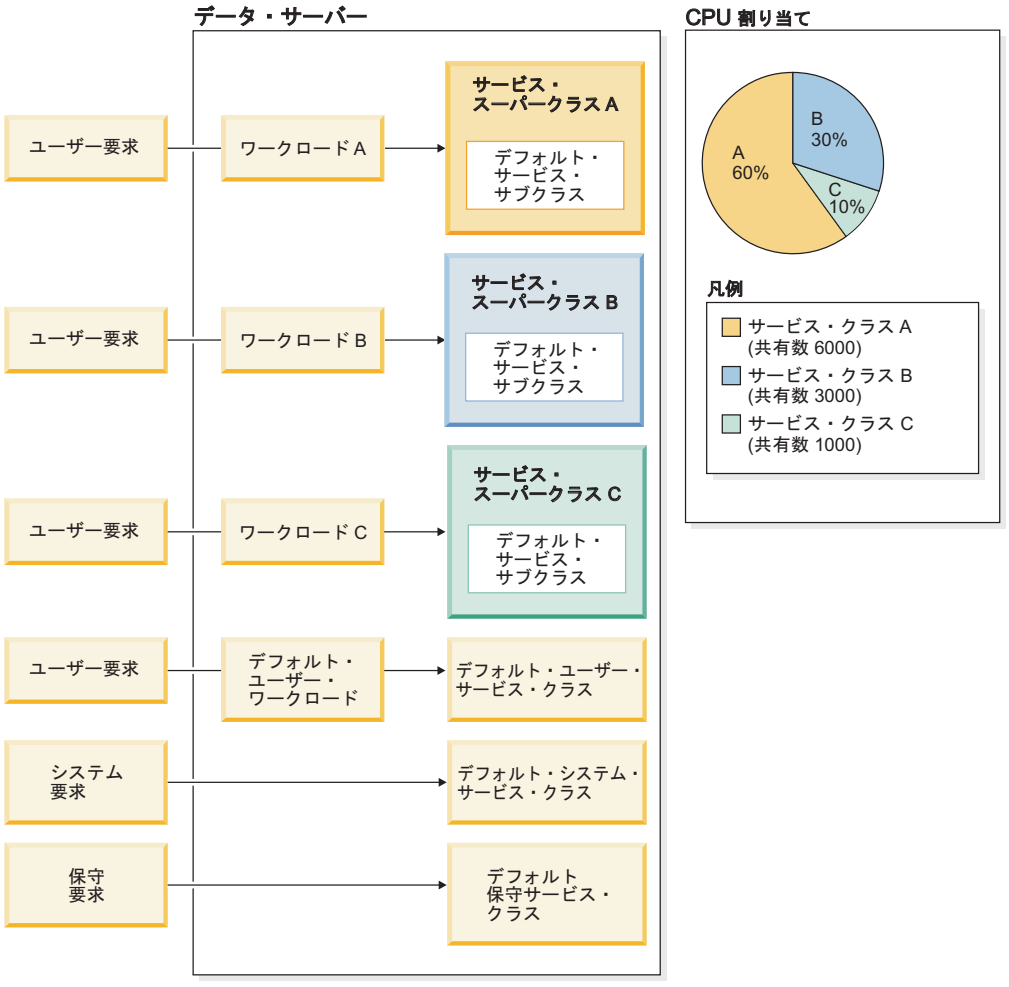


図 24. データ・サーバーの構成

図 24 に示すように、各サービス・スーパークラスの CPU 割り当て率は、以下のように計算されます。

- サービス・スーパークラス A は 60% 持っています

(6000 / (6000 + 3000 + 1000)) x 100

- サービス・スーパークラス B は 30% 持っています

  (3000 / (6000 + 3000 + 1000)) x 100

- サービス・スーパークラス C は 10% 持っています

  (1000 / (6000 + 3000 + 1000)) x 100

CPU シェアは、サービス・スーパークラスとサービス・サブクラスの両方のレベルで割り当てることができます。サービス・スーパークラスまたはサービス・サブクラスを作成する際に CPU シェアを指定しない場合、DB2 データベース・マネージャーは、デフォルトの 1000 のハード CPU シェアをそのようなサービス・クラスに割り当てます。スーパークラス・レベルで、CPU シェアの値は、システムの CPU リソースが DB2 サービス・スーパークラス間でどのように分割されるかを表します。サブクラス・レベルで、CPU シェアの値は、アクティブなサブクラス間の特定のスーパークラスが利用可能な CPU リソースの分割を表します。

シナリオ 2: ワークロード管理ディスパッチャーがインスタンス・レベルで作動し、サービス・クラスはデータベース・レベルで作動する

ワークロード管理ディスパッチャーはインスタンス・レベルで作動しますが、CPU リソースを割り振るためにディスパッチャーが使用する構成 (CPU シェアと CPU リミット) は、個々のデータベース内の個々のサービス・スーパークラスおよびサービス・サブクラスに配置されています。複数のデータベースが同じインスタンス内に存在する場合、各データベース上のサービス・スーパークラス内の CPU シェアの値の合計は、各データベースが受け取る CPU リソースの相対的な割り振りを決定します。例えば、同じインスタンスに、データベース A とデータベース B の 2 つのデータベースが存在する場合、 202 ページの図 25パネル A に示すように、各データベースには 2 つのサービス・スーパークラス (SC1 と SC2) があります。

図 25. 1 つのインスタンス内の 2 つのデータベースに対するサービス・クラスへの CPU シェアの割り振り

このサンプル・シナリオでは、パネル B に示すように、SC1 と SC2 にそれぞれ 7000 と 3000 のソフト CPU シェアを使用して構成されているデータベース A の CPU リソースを 70/30 の割合で分割します。データベース B では、パネル C に示すように、SC1 と SC2 にそれぞれ CPU リソースを 60/40 の割合で分割します。

この 60/40 の割合の分割が、データベース B の 2 つのサービス・スーパークラスの CPU シェアにどのように変換されるかを決定する前に、データベース A と B の間でどのような相対 CPU リソース割り当てが必要かを決定する必要があります。データベース B で実行されている作業が、データベース A で実行されている作業よりずっと重要です。このため、データベース B の CPU 割り当てが 80%、データベース A が 20% となるようにします。

データベース A の CPU シェアの合計は、10,000 です。これらの 10,000 の CPU シェアは、2 つのデータベース間の CPU シェアの合計の 20% を表し、CPU リソースの合計の 100% を表すには、合計で 50,000 の CPU シェアが必要です。このため、データベース B は、50,000 の CPU シェアの 80% が割り当てられ、結果として 40,000 の CPU シェアになる必要があります。データベース B の 2 つのスーパークラス間で 60/40 の割合の分割を得るには、サービス・スーパークラス SC1 が、データベース B に割り当てられている合計 40,000 の CPU シェアの 60% を受け取り、結果として 24,000 の CPU シェアになります。サービス・スーパークラス SC2 は、データベース B に割り当てられている合計 40,000 の CPU シェアの 40% を受け取り、結果として 16,000 の CPU シェアになります。

円グラフは、必要な割合の CPU リソース割り振りを達成した CPU シェアの割り当てを示しています。

シナリオ 3: 高い CPU 使用率であるワークロードを別のワークロードより優遇し、優遇されたワークロードを低い CPU 使用率で保護する

このサンプルの使用法のシナリオでは、複数のワークロードのサービス品質を管理しながら、同時にシステムの CPU 使用率を最大化することが目的です。ただし、このシナリオのワークロードでは、CPU リソースの競合が発生するだけでなく、他のリソースの競合も発生します。競合するワークロードが、CPU 割り当て全体を消費せずに、まだ存在している場合は、未使用の CPU リソースを一部のワークロードに割り振ることで、CPU キャッシュの競合、I/O 競合、またはその他の 2 次的な影響によるパフォーマンスの低下が発生する可能性があります。

このシナリオでは、CPU リソース全体が使用されていない可能性があるものの、優遇されたワークロードを優遇されない (優先順位を下げられている) ワークロードから保護する必要があります。優遇されない (優先順位を下げられている) ワークロードが制限され、優遇されたワークロードがアクティブであるが、CPU 割り当て全体を使用していない場合に、使用率の低い CPU リソースが発生する可能性があります。ハード CPU シェアを優遇されない (優先順位を下げられた) ワークロードに割り当て、ソフト CPU シェアを優遇されたワークロードに割り当てることによって、優遇されたワークロードの保護を実現できます。

ここでもまた、2 つのワークロード「Favored」と「Other」を使用します。「Favored」ワークロードは、「Other」ワークロードと同時に実行されている場合、CPU リソースの 70% を使用し、「Other」ワークロードが存在しない場合は 100%

使用します。「Other」ワークロードは、「Favored」ワークロードと同時に実行されている場合は CPU リソースの 30% を使用し、「Favored」ワークロードが存在しない場合は 100% 使用します。これはシナリオ 3 と似ていますが、全体の CPU 使用率が場合によっては 100% を下回ることがあるという点が異なります。

以下のステップでは、前の部分でリストされた目標を達成するための DB2 ワークロード管理の構成方法を示しています。

1. 「Favored」サービス・スーパークラスと「Other」サービス・スーパークラスを作成し、CPU 割り当て率 70/30 を満たすように CPU シェアを割り当てます。CPU シェアは、7 と 3、14 と 6 など、70%/30% という CPU 割り当て率を達成するならどんな組み合わせでも指定できます。以下の例では、7000 のソフト CPU シェアが「Favored」サービス・クラスに割り当てられ、3000 のハード CPU シェアが「Other」サービス・クラスに割り当てられます。

   ```
   create service class favored soft cpu shares 7000
   create service class other hard cpu shares 3000
   ```

2. 以下の例に示すように、「Favored」ワークロードと「Other」ワークロードを作成します。

   ```
   create workload favored appl_name('favoredapp') service class favored
   create workload other appl_name('otherapp') service class other
   ```

   結果として、システムは以下のようになります。

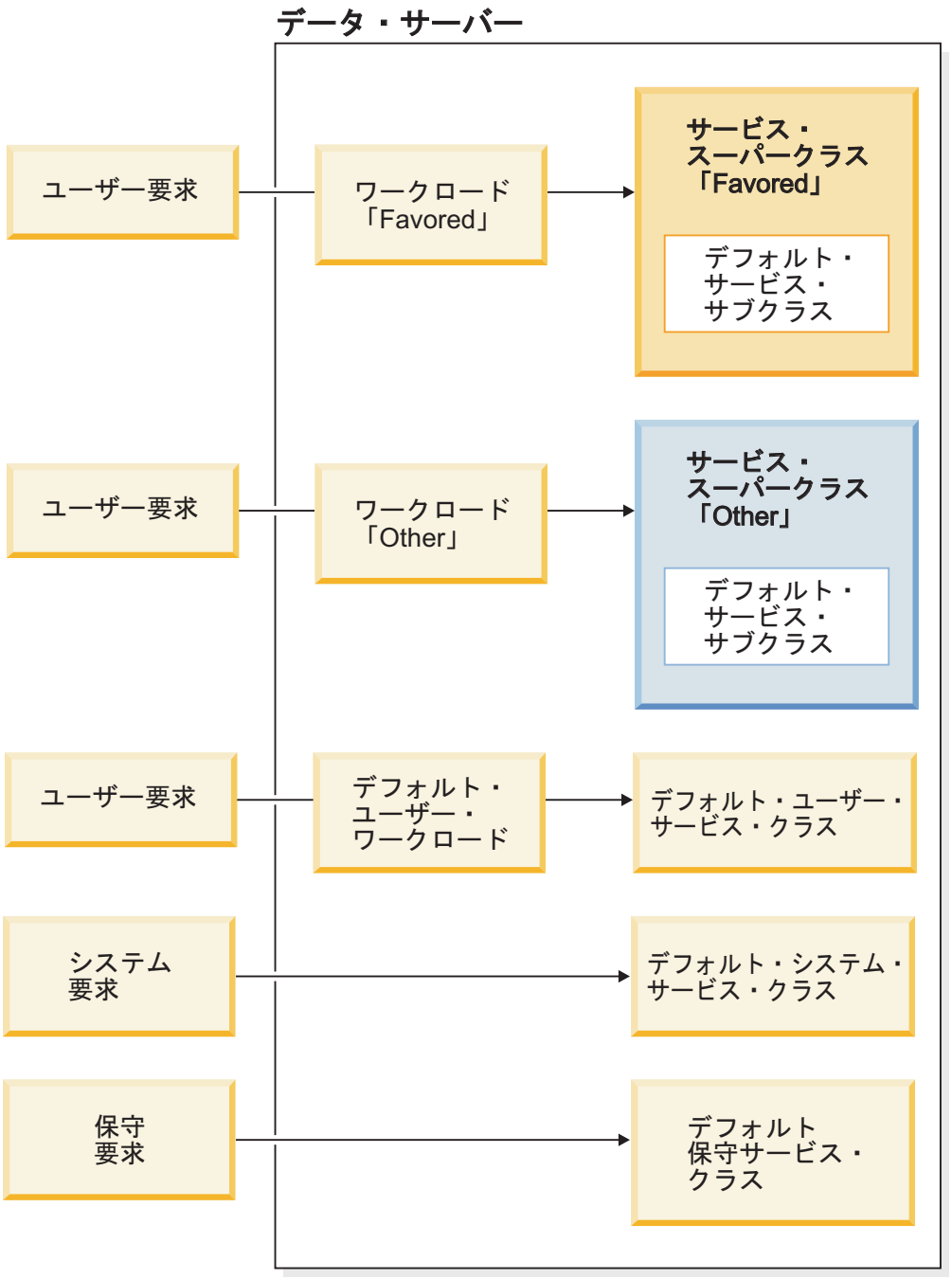


図 26. データ・サーバーの構成

単純化のために、「Favored」と「Other」以外にサービス・クラスで実行されているのは無視できる量の作業だけだとします。

3. 必要な CPU シェアの目標が達成されているかどうかを測定します。システムは、ワークロードの点から、以下のいくつかの状態の 1 つにすることができます。
   a. 100% の CPU 使用率
      1) 作業が「Favored」ワークロードと「Other」ワークロードの両方で実行されている

測定された CPU 使用率は、「Favored」サービス・クラスに対して 70%、「Other」サービス・クラスに対して 30% です。

2) 作業が「Favored」ワークロードでのみ実行されている

測定された CPU 使用率は、「Favored」サービス・クラスに対して 100%、「Other」サービス・クラスに対して 0% です。

3) 作業が「Other」ワークロードでのみ実行されている

測定された CPU 使用率は、「Favored」サービス・クラスに対して 0%、「Other」サービス・クラスに対して 100% です。

b. サービス・クラスが CPU 割り当てより少なく使用する

1) 作業が「Favored」ワークロードと「Other」ワークロードの両方で実行されている。「Favored」ワークロードは、CPU 割り当てによる 70% の全体ではなく 50% を使用しています。

「Favored」ワークロードの測定された CPU 使用率は 50% です。「Other」ワークロードは、CPU シェアがハードであるため、30% を継続して使用します。CPU 使用率の合計は 80% になります。

2) 作業が「Favored」ワークロードと「Other」ワークロードの両方で実行されている。「Other」ワークロードは、CPU 割り当てによる 30% 全体ではなく 10% を使用しています。

CPU シェアがソフトであるため、「Favored」サービス・クラスの測定された CPU 使用率が 70% から 90% に増え、未使用の 20% が「Other」サービス・クラスによって解放されることを要求します。「Other」サービス・クラスの CPU 使用率は 10% です。CPU 使用率の合計は 100% になります。

3) 作業が「Favored」ワークロードでのみ実行されており、CPU 割り当てのすべてを使用している。

「Other」サービス・クラスで実行されている作業がないため、「Favored」サービス・クラスは、それ自体の CPU 割り当ての 70% と、「Other」サービス・クラスによって解放された CPU リソースの未使用の 30% の両方を使用できます。 CPU 使用率の合計は 100% になります。

4) 作業が「Other」ワークロードでのみ実行されており、CPU 割り当てのすべてを使用している。

「Favored」サービス・クラスで実行されている作業がないため、「Other」サービス・クラスは、それ自体の CPU 割り当て 30% と、「Favored」サービス・クラスによって解放された CPU リソースの未使用の 70% の両方を使用できます。 CPU 使用率の合計は 100% になります。

シナリオ 4: 一部の CPU リソースがアイドル状態になったとしても、一貫性を実現するためにサンドボックスを作成する

このサンプルの使用例のシナリオでは、未使用の CPU リソースが利用可能かどうかに関係なく、優遇されたワークロードは、固定された比率の CPU リソースを超えて消費することはありません。

2 つのワークロード「Favored」と「Other」を使用します。「Favored」ワークロードには、未使用の CPU リソースが利用可能かどうかに関係なく、いつでも最大 70% の CPU リソースが割り振られることになります。「Other」ワークロードには、未使用の CPU リソースが利用可能かどうかに関係なく、いつでも最大 30% の CPU リソースが割り振られることになります。

以下のステップでは、前の部分でリストされた目標を達成するための DB2 ワークロード管理の構成方法を示しています。

1. 「Favored」サービス・クラスと「Other」サービス・クラスを作成し、それぞれに CPU リミットを割り当てることにより、「Favored」ワークロードと「Other」ワークロードにそれぞれ最大 70% と 30% の CPU リソースが強制されます。以下の例では、CPU リミットは、1 から 100 の間の数値として指定する必要があります。この数値は、そのサービス・クラスの CPU 使用率の最大のパーセンテージを表します。

   ```
   create service class favored cpu limit 7000
   create service class other cpu limit 3000
   ```

2. 以下の例に示すように、「Favored」ワークロードと「Other」ワークロードを作成します。

   ```
   create workload favored appl_name('favoredapp') service class favored
   create workload other appl_name('otherapp') service class other
   ```

   結果として、システムは 205 ページの図 26のようになります。

   単純化のために、「Favored」と「Other」以外にサービス・クラスで実行されているのは無視できる量の作業だけだとします。

3. 必要な CPU リミットが正常に強制されているかどうかを測定します。システムは、ワークロードの点から、以下のいくつかの状態の 1 つにすることができます。

   a. 100% の CPU 使用率

      1) 作業が「Favored」ワークロードと「Other」ワークロードの両方で実行されている

         測定された CPU 使用率は、「Favored」サービス・クラスに対して 70%、「Other」サービス・クラスに対して 30% です。

   b. 1 つ以上のサービス・クラスが CPU リミットを下回って使用している

      1) 作業が「Favored」ワークロードと「Other」ワークロードの両方で実行されている。「Favored」ワークロードは、CPU リミット割り振りによる 70% 全体ではなく 50% を使用しています。

         「Favored」ワークロードの測定された CPU 使用率は 50% です。「Other」ワークロードは、CPU リミット割り振りの最大値が 30% であるため、30% を継続して使用します。 CPU 使用率の合計は 80% になります。

2) 作業が「Favored」ワークロードと「Other」ワークロードの両方で実行されている。「Other」ワークロードは、CPU リミット割り振りによる 30% 全体ではなく 10% を使用しています。

   最大 70% の CPU 使用率を要求している CPU リミットのため、「Favored」サービス・クラスの測定された CPU 使用率は 70% のままです。「Other」サービス・クラスによって解放された CPU リソースの未使用の 20% は、要求されないままです。「Other」サービス・クラスの CPU 使用率は 10% です。CPU 使用率の合計は 80% になります。

3) 作業が「Favored」ワークロードでのみ実行されており、CPU リミットによって要求された CPU リソースの最大量を使用している。

   最大 70% の CPU 使用率を要求している CPU リミットのため、「Favored」サービス・クラスの測定された CPU 使用率は 70% のままです。「Other」サービス・クラスによって解放された CPU リソースの未使用の 30% は、要求されないままです。「Other」サービス・クラスの CPU 使用率は 0% です。CPU 使用率の合計は 70% になります。

4) 作業が「Other」ワークロードでのみ実行されており、CPU リミットによって要求された CPU リソースの最大量を使用している。

   最大 30% の CPU 使用率を要求している CPU リミットのため、「Other」サービス・クラスの測定された CPU 使用率は 30% のままです。「Favored」サービス・クラスによって解放された CPU リソースの未使用の 70% は、要求されないままです。「Favored」サービス・クラスの CPU 使用率は 0% です。CPU 使用率の合計は 30% になります。

シナリオ 5: ユーザーによってワークロードを分割し、特定のユーザーに対してあるタイプの作業を別の作業より優遇する必要がある

これまでのシナリオでは、サービス・クラスは、業務の優先順位に基づき、異なるワークロード間での CPU リソースの割り振りの制御に使用されていました。場合によっては、業務の優先順位に基づき、ワークロード間でリソースの共有を制御し、全体的なスループットや応答時間を向上するために、長時間実行する作業より短い時間実行する作業を優先する必要があることがあります。

このサンプルの使用法のシナリオでは、CEO およびその他の 1000 名の従業員のすべてが、mybizapp という 1 つのアプリケーションを使用してデータベースにアクセスします。ここでの目的は、CEO 1 人に CPU リソースの完全な 10% の共有を与え、残りの 90% の CPU リソースが、データベースにアクセスする他の 1000 名のユーザーによって使用されることです。この目標を達成するために、CEO に ceo_sc というサービス・スーパークラスを作成します。次に、以下の例に示すように、CEO を自分のサービス・スーパークラスにマップするワークロードを定義します。

```
create service class ceo_sc soft cpu shares 1000
create workload ceo_wl
  applname('mybizapp')
  session_user('ceo')
  service class ceo_sc
```

他のすべてのユーザーのために、サービス・スーパークラス mybizapp_sc を作成します。次に、以下の例に示すように、その他すべての mybizapp アプリケーション

のユーザーを、新しく作成したサービス・スーパークラス mybizapp_sc にマップするようワークロード (mybizapp_wl) を定義します。

```
create service class mybizapp_sc hard cpu shares 9000
create workload mybizapp_wl
  applname('mybizapp')
  service class mybizapp_sc
```

mybizapp アプリケーションのワークロード内に、コストの高い照会とコストの低い照会の両方があります。業務組織では、一般に、コスト計算の結果が 10,000 timeron 未満の低コスト照会のほうが、コスト計算の結果が 10,000 timeron を超える高コスト照会より優先順位が高いと判断しています。このため、コストが低い照会には、コストが高い照会と比べて、CPU リソースの共有が 2:1 の比率で大きく与えられることになります。この目標を達成するため、以下の例に示すように、mybizapp_sc サービス・スーパークラス内に 2 つのサービス・サブクラス (lowcost_ssc と highcost_ssc) を作成し、ソフト CPU シェアを 2:1 の比率で割り当てます。

```
create service class lowcost_ssc under mybizapp_sc soft cpu shares 2000
create service class highcost_ssc under mybizapp_sc soft cpu shares 1000
```

比率が 2:1 のソフト CPU シェアは、mybizapp_sc サービス・スーパークラスに割り当てられた 90% の CPU リソースのうちのどれだけがコストの低い照会に割り振られ、どれだけがコストの高い照会に割り振られるかを決定します。

以下の例に示すように、作業クラス・セット (splitbycost_wcs) および作業アクション・セット (mybizapp_was) を作成して、コストの低い照会を lowcost_ssc サービス・サブクラスに経路指定し、コストの高い照会を highcost_ssc サービス・サブクラスに経路指定します。

```
create work class set splitbycost_wcs
 (work class lowcost_wc work type dml for timeroncost from 0 to 10000,
 work class highcost_wc work type dml for timeroncost from 10001)

create work action set mybizapp_was for service class mybizapp_sc
 using work class set splitbycost_wcs
 (work action maplowcost_wa on work class lowcost_wc map activity
 to lowcost_ssc, work action maphighcost_wa on work class highcost_wc
 map activity to highcost_ssc)
```

各サービス・クラスで作業が実行されており、CPU リソース割り振り全体を消費している場合、上記の構成によって、以下の図に示すように CPU リソースが分割されます。

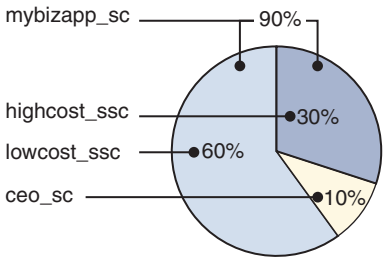


図 27. CPU リソース割り振り

ceo_sc サービス・スーパークラスは、10,000 の CPU シェアのうち 1000 を持っており、結果として CPU リソース割り振りは 10% になります。 mybizapp_sc サー

ビス・スーパークラスは、10,000 の CPU シェアのうち 9000 を持っており、結果として CPU リソース割り振りは 90% になります。 mybizapp_sc サービス・スーパークラスの CPU リソース割り振りの 90% のうち、lowcost_ssc サービス・サブクラスは 3 の共有のうち 2 を持っており、結果として CPU リソース割り振りがサービス・スーパークラスの CPU リソースの 66.7%、または CPU リソースの合計の 60% になります。 highcost_ssc サービス・サブクラスは 3 の共有のうち 1 を持っており、結果として CPU リソース割り振りがスーパークラスの CPU リソースの 33.3%、または CPU リソースの合計の 30% になります。

その他の情報

以下の主題について、完全な詳細が提供されています。

- ハード CPU シェアについては、 214 ページの『ハード CPU シェア』を参照してください。
- ソフト CPU シェアについては、 222 ページの『ソフト CPU シェア』を参照してください。
- CPU リミットについては、 228 ページの『CPU リミット』を参照してください。
- アクティブと見なされるサービス・クラスの CPU 使用率の最小レベルについては、 245 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率』を参照してください。
- ディスパッチ並行性レベルについては、 249 ページの『ディスパッチ並行性レベル』を参照してください。
- ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニングについては、 251 ページの『ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング』を参照してください。

マルチメンバー環境でのワークロード管理ディスパッチャーの動作

マルチメンバー DB2 インスタンスでワークロード管理ディスパッチャーが使用可能になっているとき、CPU リソースのスケジューリングは、指定されたホストのすべてのメンバーで作動します。ワークロード管理ディスパッチャーは AIX 上の共有 LPAR (マイクロパーティション) 環境もサポートすることに注意してください。

マルチメンバー・データベース環境

マルチメンバー DB2 インスタンスでは、ディスパッチャーは以下のように動作します。

- CPU シェアの相対的な数量が評価され、全体としてそのホストまたは LPAR 上のインスタンスでアクティブな作業に基づき、そのインスタンスのすべてのメンバー間に CPU リソースが割り振られます。例えば、2 つのサービス・クラス A と B で実行されている作業がある 2 つのメンバー・データベースを考えてみましょう。サービス・クラス A に 3500 のソフト CPU シェアが割り当てられ、サービス・クラス B に 6500 のソフト CPU シェアが割り当てられています。あるメンバーで実行されている作業が、もう一方で実行されている作業より相対的に多い可能性があるにもかかわらず、ホストまたは LPAR 全体の CPU 使用率は、サービス・クラス A で 35%、サービス・クラス B で 65% です。ワークロード管理ディスパッチャーは、任意の時点でどちらのサービス・クラスが、より多くの CPU リソースを取得する必要があるかに関して決定するときに、両方のメン

バーを見ています。

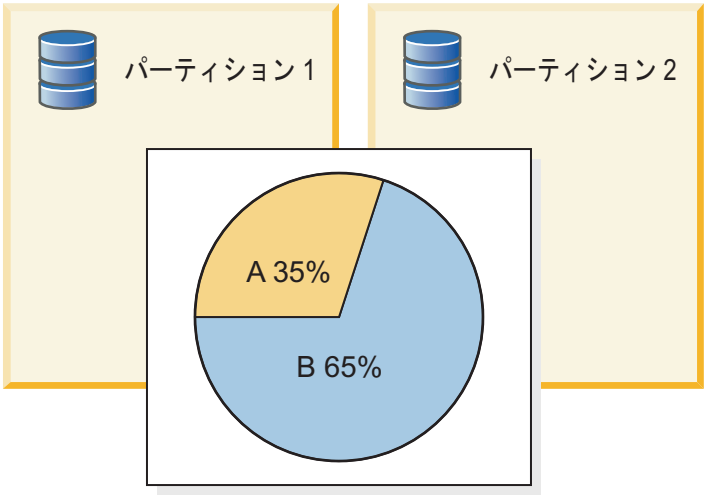


図 28. マルチメンバー・データベース環境: 2 つのデータベース・メンバー間のソフト CPU シェアの割り振りの円グラフ

- データベース・マネージャーの構成パラメーター **wlm_disp_concur** の値は、各ホストまたは LPAR に適用されます。マルチメンバー・データベースでは、**wlm_disp_concur** 構成パラメーターによって指定されたディスパッチ並行性レベルは、すべてのメンバーで適用されます。
- マルチメンバー・データベースでは、CPU リミットがホストごと、または LPAR ごとに強制されます。複数のメンバーが使用される場合、各パーティションは、すべてのメンバー間の合計が CPU リミット以下である限り、指定された CPU リミットのすべてを使用できます。単一のサービス・クラス A の CPU リミットが 35% で、2 つのメンバーがある場合、メンバー 1 は、メンバー 1 と 2 の間の CPU 使用率の合計が 35% を超えない限り、最大 35% 使用できます。

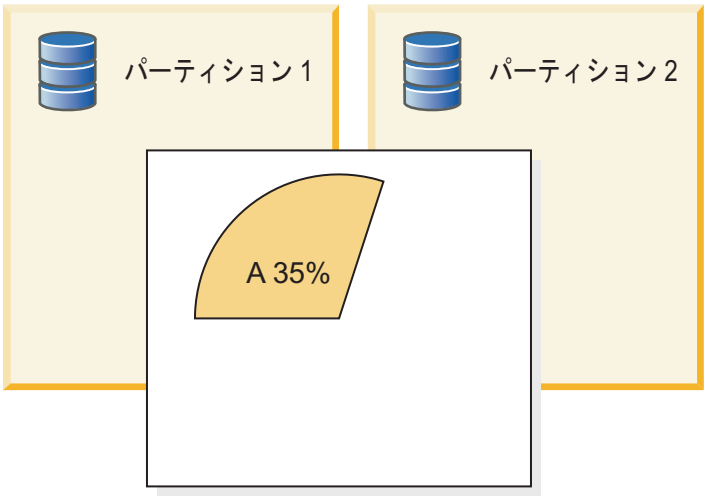


図 29. マルチメンバー・データベース環境: 2 つのデータベース・メンバー間の CPU リミットの円グラフ

マイクロパーティション (共有 LPAR) 環境

ワークロード管理ディスパッチャーは、AIX のマイクロパーティション環境をサポートします。マイクロパーティション環境では、ワークロード管理ディスパッチャーの制御とモニターの両方に対する CPU 使用率のパーセンテージは、ベースライン・レベルとして LPAR に対して許可された (保証された) CPU リソースを使用して、最新のディスパッチャーの CPU リソースのスケジューリング・サイクルで、オペレーティング・システムまたはハイパーバイザーによって LPAR に割り振られた CPU リソースに相対的に計算されます。 LPAR に割り振られた CPU リソースが許可された CPU リソースより大きい場合、ディスパッチャーは、より大きい量に相対的な CPU 使用率を計算し、CPU リソース割り振りが許可された CPU リソースより少ない場合、ディスパッチャーは許可された CPU リソースに相対的な CPU 使用率を計算します。

注: ワークロード管理ディスパッチャーの動作は、AIX WLM が計算を実行する方法と一貫性があります。 AIX 上の専用の LPAR、およびすべての非 AIX 環境では、CPU 使用率は、DB2 データベース・マネージャーが利用できる物理コアの完全な CPU キャパシティーに相対的に計算されます。

ワークロード管理ディスパッチャーを使用可能にする

DB2 ワークロード管理ディスパッチャーを使用可能にするには、**wlm_dispatcher** データベース・マネージャー構成パラメーターの値を YES に設定します。ワークロード管理ディスパッチャーは、DB2 ユーザー・サービス・クラスおよび保守サービス・クラスに割り振られる CPU リソースを管理します。

手順

コマンド行プロセッサー (CLP) を使用してワークロード管理ディスパッチャーを使用可能にするには、以下のようにします。

1. DB2 インスタンスにアタッチします。

   ```
   ATTACH TO instance-name
   ```

2. **UPDATE DATABASE MANAGER CONFIGURATION** (または **UPDATE DBM CFG**) コマンドを発行します。 **wlm_dispatcher** データベース・マネージャー構成パラメーターの値を YES に設定します。

   ```
   UPDATE DBM CFG USING wlm_dispatcher yes
   ```

3. オプション: DB2 インスタンスからデタッチします。

   ```
   DETACH
   ```

タスクの結果

ワークロード管理ディスパッチャーが使用可能になります。 CPU リソースの CPU リミットの割り振りを DB2 サービス・クラスに対して指定することができます。

次のタスク

ワークロード管理ディスパッチャーで CPU リソースを管理できるようにした後、以下のタスクを完了することを考慮してください。

- CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスを作成または変更し、CPU リミットを構成します。 243 ページの『CPU リミットの設定』を参照してください。
- **wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターを構成することにより、ハードおよびソフト CPU シェアを有効にします。 226 ページの『CPU シェアの使用可能化および設定』のステップ 1 を参照してください。
- CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスを作成または変更し、ハードまたはソフト CPU シェアを構成します。 226 ページの『CPU シェアの使用可能化および設定』を参照してください。
- **wlm_disp_min_util** データベース・マネージャー構成パラメーターを構成することにより、ワークロード管理ディスパッチャーでサービス・クラスをアクティブと見なすための最小 CPU 使用率 (パーセンテージ) を設定します。 247 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率の設定』を参照してください。
- **wlm_disp_concur** データベース・マネージャー構成パラメーターを構成することにより、ディスパッチ並行性レベルを設定します。 250 ページの『ディスパッチ並行性レベルの設定』を参照してください。

DB2 ワークロード管理ディスパッチャー・スケジューリングの正確性を最大にする

ワークロード管理ディスパッチャーのスケジューリングの正確性を最大にするために、DB2 データベース・マネージャーは **db2wlmt** タイマー・スレッドおよび **db2wlmtm** スケジューリング・スレッドへのリアルタイム優先度の割り当てを試みます。このような優先度の割り当てが成功するためには、オペレーティング・システムに応じて特定の認可が DB2 データベース・マネージャーに付与される必要があります。

手順

仮にタイマー・スレッドとスケジューリング・スレッドにリアルタイム優先度を割り当てることができない場合でも、DB2 ワークロード管理ディスパッチャーは作業の優先順位付けを実行できますが、正確さが劣り、最適な状態ではなくなります。以下では、ワークロード管理ディスパッチャーのスケジューリングの正確性を最大化するためのタスクをオペレーティング・システム別にリストします。該当するオペレーティング・システムを選んで、その指示に従ってください。

- AIX オペレーティング・システムで、AGENT PRIORITY を使用してサービス・クラスのエージェントにより高い相対優先順位を設定するためには、インスタンスの所有者に CAP_NUMA_ATTACH 機能および CAP_PROPAGATE 機能が必要です。これらの機能を付与するには、root としてログオンし、次のコマンドを実行します。

  ```
  chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
  ```

- Solaris 10 以降で、AGENT PRIORITY を使用してサービス・クラスのエージェントにより高い相対優先順位を設定するためには、インスタンス所有者に proc_priocntl 特権が必要です。この特権を付与するには、root としてログオンし、次のコマンドを実行します。

  ```
  usermod -K defaultpriv=basic,proc_priocntl db2user
  ```

この例では、ユーザー db2user のデフォルトの特権セットに proc_priocntl が追加されます。

さらに、Solaris の非グローバル・ゾーンで DB2 データベース・マネージャーが実行されている場合、ゾーンの制限特権セットに proc_priocntl 特権を追加する必要があります。この特権をゾーンに付与するには、root としてログオンし、次のコマンドを実行します。

```
global# zonecfg -z db2zone
zonecfg:db2zone> set limitpriv="default,proc_priocntl"
```

この例では、ゾーン db2zone の制限特権セットに proc_priocntl が追加されます。

- Solaris 9 では、DB2 データベース・マネージャーがスレッドの相対優先順位を引き上げるための機能はありません。このフィーチャーを使用するには Solaris 10 以降にアップグレードしてください。

ハード CPU シェア

DB2 ワークロード管理ディスパッチャーは、サービス・クラスに割り当てられている共有ベースの割り振りを使用して、CPU リソースを管理できます。影響が大きい、あるいは優先順位が低いと管理者が判断する作業を含むサービス・クラスにハード CPU シェアを割り当てると、そのサービス・クラスは、システム上で実行されている他のサービス・クラスの作業が存在している時に、CPU リソースのシェアを超えて消費することができなくなります。

概要

ハード CPU シェアは、任意のユーザー・サービス・クラスおよび保守サービス・クラスに割り当てることができますが、システム・サービス・クラスには割り当てることができません。ワークロード管理ディスパッチャーを使用可能にし、既存のワークロードをモニターして CPU リソースの消費の程度を判別し、サービス・クラスの CPU シェアの属性を使用可能にした後で、優先順位が低いあるいは影響が大きい作業を実行していると見なしているサービス・クラスに、ハード CPU シェアを割り当てることができます。ハード CPU シェアは、他のワークロードが存在する場合に、こうしたサービス・クラスの CPU 消費を制限します。これにより、これらのサービス・クラスによるシステムへの影響が抑制され、さらに残った CPU が確実に他のより高い優先順位を持つ作業のために確保されるようになります。

以下のセクションでは、ハード CPU シェアのフィーチャーと機能を、より詳細に説明します。使用法のシナリオのセクションでは、使用例と共にハード CPU シェアのフィーチャーと機能を説明します。

フィーチャーおよび機能

ホストまたはロジカル・パーティション (LPAR) が 100% の CPU 使用率で実行されている場合、サービス・クラス間の CPU リソースの割り振りには、単純に相対的シェアの割合が反映されます。一方、ホストまたは LPAR が完全な CPU 使用率に満たずに実行を開始する際、CPU リソースの再割り振りは複雑であり、アクティブな各サービス・クラスの CPU シェアの属性が、ソフト CPU シェアとハード CPU シェアのどちらに設定されているかによって異なります。

ハード CPU シェアが割り当てられているサービス・クラスは、CPU シェアの構成に示されている CPU リソース割り振りを超えて、ホストまたは LPAR で利用できる未使用の CPU リソースを消費することはできません。ワークロード管理ディスパッチャーは、作業が競合するサービス・スーパークラスで実行されている際、または同じサービス・スーパークラス内の競合するサービス・サブクラスで実行されている際に、割り当てられたハード CPU シェアの相対的な量によって決定される CPU リソース割り振りを常に考慮に入れます。競合するワークロードが存在しない場合、または競合するワークロードが一時的に完全にアイドル状態になった場合、ハード CPU シェアが指定されたサービス・クラスは、未使用の CPU リソースを要求することができます。

ハード CPU シェアの設定は、このサービス・クラスで実行されている作業が、ホストまたは LPAR で実行されているより重要な作業を中断しないように、サービス・クラスに CPU リソース割り振りを厳密に強制する場合に使用すると、最も効果があります。ハード CPU シェアは、入出力、バッファー・プール、CPU キャッシュなど、リソースの競合のために優先順位の高い作業のパフォーマンスに影響を与える可能性がある、複雑な照会または負荷の高い照会を実行するサービス・クラスに割り当てます。

CPU シェアの属性を使用可能にするには、データベース・マネージャーの構成パラメーター **wlm_disp_cpu_shares** の値を YES に設定する必要があります。このパラメーターのデフォルトの設定値は NO です。このパラメーターを使用可能にした後で、すべての既存のサービス・クラスおよび新しく作成されたサービス・クラスは、デフォルトで 1000 のハード CPU シェアが割り当てられ、CPU リソースの等しい配布が最初になされます。 CREATE SERVICE CLASS ステートメントおよび ALTER SERVICE CLASS ステートメントを使用することによって、ハード CPU シェアを割り当てて調整できます。ハード CPU シェアを使用可能にする方法、およびその設定方法の完全な詳細については、226 ページの『CPU シェアの使用可能化および設定』を参照してください。

サービス・クラスに割り当てられている CPU シェアの数に基づいて、ワークロード管理は、各サービス・クラスが使用を許可されている CPU リソースの比率を計算します。各サービス・スーパークラスが許可されている CPU リソースの比率を判別するために、以下の公式を使用して、特定のサービス・スーパークラスの CPU シェアの数を、ワークロード管理ディスパッチャーによって割り振られている CPU リソースのパーセンテージに変換できます。

```
% CPU(スーパークラス) = (スーパークラス共有の数 /
  アクティブ・スーパークラスすべての共有総数) x 100
```

各サービス・サブクラスが許可されている CPU リソースの比率を判別するには、以下の公式を使用して、特定のサービス・サブクラスの CPU シェアの数を、ワークロード管理ディスパッチャーによって割り振られている CPU リソースのパーセンテージに変換できます。

```
% CPU(サブクラス) = % CPU(スーパークラス) x (サブクラス共有の数 /
  スーパークラス内のアクティブ・サブクラスすべての共有総数)
```

注: すべてのアクティブなスーパークラスの (ハードとソフトの両方の) CPU シェアの総数は、ホストまたは LPAR 上のすべてのデータベースおよびすべてのメンバー間でカウントされます。

使用法のシナリオ

シナリオ 1

217 ページの図 30 パネル A では、サービス・クラス B にハード CPU シェアが割り当てられ、サービス・クラス A と C にソフト CPU シェアが割り当てられています。その量は、図の凡例で説明されています。円グラフは、これらのアクティブな各サービス・クラスが許可された、割り振られた CPU リソースの比率を表し、各サービス・クラスは、CPU リソースの完全な共有を使用しているため、この例では、合計が 100% の CPU 使用率になっています。パネル B では、サービス・クラス A には CPU 割り当て全体に使用するのに十分な作業がなく、CPU 使用率が 60% から 50% に落ちています。サービス・クラス A によって一時的に解放された CPU リソースの未使用の 10% は、ソフト CPU シェアの割り当てに基づき、競合するサービス・クラス C のみが要求できます。この例では、サービス・クラス B で 30% の CPU リソース割り振りを超えることはできません。これは、ハード CPU シェアが割り当てられており、サービス・クラス A と C で実行されている作業が、ディスパッチャーによってそれらのサービス・クラスがアクティブと見なされるまでに十分存在するためです。パネル C は、サービス・クラス C への CPU リソースの再割り振りの総計を示しており、利用可能な CPU リソースの総計の 10% から 20% に増えています。

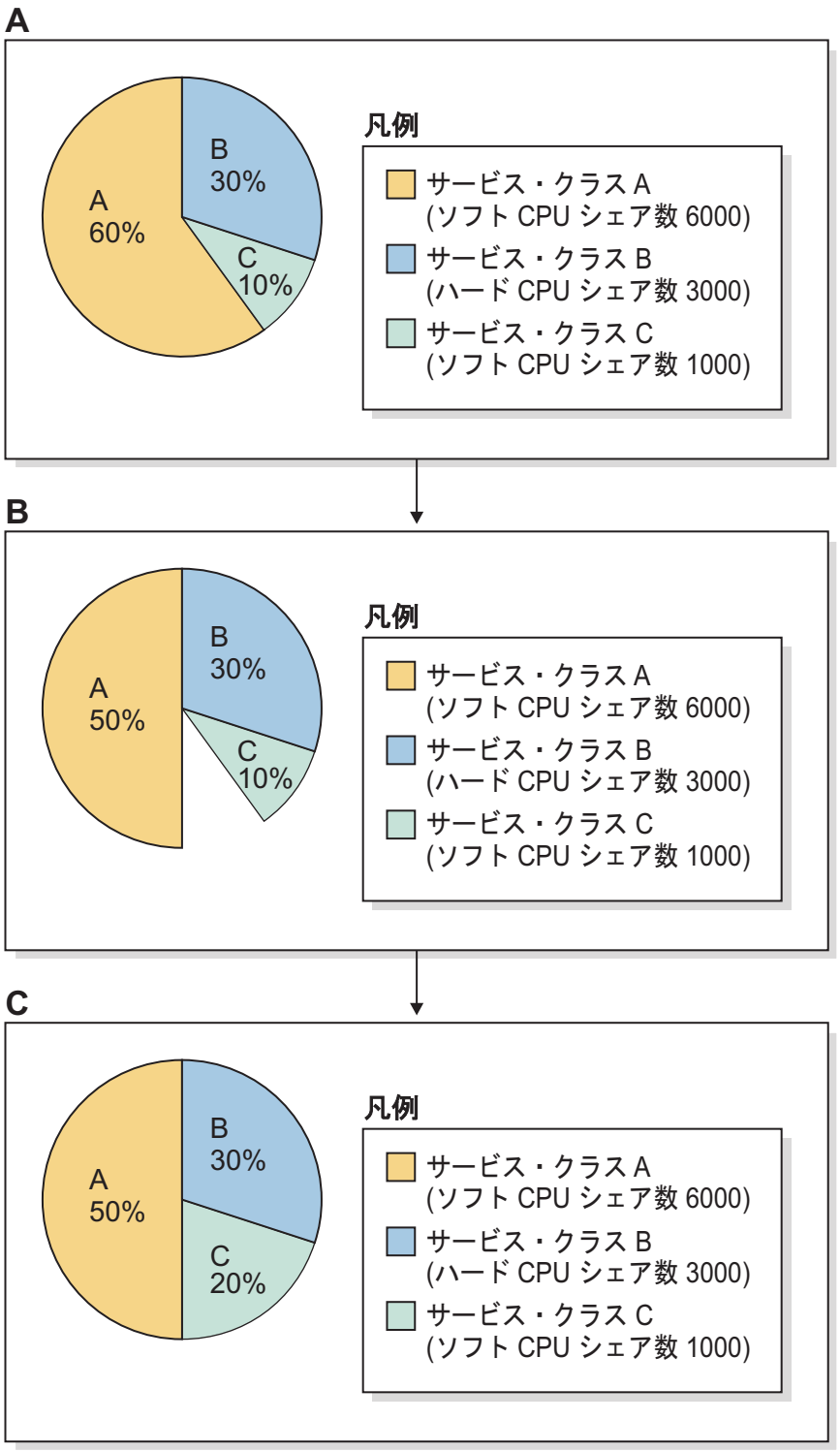


図 30. ハード CPU シェアとソフト CPU シェアの円グラフ: シナリオ 1

サービス・クラス A でワークロードが増えた場合は、CPU リソースの要求を効率的に増やします。この状況では、サービス・クラス C はサービス・クラス A にすべての要求された CPU リソースをすぐに解放するため、CPU リソース割り振りの状態は、パネル A の円グラフで示されている状態に戻ります。

注: サービス・クラスが競合して CPU リソースを消費する際、各サービス・クラスによる CPU リソースの要求は、先着順でワークロード管理ディスパッチャーによって処理されます。ビジー状態のホストまたは LPAR での CPU リソースの要求は、多くの場合、頻度が高く短期間であるため、時間の経過と共に発生する未使用 CPU リソースの再割り振りは、相対的な CPU シェアの割り当てに比例した CPU リソースの平滑化された再配布になります。

シナリオ 2

ハード共有の使用により、CPU リソースの一部が、データベース・サーバーで使用率が低い状態になることがあります。使用率の低い CPU リソースは、他のサービス・クラスが、CPU リソース割り振りを完全に使用するのに十分大きいワークロードを実行していない場合に発生することがあります。使用率の低い CPU リソースは、CPU リソースが完全な使用率を下回る場合でも、他のサービス・クラスの進行を干渉する可能性がある負荷の高いワークロードを制限するために、ハード共有が使用されている場合に適しています。この状況は、通常、 I/O または CPU キャッシュなどのリソースの競合によって発生します。

219 ページの図 31 パネル A では、サービス・クラス B と C にハード CPU シェアが割り当てられ、サービス・クラス A にソフト CPU シェアが割り当てられています。その量は、図の凡例で説明されています。円グラフは、これらのアクティブな各サービス・クラスが許可された、割り振られた CPU リソースの比率を表し、各サービス・クラスは、CPU リソースの完全な共有を使用しているため、この例では、合計が 100% の CPU 使用率になっています。パネル B では、サービス・クラス A には CPU 割り当て全体に使用するのに十分な作業がなく、CPU 使用率が 60% から 50% に落ちています。サービス・クラス A によって一時的に解放された CPU リソースの未使用の 10% は、ハード CPU シェアの割り当てに基づき、競合するサービス・クラス B と C が要求することはできません。この例では、サービス・クラス B と C において、それぞれ 30% と 10% の CPU リソース割り振りを超えることはできません。これは、両方ともハード CPU シェアが割り当てられており、サービス・クラス A で実行されている作業が、ディスパッチャーによってそのサービス・クラスがアクティブと見なされるほど十分存在するためです (CPU 使用率は、データベース・マネージャーの構成パラメーター **wlm_disp_min_util** で構成されているレベルを下回ります。デフォルトは 5% です)。パネル C は、未使用の CPU リソースが再割り振りされておらず、このシナリオで、CPU リソースの使用率が低いままであることを示しています。

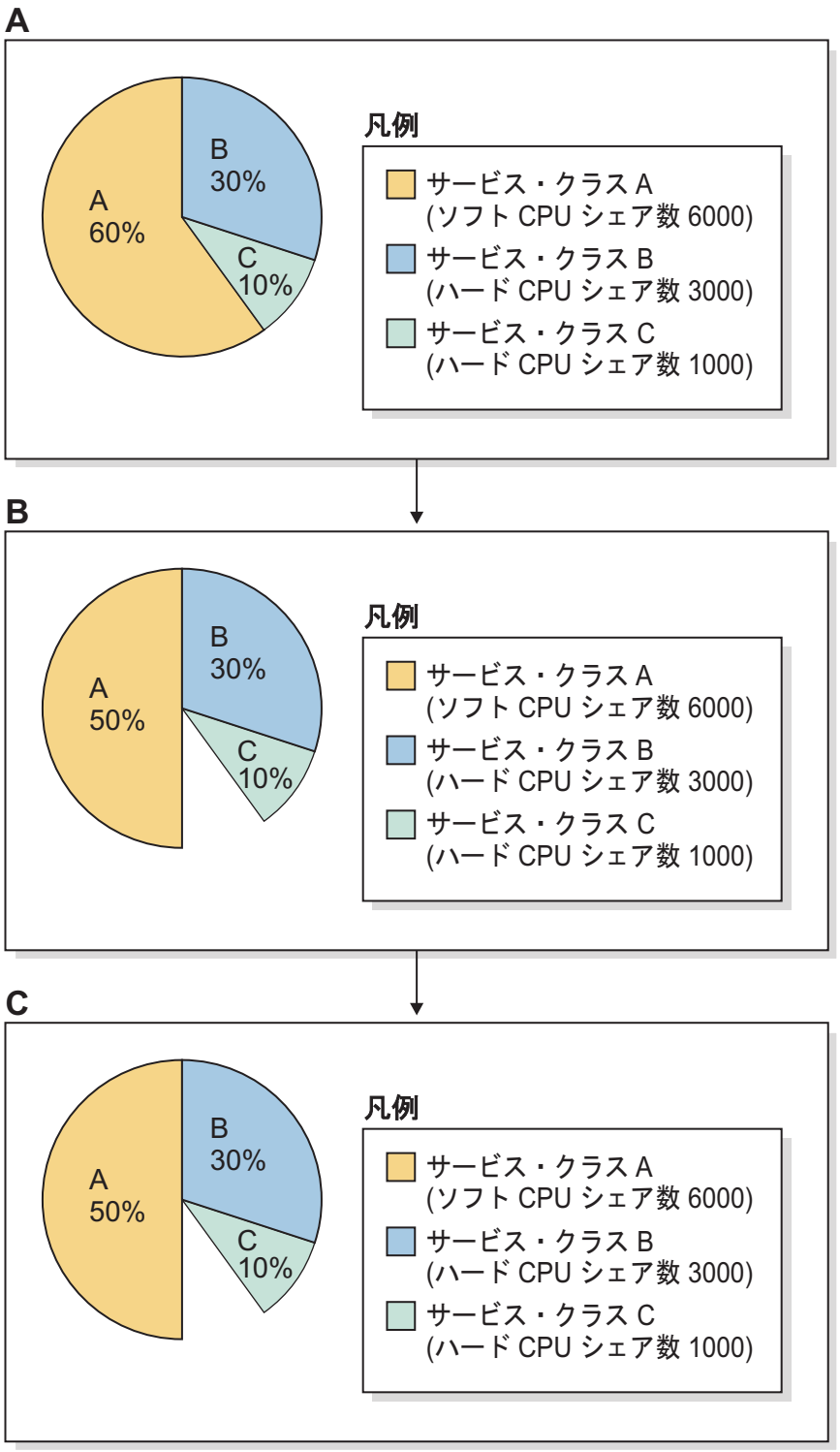


図 31. ハード CPU シェアとソフト CPU シェアの円グラフ: シナリオ 2

このシナリオは、優先順位の高いサービス・クラスで実行している作業の進行を、優先順位の低いサービス・クラスで実行している作業による中断から保護できることを示しています。

シナリオ 3

ハード CPU シェアは、従来の固定パーセント CPU リミットよりもメリットがあります。ハード CPU シェアが指定されたサービス・クラスは、他のワークロードが存在しない場合に、未使用の CPU リソースを要求する柔軟性があります。このため、ハード CPU シェアが割り当てられているサービス・クラスは、固定された CPU リミットが指定されたサービス・クラスで発生するのとは異なり、他の作業がホストまたは LPAR に存在しない場合に、故意に制限されることはありません。

219 ページの図 31 パネル A では、サービス・クラス B と C にハード CPU シェアが割り当てられ、サービス・クラス A にソフト CPU シェアが割り当てられています。その量は、図の凡例で説明されています。円グラフは、これらのアクティブな各サービス・クラスが許可された、割り振られた CPU リソースの比率を表し、各サービス・クラスは、CPU リソースの完全な共有を使用しているため、この例では、合計が 100% の CPU 使用率になっています。パネル B では、サービス・クラス A には CPU 割り当て全体を使用するための作業がまったくなく、CPU 使用率が 60% から 0% に落ちています。ディスパッチャーは、サービス・クラス A を非アクティブと見なします。サービス・クラス A によって一時的に解放された CPU リソースの未使用の 60% は、ハード CPU シェアの割り当て、および非アクティブなサービス・クラスに基づき、競合するサービス・クラス B と C が要求できます。この状況では、サービス・クラス B と C において、それぞれ 30% と 10% の CPU リソース割り振りを超えることができます。これは、両方ともハード CPU シェアが割り当てられており、サービス・クラス A で実行されている作業が、ディスパッチャーによってそのサービス・クラスがアクティブと見なされるのには不十分であるためです (CPU 使用率は、データベース・マネージャーの構成パラメーター **wlm_disp_min_util** で構成されているレベルを下回ります。デフォルトは 1% です)。パネル C は、サービス・クラス B に CPU リソースの 75% ((3000 / (3000 + 1000)) x 100) が割り振られており、サービス・クラス C に 25% ((1000 / (3000 + 1000)) x 100) が割り振られていることを示しています。

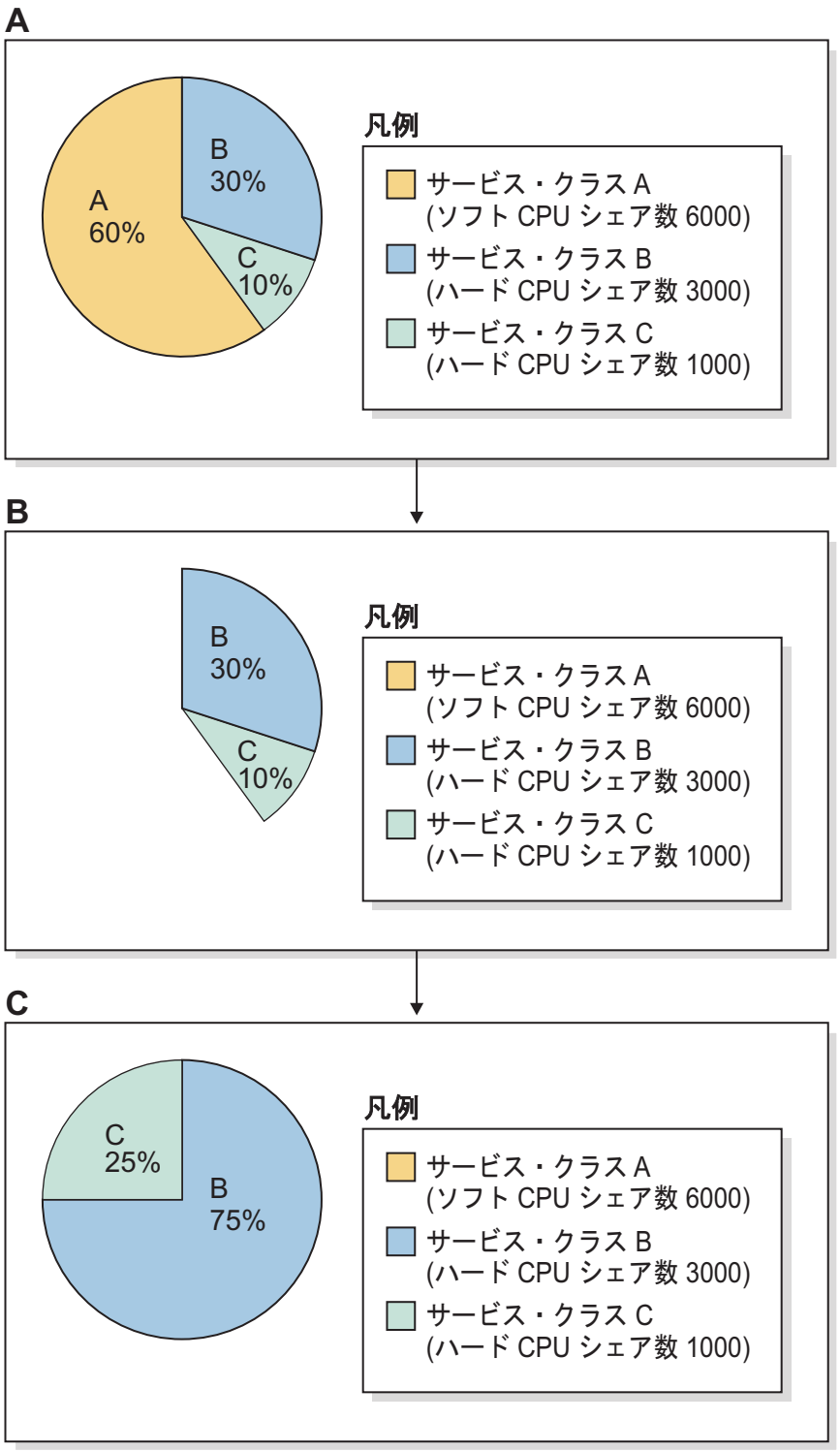


図 32. ハード CPU シェアとソフト CPU シェアの円グラフ: シナリオ 3

サービス・クラス A でワークロードが増えた場合は、CPU リソースの要求を効率的に増やします。この状況では、サービス・クラス B と C はサービス・クラス A にすべての要求された CPU リソースをすぐに解放するため、CPU リソース割り振りの状態を、パネル A の円グラフで示されている状態にリストアします。

このシナリオは、優先順位の高いサービス・クラスで実行している作業の進行を、優先順位の低いサービス・クラスで実行している作業による中断から保護できるが、優先順位の高い作業が存在しなくなったときに (オフピークの営業時間など)、ハード CPU シェアが指定された優先順位の低いサービス・クラスが、未使用の CPU リソースを要求する柔軟性を持っていることを示しています。

注: サービス・クラス B または C のいずれかが、CPU リソース割り振りを完全に使用していないにもかかわらず、ディスパッチャーによってアクティブに実行されている作業と見なされている場合、ハード CPU シェアが指定されているもう一方のサービス・クラスは、それを利用して割り振りより多く使用することができません。

その他の情報

以下のワークロード管理ディスパッチャーの主題について、完全な詳細が提供されています。

- ワークロード管理ディスパッチャーについては、190 ページの『ワークロード管理ディスパッチャー』を参照してください。
- ソフト CPU シェアについては、『ソフト CPU シェア』を参照してください。
- CPU リミットについては、228 ページの『CPU リミット』を参照してください。
- アクティブと見なされるサービス・クラスの CPU 使用率の最小レベルについては、245 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率』を参照してください。
- ディスパッチ並行性レベルについては、249 ページの『ディスパッチ並行性レベル』を参照してください。
- ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニングについては、251 ページの『ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング』を参照してください。

ソフト CPU シェア

DB2 ワークロード管理ディスパッチャーは、サービス・クラスに割り当てられている共有ベースの割り振りを使用して、CPU リソースを管理できます。ソフト CPU シェアは、管理者によってサービス・クラスに割り当てられると、CPU リソースが未使用な場合、そのシェアより多く消費できる機能をサービス・クラスに与えます。ハード CPU シェアによってバインドされている他のサービス・クラスと連携して使用すると、CPU リソースに関してそのサービス・クラスを優遇する方法が提供されます。

概要

ソフト CPU シェアは、任意のユーザー・サービス・クラスおよび保守サービス・クラスに割り当てることができますが、システム・サービス・クラスには割り当てることができません。ワークロード管理ディスパッチャーを使用可能にし、既存のワークロードをモニターして CPU リソースの消費の程度を判別した後で、優先順位が高いと見なされているサービス・クラスにソフト CPU シェアを割り当てることができます。ソフト CPU シェアは、優先順位が高いワークロードで使用される場合に最も効果的です。これは、システム上にアイドル状態の CPU リソースが存

在する場合、指定された割り振りより多くの割り振りをワークロードが消費できるようになるためです。優先順位が低い作業あるいは影響が大きい作業の CPU 使用量を制約する場合は、ソフト CPU シェアの使用は推奨されていません。この場合は、代わりにハード CPU シェアを使用してください。

以下のセクションでは、ソフト CPU シェアのフィーチャーと機能を、より詳細に説明します。使用法のシナリオのセクションでは、使用例と共にソフト CPU シェアのフィーチャーと機能を説明します。

フィーチャーおよび機能

ホストまたはロジカル・パーティション (LPAR) が 100% の CPU 使用率で実行されている場合、サービス・クラス間の CPU リソースの割り振りには、単純に相対的な共有の割合が反映されます。一方、ホストまたは LPAR が完全な CPU 使用率に満たずに実行を開始する際、CPU リソースの再割り振りは複雑であり、アクティブな各サービス・クラスの CPU シェアの属性が、ソフト CPU シェアとハード CPU シェアのどちらに設定されているかによって異なります。

ソフト CPU シェアが割り当てられているサービス・クラスは、CPU シェアの構成に示されている CPU リソース割り振りを超えて、ホストまたは LPAR で利用できる未使用の CPU リソースを消費することができます。 2 つ以上のサービス・クラスがソフト共有を持ち、未使用の CPU リソースが、予備のキャパシティーを消費するための各サービス・クラスからの十分な CPU リソースの要求で使用可能になった場合、競合するサービス・クラスへの CPU リソースの割り振りは、アクティブな各サービス・クラスの相対的な共有に従った比率でなされます。ソフト CPU シェアの設定は、利用可能な予備の CPU リソースを一時的に要求可能にしようとしている、優先順位の高い作業に対して最も効率的です。また、ソフト CPU シェアの設定は、即時の CPU 消費の外部の、データベース・リソースに比較的影響が少ないと予想されている短い照会から構成されるワークロードに対して最も効率的です。

CPU シェアの属性を使用可能にするには、データベース・マネージャーの構成パラメーター **wlm_disp_cpu_shares** の値を YES に設定する必要があります。このパラメーターのデフォルトの設定値は NO です。CREATE SERVICE CLASS ステートメントおよび ALTER SERVICE CLASS ステートメントを使用することによって、ソフト CPU シェアを割り当てて調整できます。ソフト CPU シェアを使用可能にする方法、またその設定方法の完全な詳細については、226 ページの『CPU シェアの使用可能化および設定』を参照してください。

サービス・クラスに割り当てられている CPU シェアの数に基づいて、ワークロード管理は、各サービス・クラスが使用を許可されている CPU リソースの比率を計算します。各サービス・スーパークラスが許可されている CPU リソースの比率を判別するために、以下の公式を使用して、特定のサービス・スーパークラスの CPU シェアの数を、ワークロード管理ディスパッチャーによって割り振られている CPU リソースのパーセンテージに変換できます。

```
% CPU(スーパークラス) = (スーパークラス共有の数 /
  アクティブ・スーパークラスすべての共有総数) x 100
```

各サービス・サブクラスが許可されている CPU リソースの比率を判別するには、以下の公式を使用して、特定のサービス・サブクラスの CPU シェアの数を、ワークロード管理ディスパッチャーによって割り振られている CPU リソースのパーセンテージに変換できます。

```
% CPU(サブクラス) = % CPU(スーパークラス) x (サブクラス共有の数 /
  スーパークラス内のアクティブ・サブクラスすべての共有総数)
```

注: すべてのアクティブなスーパークラスの (ハードとソフトの両方の) CPU シェアの総数は、ホストまたは LPAR 上のすべてのデータベースおよびすべてのメンバー間でカウントされます。

使用法のシナリオ

225 ページの図 33 パネル A では、サービス・クラス A、B、および C にソフト CPU シェアが割り当てられ、その量は、図の凡例で説明されています。円グラフは、これらのアクティブな各サービス・クラスが許可された、割り振られた CPU リソースの比率を表し、各サービス・クラスは、CPU リソースの完全な共有を使用しているため、この例では、合計が 100% の CPU 使用率になっています。パネル B では、サービス・クラス A には CPU 割り当て全体に使用するのに十分な作業がなく、CPU 使用率が 60% から 50% に落ちています。サービス・クラス A によって一時的に解放された CPU リソースの未使用の 10% は、相対的なソフト CPU シェアの割り当てに基づき、競合するサービス・クラス B と C が比率に応じて要求することができます。パネル C は、サービス・クラス B と C の間での CPU リソースの比例再割り振りを表しています。サービス・クラス A によって解放された未使用の 10% のうち、サービス・クラス B は 7.5% (10% x (3000/4000)) を取得し、サービス・クラス C は 2.5% (10% x (1000/4000)) を取得します。

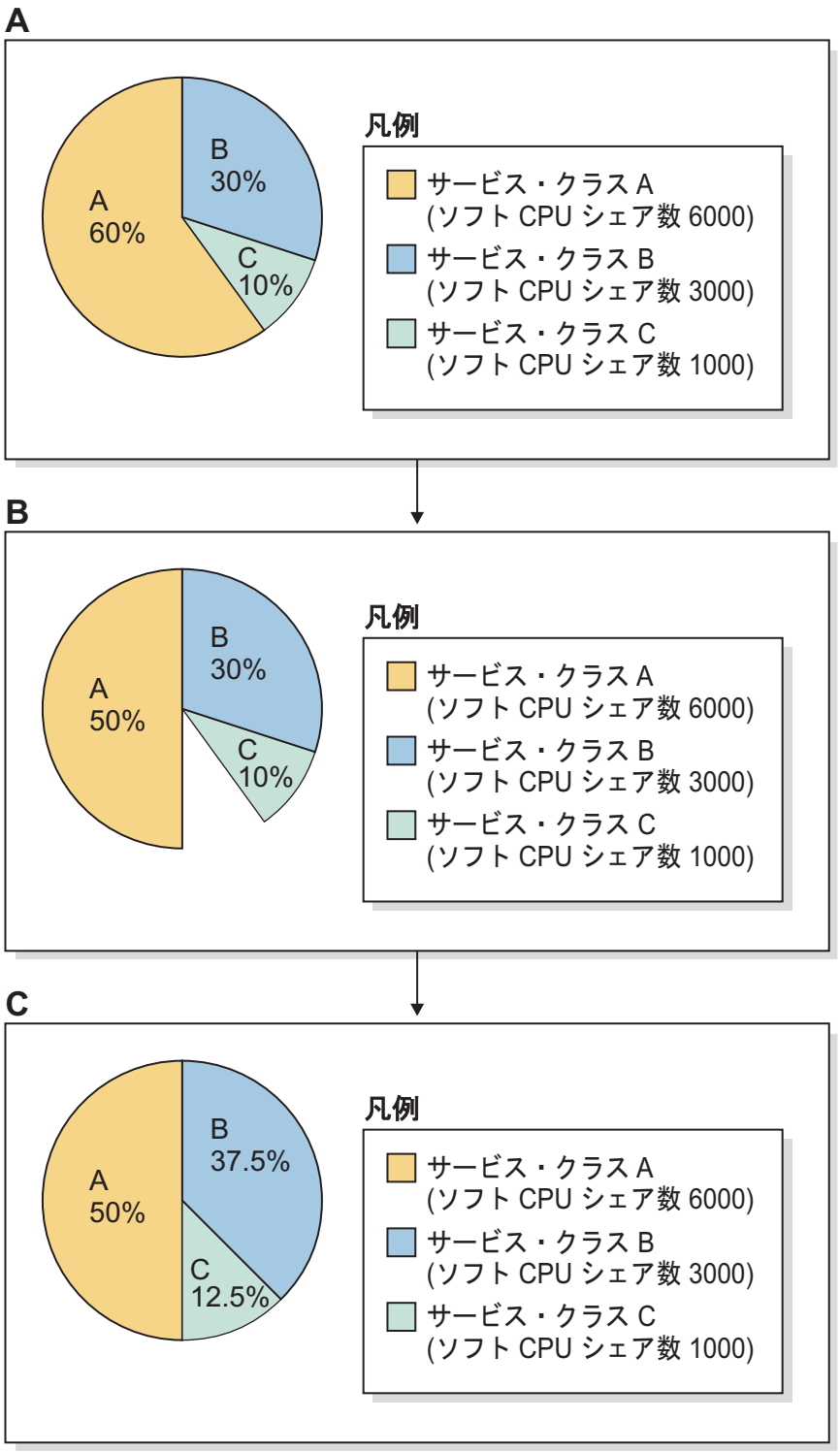


図 33. ソフト CPU シェアの円グラフ

サービス・クラス A でワークロードが増えた場合は、CPU リソースの要求を効率的に増やします。この状況では、サービス・クラス B と C はサービス・クラス A にすべての要求された CPU リソースをすぐに解放するため、CPU リソース割り振りの状態は、パネル A の円グラフで示されている状態に戻ります。

注: サービス・クラスが競合して CPU リソースを消費する際、各サービス・クラスによる CPU リソースの要求は、先着順でワークロード管理ディスパッチャーによって処理されます。ビジー状態のホストまたは LPAR での CPU リソースの要求は、多くの場合、頻度が高く短期間であるため、未使用 CPU リソースの再割り当ては、時間の経過とともに、相対 CPU シェアの割り当てに比例した平滑化された CPU リソースの分配になります。

その他の情報

以下のワークロード管理ディスパッチャーの主題について、完全な詳細が提供されています。

- ワークロード管理ディスパッチャーについては、 190 ページの『ワークロード管理ディスパッチャー』を参照してください。
- ハード CPU シェアについては、 214 ページの『ハード CPU シェア』を参照してください。
- CPU リミットについては、 228 ページの『CPU リミット』を参照してください。
- アクティブと見なされるサービス・クラスの CPU 使用率の最小レベルについては、 245 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率』を参照してください。
- ディスパッチ並行性レベルについては、 249 ページの『ディスパッチ並行性レベル』を参照してください。
- ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニングについては、 251 ページの『ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング』を参照してください。

CPU シェアの使用可能化および設定

wlm_disp_cpu_shares データベース・マネージャー構成パラメーターの値を YES に設定することにより、CPU シェア属性を使用可能にします。 CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってハードまたはソフト CPU シェア属性を設定します。通常は、優先順位が低い作業あるいは影響が大きい作業を実行しているサービス・クラスでハード CPU シェアを設定し、CPU リソースの負担が大きい場合は、優先度の高いサービス・クラスにソフト CPU シェアを設定することができます。

始める前に

注: CPU シェアの設定を有効にするには、**wlm_dispatcher** データベース・マネージャー構成パラメーターを構成することでワークロード管理ディスパッチャーを使用可能にする必要があります。これを行うタイミングは、CPU シェアを使用可能化/設定する前、または使用可能化/設定した後のどちらでも可能です。 212 ページの『ワークロード管理ディスパッチャーを使用可能にする』を参照してください。

このタスクについて

このタスクでは、コマンド行プロセッサー方式を使って CPU シェア属性を使用可能にし、CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスの CPU シェア属性を設定します。

制約事項

ワークロード管理ディスパッチャーが制御できるサービス・クラス、つまりユーザー・サービス・クラスおよび保守サービス・クラスにのみ、ソフト CPU シェアを割り当てることができます。システム・サービス・クラスに関する CPU リソース割り振りをワークロード管理ディスパッチャーで制御することはできません。

手順

CPU シェア属性が既に使用可能になっている場合は、スキップしてステップ 2 に進んでください。
CPU シェア属性を使用可能にするには、次のようにします。

1. 次のようにコマンド行プロセッサー (CLP) を使って **UPDATE DATABASE MANAGER CONFIGURATION** (または **UPDATE DBM CFG**) コマンドを発行し、**wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターの値を YES に設定します。最初に DB2 インスタンスに接続することで、パラメーター値が直ちに更新されます。

   ```
   attach to instance-name
   update dbm cfg using wlm_disp_cpu_shares yes
   detach
   ```

既存のサービス・クラスのハードまたはソフト CPU シェア設定を変更する場合は、スキップしてステップ 3 に進んでください。

新しいサービス・クラスを作成してハードまたはソフト CPU シェア属性を設定するには、次のようにします。

2. CREATE SERVICE CLASS ステートメントを発行して新しいサービス・クラスを作成し、ハードまたはソフト CPU シェア値を (この例では) 5000 に設定します。
   - `create service class service-class-name hard cpu shares 5000`
   - `create service class service-class-name soft cpu shares 5000`

既存のサービス・クラスのハードまたはソフト CPU シェアの設定を変更するには、次のようにします。

3. ALTER SERVICE CLASS ステートメントを発行して、ハードまたはソフト CPU シェア値を (この例では) 15000 に変更します。
   - `alter service class service-class-name hard cpu shares 15000`
   - `alter service class service-class-name soft cpu shares 15000`

タスクの結果

CPU シェア属性が使用可能になり、指定したサービス・クラスのハードまたはソフト CPU シェアの設定または変更が完了しました。ハード CPU シェアが割り当てられたサービス・クラスは、制限された状態で、ワークロード要件が緩和されたために他のサービス・クラスによって解放された CPU リソースを要求することができます。一方、ソフト CPU シェアが割り当てられたサービス・クラスは、ワークロード要件が緩和されたために他のサービス・クラスによって解放された CPU リソースを、常に要求することができます。

次のタスク

ワークロード管理ディスパッチャーによる CPU リソース管理を有効にし、CPU シェア属性を使用可能にしてハードまたはソフト CPU シェア属性を設定した後、以下のタスクを完了することを考慮してください。

- ハードまたはソフト CPU シェアの設定を有効にするには、**wlm_dispatcher** データベース・マネージャー構成パラメーターを構成することでワークロード管理ディスパッチャーを使用可能にする必要があります。 212 ページの『ワークロード管理ディスパッチャーを使用可能にする』を参照してください。
- **wlm_disp_concur** データベース・マネージャー構成パラメーターを構成することにより、ディスパッチ並行性レベルを設定します。 250 ページの『ディスパッチ並行性レベルの設定』を参照してください。
- CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスを作成または変更し、CPU リミットを構成します。 243 ページの『CPU リミットの設定』を参照してください。
- **wlm_disp_min_util** データベース・マネージャー構成パラメーターを構成することにより、ワークロード管理ディスパッチャーでサービス・クラスをアクティブと見なすための最小 CPU 使用率 (パーセンテージ) を設定します。 247 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率の設定』を参照してください。

CPU リミット

DB2 ワークロード管理ディスパッチャーは、サービス・スーパークラスとサービス・サブクラスに割り当て可能な、固定の CPU リミットを設定できます。 CPU リミットを適用することにより、DB2 データベース・マネージャーで実行されている他の作業に関係なく、サービス・クラスによって消費される CPU をシステム上の固定量に制限できます。これにより、CPU リソースの残りの部分は他のコンシューマーが使用できます。CPU リミットを CPU シェアと一緒に使用する場合、最も制限的または限定的な条件が常に優先されます。

概要

CPU リミットは、任意のユーザー・サービス・クラスおよび保守サービス・クラスに割り当てることができますが、システム・サービス・クラスには割り当てることができません。ワークロード管理ディスパッチャーを使用可能にし、既存のワークロードをモニターして CPU リソースの消費の程度を判別した後、CPU 使用量を厳密に制限するサービス・クラスに CPU リミットを割り当てることができます。

CPU シェアでは、ホストまたは LPAR の全体的なワークロードの負荷が高い場合には個別のワークロードの CPU リソース割り振りを制御し、全体的なワークロードの負荷が低い場合には CPU リソースを無駄にしないよう制御する機能が提供されます。しかし、ワークロードによっては、ホストまたは LPAR の全体的なワークロードの負荷が低いにもかかわらず、CPU リソース割り振りを常に制限することが望ましい場合もあります。例えば、複数の部門がデータベース・サーバーの購入コストを共有している場合、選択された構成によってはホストまたは LPAR の CPU リソースの使用率が低くなる可能性があるにもかかわらず、どの部門も、他の部門がそれぞれの割り振り分を超えて CPU リソースを消費することがないようにする

ことを望むということがあるかもしれません。 CPU シェアでは、このレベルの制御が提供されませんが、CPU リミットではそれが可能です。

以下のセクションでは、CPU リミットのフィーチャーと機能を、より詳細に説明します。使用法のシナリオのセクションでは、使用例と共に CPU リミットのフィーチャーと機能を説明します。

フィーチャーおよび機能

CPU リミットは、サービス・クラスの作業により CPU リソース割り振りに一定の制限を与えることができます。 CPU リミットがすべてのサービス・クラスに設定されている場合、作業を実行する CPU リソースの一部を、DB2 データベース・マネージャーで実行されている他の作業に関係なく予約できます。サービス・クラスに CPU リミットを構成すると、ワークロードに厳密に強制されたサンドボックスを効率的に提供し、それによりワークロード間の CPU リソースの消費の公平性を達成できますが、CPU リソースの完全な使用率を達成できないことがあるという犠牲を払います。

CPU リミットは、ホストまたは LPAR 上に複数の DB2 インスタンスがある場合にも役立ちます。インスタンス・レベルのワークロード管理ディスパッチャーの操作により、サービス・クラスの CPU リソース割り振りは、そのインスタンス内のその他すべてのサービス・クラスの共有に相対的なサービス・クラスの共有から計算されます。ただし、CPU リミットは、ホストまたは LPAR 上に存在する DB2 インスタンスの数に関係なく、そのようなホストまたは LPAR の CPU リソースのパーセンテージとして表現されます。 CPU リミットをスーパークラスに、共有をサブクラスに割り当てることによって、各スーパークラスの絶対的な CPU リソース割り振りを制御するために CPU リミットを使用でき、インスタンスを拡張することによって、共有を使用して、それらのスーパークラス内で実行されているサービス・サブクラスの相対的な CPU リソース割り振りを制御できます。

CPU リミットは、サービス・スーパークラスのすべてのサブクラスによるホストまたは LPAR 上の CPU リソース割り振りの制限のパーセンテージを表すサービス・スーパークラス・レベルで構成するか、あるいは特定のサブクラスによるホストまたは LPAR 上の CPU リソース割り振りの制限のパーセンテージを表すサブクラス・レベルで構成することができます。

CPU リミットの属性を使用可能にするには、データベース・マネージャーの構成パラメーター **wlm_dispatcher** の値を ON に設定することによって、ワークロード管理ディスパッチャーを使用可能にする必要があります。このパラメーターのデフォルトの設定値は OFF です。ワークロード管理ディスパッチャーを使用可能にすることによって、CPU リミットの属性を使用した CPU リソースの制御がデフォルトで利用可能になります。CREATE SERVICE CLASS ステートメントおよび ALTER SERVICE CLASS ステートメントを使用することによって、CPU リミットを割り当てて調整できます。 CPU リミットを設定する方法の完全な詳細については、 243 ページの『CPU リミットの設定』を参照してください。

ワークロード管理ディスパッチャーは、CPU リソースをサービス・クラスに割り振るときに、最も制限的な CPU リミットまたは CPU シェアの割り当てを常に尊重します。 例えば、CPU リミットがスーパークラス・レベルおよびサブクラス・レベルの両方で設定されている場合、より制限的な CPU リミットが尊重されます。

同じように、サービス・クラスが共有ベースの CPU リソース割り振り全体に使用する前に、そのサービス・クラスの CPU リミットに到達した場合、ディスパッチャーは CPU リミットを尊重します。

使用法のシナリオ

CPU リミットと複数のスーパークラス

この使用法の例は、複数のスーパークラスがある環境での CPU リミットの動作を説明します。

231 ページの図 34は、2 つのスーパークラス A と B で構成されているホストまたは LPAR を示します。基本的な概念を説明する図では、デフォルトのユーザー、保守、およびシステムのサービス・クラスで実行されている無視できる作業があると仮定します。以下のシナリオでは、1 つのデータベースを持つ 1 つの DB2 インスタンスしかなく、このホストまたは LPAR に 1 つのメンバーしかありません。

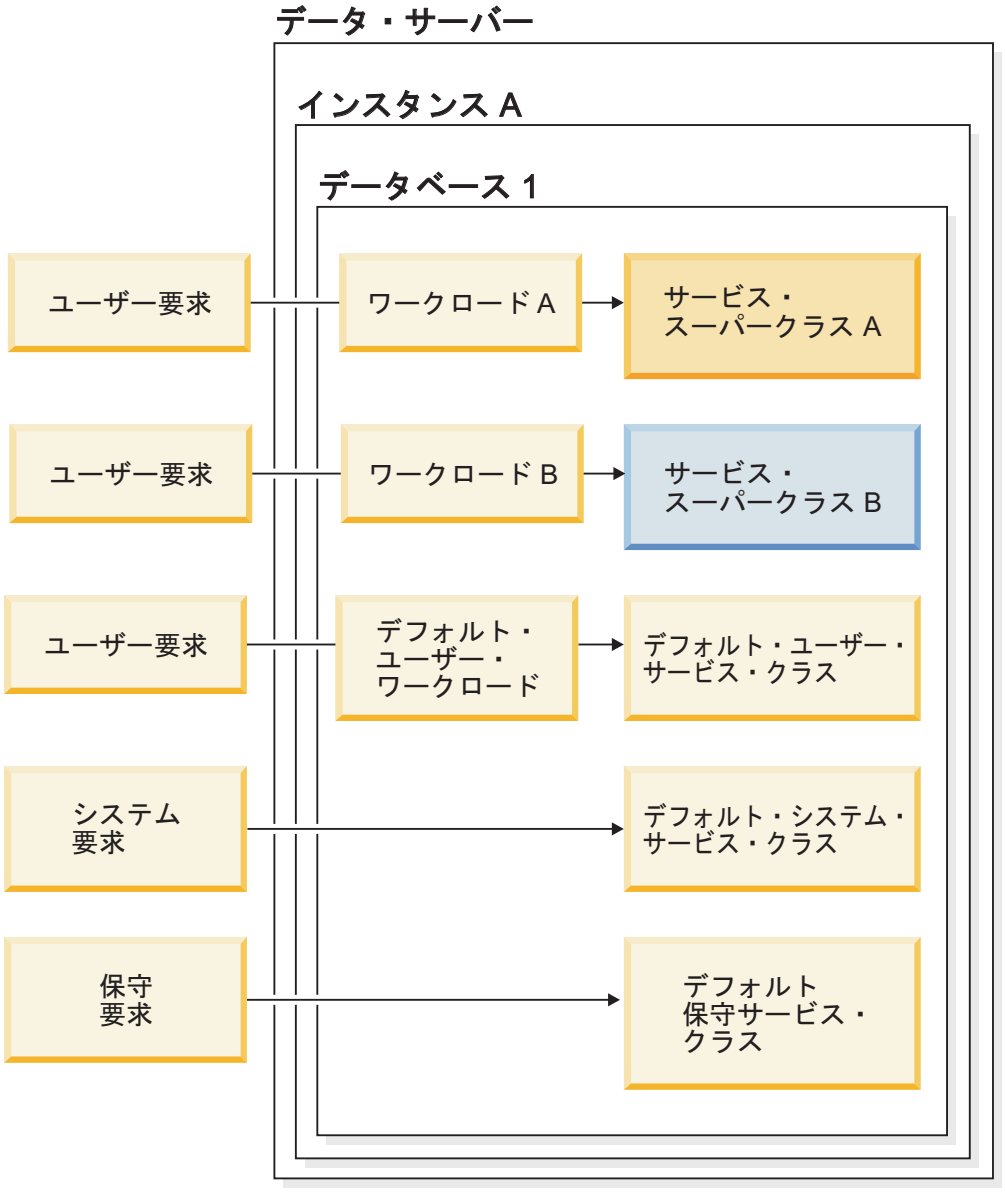


図 34. データ・サーバーの構成: 複数のスーパークラス

CPU リミットと複数のスーパークラス: シナリオ 1

この例では、 232 ページの図 35パネル A に示すように、サービス・クラス A には 30% の CPU リミットがあり、サービス・クラス B には CPU リミットがありません。このシナリオの最初の時点で、サービス・クラス A には少なくとも CPU リソースの使用率を 30% にするのに十分な作業があり、サービス・クラス B には少なくとも CPU リソースの使用率を 70% にするのに十分な作業があります。サービス・クラス A と B には、いずれも 1000 のソフト CPU シェアがあります。

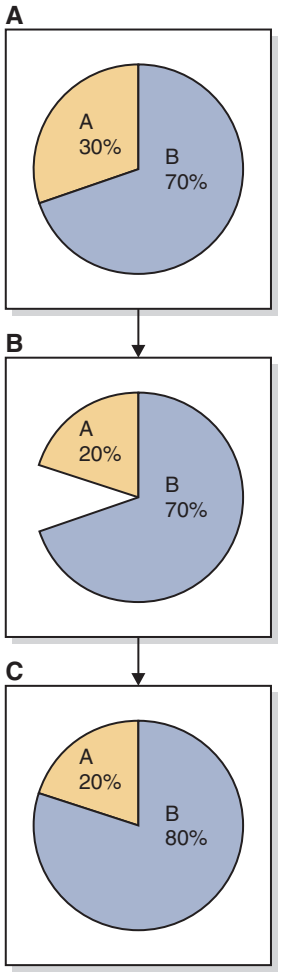


図 35. CPU リミットと複数のスーパークラス: シナリオ 1

パネル B に示すように、サービス・クラス A は、CPU リソースの要求が 30% から 20% に減少しています。パネル C に示すように、サービス・クラス B の CPU リソース所要量は、サービス・クラス A によって一時的に解放された CPU リソースを要求するのに十分な量を上回っています。

CPU リミットと複数のスーパークラス: シナリオ 2

この例では、 233 ページの図 36パネル A に再度示すように、サービス・クラス A には 30% の CPU リミットがあり、サービス・クラス B には CPU リミットがありません。このシナリオの最初の時点で、サービス・クラス A には少なくとも CPU リソースの使用率を 30% にするのに十分な作業があり、サービス・クラス B には少なくとも CPU リソースの使用率を 70% にするのに十分な作業があります。サービス・クラス A と B には、いずれも 1000 のソフト CPU シェアがあります。

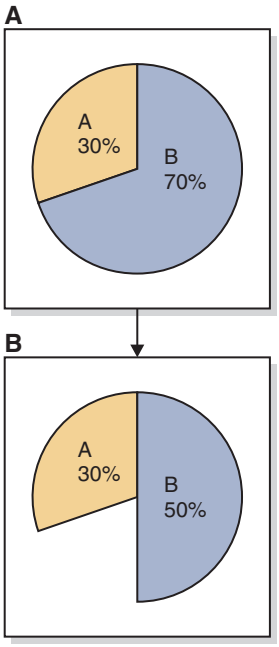


図 36. CPU リミットと複数のスーパークラス: シナリオ 2

パネル B に示すように、サービス・クラス B は、CPU リソースの要求が 70% から 50% に減少しています。 CPU リミットにより、サービス・クラス A は、サービス・クラス B が一時的に解放した CPU リソースの 20% を消費できません。ホストまたは LPAR の CPU 使用率の合計は 80% のままです。

CPU リミットと複数のスーパークラス: シナリオ 3

この例では、 234 ページの図 37パネル A に再度示すように、サービス・クラス A には 30% の CPU リミットがあり、サービス・クラス B には CPU リミットがありません。このシナリオの最初の時点で、サービス・クラス A には少なくとも CPU リソースの使用率を 30% にするのに十分な作業があり、サービス・クラス B には少なくとも CPU リソースの使用率を 70% にするのに十分な作業があります。サービス・クラス A と B には、いずれも 1000 のソフト CPU シェアがあります。

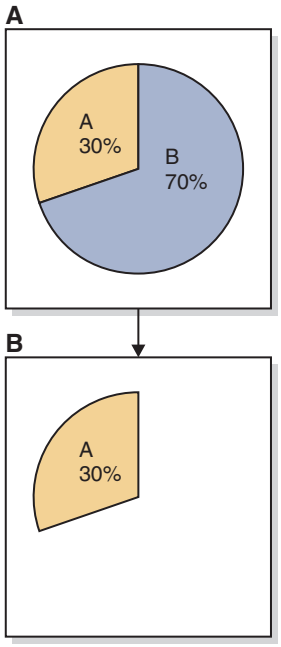


図 37. CPU リミットと複数のスーパークラス: シナリオ 3

パネル B に示すように、サービス・クラス B は、CPU リソースの要求が 70% から 0% に減少しています。 CPU リミットにより、サービス・クラス A は、サービス・クラス B が一時的に解放した CPU リソースの 70% を消費できません。ホストまたは LPAR の CPU 使用率の合計は 30% のままです。

CPU リミットと複数のサブクラス

この使用法の例の次のセットは、複数のサブクラスがある環境での CPU リミットの動作について説明するものです。

235 ページの図 38は、2 つのスーパークラス A と B で構成されているホストまたは LPAR を示します。サービス・スーパークラス A の内部は、サービス・サブクラス A1 と A2 です。基本的な概念を説明するのに役立てる図の目的のために、デフォルトのユーザー、保守、およびシステムのサービス・クラスで実行されている無視できる作業があると仮定します。以下のシナリオでは、1 つのデータベースを持つ 1 つの DB2 インスタンスしかなく、このホストまたは LPAR に 1 つのロジカル・パーティションしかありません。

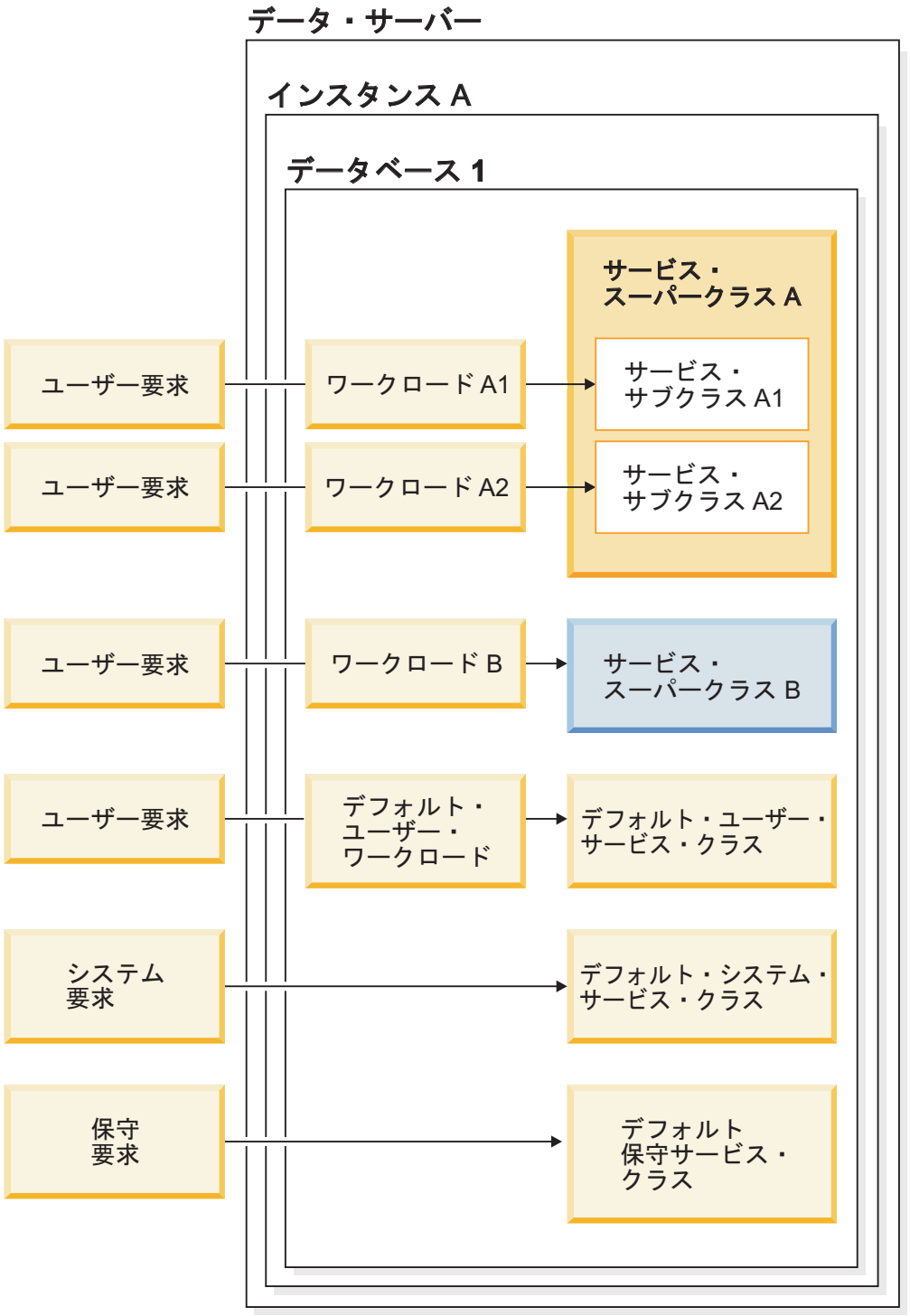


図 38. データ・サーバーの構成: 複数のサブクラス

CPU リミットと複数のサブクラス: シナリオ 1

この例では、 236 ページの図 39パネル A に示すように、サービス・スーパークラス A には 50% の CPU リミットがあり、サービス・サブクラス A1 には 20% の CPU リミットがあります。サービス・スーパークラス B には CPU リミットがありません。このシナリオの最初の時点で、サービス・サブクラス A1 には少なくとも CPU リソースの使用率を 20% にするのに十分な作業があり、サービス・サブク

ラス A2 には少なくとも CPU リソースの使用率を 30% にするのに十分な作業があり、サービス・スーパークラス A の CPU リソース使用率は合計 50% になっています。サービス・スーパークラス B には少なくとも CPU リソースの使用率を 50% にするのに十分な作業があります。サービス・スーパークラス A と B には、いずれも 1000 のソフト CPU シェアがあります。

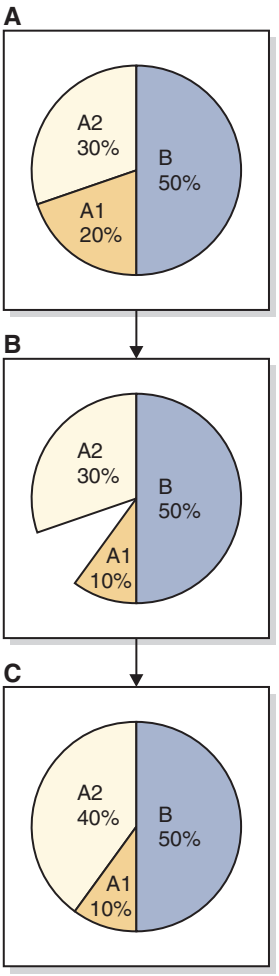


図 39. CPU リミットと複数のサブクラス: シナリオ 1

パネル B に示すように、サービス・サブクラス A1 は、CPU リソースの要求が 20% から 10% に減少しています。サービス・サブクラス A2 とサービス・スーパークラス B の CPU 所要量は、それぞれ、サービス・クラス A1 が一時的に解放した未使用の CPU リソースの全体量を要求するのに十分な量を上回っています。ただし、パネル C に示すように、サービス・サブクラス A2 に解放された CPU リソースのすべてが割り振られ、CPU 使用率が 30% から 40% に増えています。このような CPU リソース割り振りの結果になったのは、各スーパークラスに割り当てられた 1000 のソフト CPU シェアの結果として、CPU リソースの合計を、50%/50% の等分割によりサービス・スーパークラス A と B が既に共有していることが原因です。

CPU リミットと複数のサブクラス: シナリオ 2

この例では、図 40パネル A に再度示すように、サービス・スーパークラス A の CPU リミットは 50% であり、サービス・サブクラス A1 の CPU リミットは 20% になっています。サービス・スーパークラス B には CPU リミットがありません。このシナリオの最初の時点で、サービス・サブクラス A1 には少なくとも CPU リソースの使用率を 20% にするのに十分な作業があり、サービス・サブクラス A2 には少なくとも CPU リソースの使用率を 30% にするのに十分な作業があり、サービス・スーパークラス A の CPU リソース使用率は合計 50% になっています。サービス・スーパークラス B には少なくとも CPU リソースの使用率を 50% にするのに十分な作業があります。サービス・スーパークラス A と B には、いずれも 1000 のソフト CPU シェアがあります。

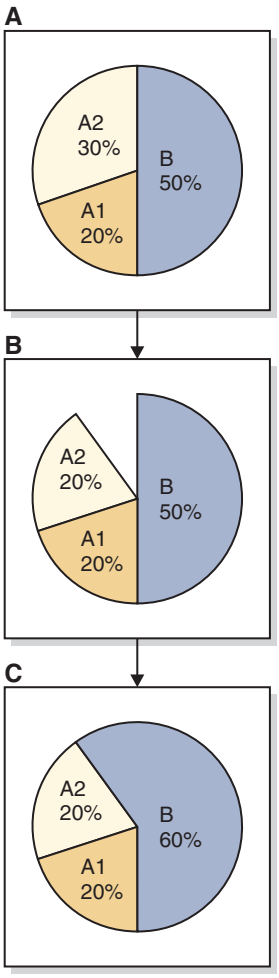


図 40. CPU リミットと複数のサブクラス: シナリオ 2

パネル B に示すように、サービス・サブクラス A2 は、CPU リソースの要求が 30% から 20% に減少しています。サービス・サブクラス A1 の CPU リミットが 20% であるため、サービス・サブクラス A1 において、現在の CPU 使用率の 20% を超えることはできません。サービス・スーパークラス A の CPU 使用率が、合計 40% に減少します。パネル C に示すように、サービス・スーパークラス B の CPU リソース所要量は、サービス・サブクラス A2 によって一時的に解放された CPU リソースを要求するのに十分な量であり、サービス・スーパークラス B の CPU 使用率が 50% から 60% に増えています。

CPU リミットと複数の DB2 インスタンス

この使用法の例の次のセットは、複数の DB2 インスタンスがある環境での CPU リミットの動作を説明するものです。

図 41は、2 つの DB2 インスタンスのインスタンス A とインスタンス B で構成されているホストまたは LPAR を示します。インスタンスにはそれぞれ 1 つのデータベース、データベース 1 とデータベース 2 が含まれています。データベースにはそれぞれ 1 つのサービス・スーパークラス、サービス・スーパークラス A と B が含まれています。サービス・スーパークラス A の内部には、サービス・サブクラス A1 と A2 があります。サービス・スーパークラス B の内部には、サービス・サブクラス B1 と B2 があります。シナリオを単純にするために、各データベースのユーザー要求は、2 つのサービス・サブクラスをターゲットとするもののみであり、デフォルトのサービス・サブクラスをターゲットとする作業はありません。さらに、基本的な概念を説明するのに役立てるために、デフォルトのユーザー、保守、およびシステムのサービス・クラスで実行されている無視できる作業があると仮定します。以下のシナリオでは、各インスタンスに 1 つのデータベースのある 2 つの DB2 インスタンスがあり、このホストまたは LPAR に 1 つのメンバーしかありません。

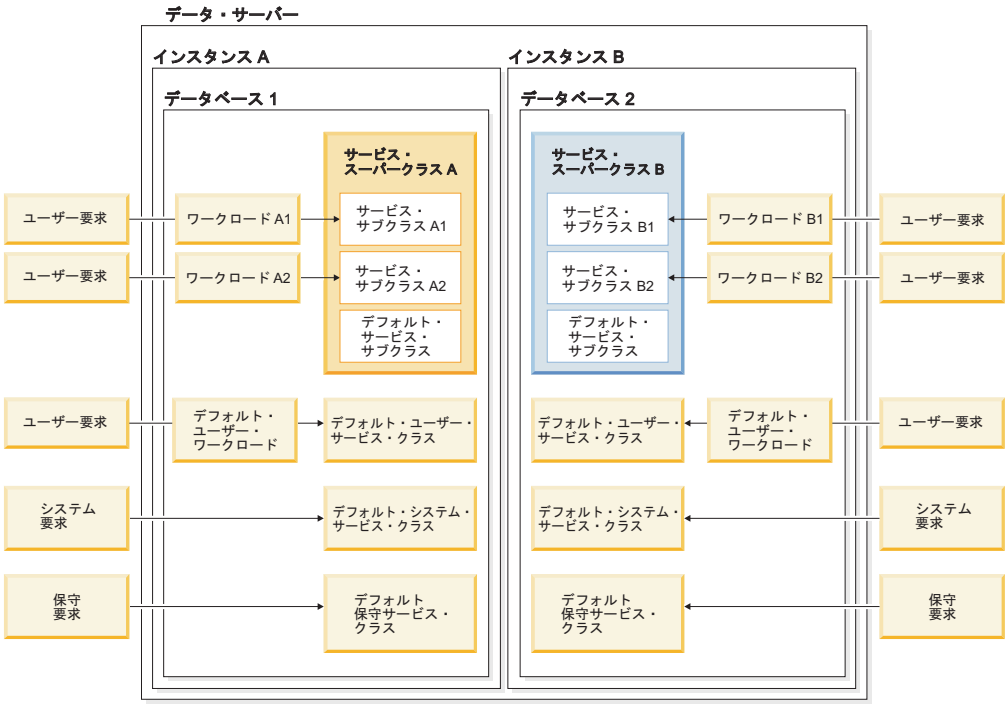


図 41. データ・サーバーの構成: 複数の DB2 インスタンス

CPU リミットと複数の DB2 インスタンス: シナリオ 1

この例では、インスタンス A はホストまたは LPAR の CPU リソースの合計の 50% 以下のみ受け取り、インスタンス B はホストまたは LPAR の CPU リソースの合計の 50% 以下のみ受け取ります。インスタンス A 内では、8000 のソフト CPU シェアを A1 に、2000 のソフト CPU シェアを A2 に割り当てることによって、インスタンスの CPU リソースの 80% をサービス・サブクラス A1 に割り振

り、20% をサービス・サブクラス A2 に割り振ります。サービス・サブクラス A1 が完全な 80% を消費しない場合、インスタンス A 内の CPU 使用率を最大化するために、サービス・サブクラス A2 が未使用の CPU リソースを要求したりその逆の要求がなされたりするようにします。インスタンス B 内では、6000 のソフト CPU シェアを B1 に、4000 のソフト CPU シェアを B2 に割り当てることによって、インスタンスの CPU リソースの 60% がサービス・サブクラス B1 に、40% がサービス・サブクラス B2 に割り振られるようにします。サービス・サブクラス A1 および A2 と同じように、サービス・サブクラス B2 が CPU リソース割り振り全体を使用していない場合にサービス・サブクラス B1 が未使用の CPU リソースを要求するようにしたり、その逆が要求されるようにしたりすることにより、インスタンス B 内の CPU 使用率を最大化します。

前の段落で説明したように、このシナリオに適切な条件を構成するために、データベース 1 のインスタンス A のサービス・スーパークラス A の 50% の CPU リミットを作成し、データベース 2 のインスタンス B のサービス・スーパークラス B の 50% の CPU リミットを作成することで、各インスタンスの CPU リソースを制限します。 2 つのインスタンスにより CPU リソース割り振りのすべてが使用されている場合に、ホストまたは LPAR は全体として CPU 使用率が 100% であると見なされます。どちらかのインスタンスが CPU 割り当ての全体を使用していない場合、もう一方のインスタンスが未使用の CPU リソースを要求することはできません。

各インスタンスの各サービス・スーパークラスに対し、ワークロード管理ディスパッチャーは、相対的なソフト CPU シェアの割り当てを使用して、サブクラス間で利用可能な CPU リソースを分割します。サービス・サブクラスに割り当てられたソフト CPU シェアを使用することによって、インスタンスが CPU リソース割り振り全体を利用しないことになる唯一の状況は、インスタンスの各サブクラスの中で、割り振り CPU 使用率全体を達成するだけの十分な作業が実行されていない場合のみになります。

240 ページの図 42パネル A に示すように、サービス・スーパークラス A は 50% の CPU リミットを持ち、サービス・スーパークラス B も 50% の CPU リミットを持っています。サービス・サブクラス A1 には、インスタンス A の CPU リソースの 80% (ホストまたは LPAR の CPU リソースの 40%) があり、サービス・サブクラス A2 には、残りの 20% (ホストまたは LPAR の CPU リソースの 10%) があります。サービス・サブクラス B1 には、インスタンス B の CPU リソースの 60% (ホストまたは LPAR の CPU リソースの 30%) があり、サービス・サブクラス B2 には、残りの 40% (ホストまたは LPAR の CPU リソースの 20%) があります。このシナリオの最初の時点で、サービス・サブクラス A1 には少なくともホストまたは LPAR の CPU リソースの使用率を 40% にするのに十分な作業があり、サービス・サブクラス A2 には少なくともホストまたは LPAR の CPU リソースの使用率を 10% にするのに十分な作業があり、サービス・スーパークラス A の CPU リソース使用率の合計は 50% になっています。サービス・サブクラス B1 には、少なくともホストまたは LPAR の CPU リソースの使用率を 30% にするのに十分な作業があり、サービス・サブクラス B2 には、少なくともホストまたは LPAR の CPU リソースの使用率を 20% にするのに十分な作業があり、サービス・スーパークラス B の CPU リソース使用率の合計は 50% になっています。

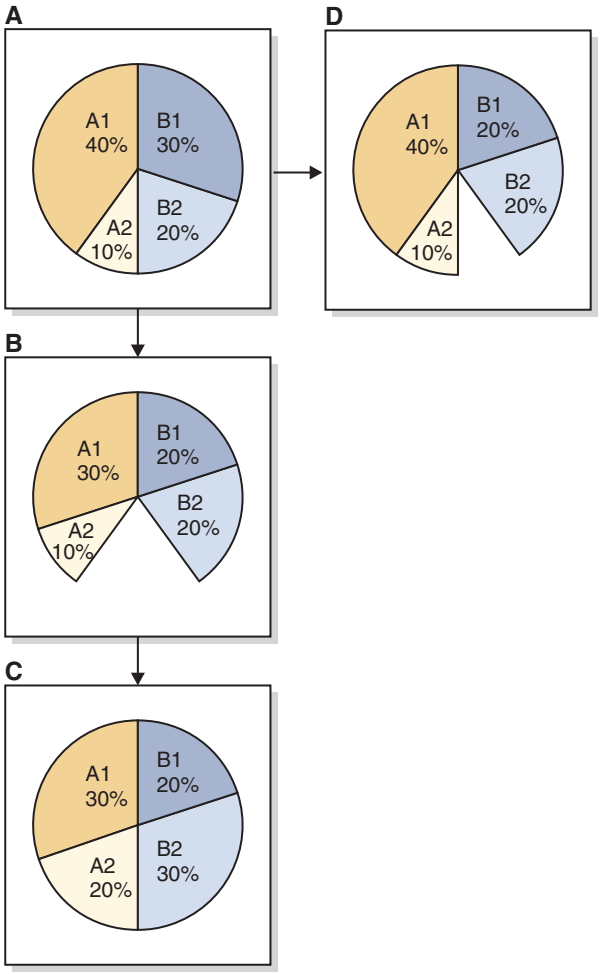


図 42. CPU リミットと複数の DB2 インスタンス: シナリオ 1

パネル B に示すように、サービス・サブクラス A1 は、CPU リソースの要求が 40% から 30% に減少しており、サービス・サブクラス B1 は、CPU リソースの要求が 30% から 20% に減少しています。パネル C に示すように、サービス・サブクラス A2 は、利用可能になった未使用の CPU リソースを要求するのに十分な作業がその中で実行されていると仮定すると、A2 にソフト CPU シェアが割り当てられているため、A2 の CPU 使用率は 10% から 20% に増えます。同じことがサービス・サブクラス B2 にも当てはまり、パネル C に示すように、CPU 使用率が 20% から 30% に増えます。

今度は、別の例のためにパネル A で説明している元の開始条件を考えてみましょう。パネル D に示すように、サービス・サブクラス B1 が CPU の要求を 30% から 20% に減らし、サービス・サブクラス B2 が、20% の CPU の要求を超えるのに十分な作業をその中で実行していない場合、インスタンス B のサービス・スーパークラス B は、CPU 割り当て全体のうち 50% を使用せず、40 % の CPU 使用率のままになります。結果として、ホストまたは LPAR の CPU 使用率は合計 CPU リソースのうちの 90% のみになります。

CPU リミットと複数の DB2 インスタンス: シナリオ 2

このシナリオの例では、サービス・サブクラスの CPU リミットの合計が、親サービス・スーパークラスの CPU リミットを超過しない場合に何が起こるのかについて説明します。

『CPU リミットと複数の DB2 インスタンス: シナリオ 1』と類似した初期条件を使用して、図 43パネル A に示すように、サービス・サブクラス A1 と A2 に割り当てられたソフト CPU シェアを、ホストまたは LPAR の CPU リソースの合計のそれぞれ 40% と 10% の CPU リミットに変更します。両方のサービス・サブクラスが最大で割り当てられた CPU リミットまで CPU リソースを使用する場合、サービス・スーパークラス A の CPU 使用率の合計は 50% であり、サービス・スーパークラス A の 50% という CPU リミットの付加的な制約は冗長なものとなります。

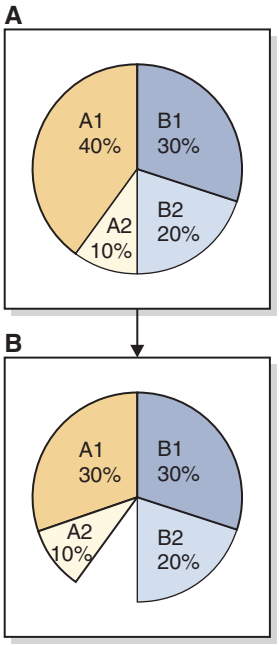


図 43. CPU リミットと複数の DB2 インスタンス: シナリオ 2

パネル B に示すように、サービス・サブクラスで実行されている作業の量が減少したため、サービス・サブクラス A1 は、CPU リソースの要求が 40% から 30% に減少しています。この状況でディスパッチャーは、サービス・サブクラス A1 によって一時的に解放された未使用の CPU リソースを、サービス・サブクラス A2 に割り振ることができません。サービス・サブクラス A2 は、ホストまたは LPAR の CPU リソースの 10% の CPU リミットで、ワークロードの実行を継続します。この状態でインスタンス A が 50% の CPU リソース割り振りの全体を使用することはできません。

CPU リミットと複数の DB2 インスタンス: シナリオ 3

このシナリオの例では、サービス・サブクラスの CPU リミットの合計が、親サービス・スーパークラスの CPU リミットを超過する場合に何が起こるのかについて説明します。

『CPU リミットと複数の DB2 インスタンス: シナリオ 1』と類似した初期条件を使用して、サービス・サブクラス A1 と A2 に割り当てられたソフト CPU シェアを、ホストまたは LPAR の CPU リソースの合計のそれぞれ 40% と 40% の CPU リミットに変更します。この例では、サービス・サブクラス A1 と A2 に割り当てられた CPU リミットの合計は、サービス・スーパークラス A に割り当てられた 50% の CPU リミットを超える 80% です。ワークロード管理ディスパッチャーは、サービス・スーパークラス A が 50% の CPU リミットを超えないようにします。スーパークラス A の各サービス・サブクラスに割り振られた CPU リソースの量は、サブクラスに割り当てられた CPU シェアによって決定されます。 CPU シェアは、サービス・サブクラス A1 と A2 に対して明示的に割り当てられてはいませんが、これらのサブクラスにはそれぞれ、サブクラスが作成された時点で割り当てられた 1000 のソフト CPU シェアがあり、各サブクラスに均等の CPU 割り当てとなっています。ディスパッチャーは、サービス・スーパークラス A に許可されたホストまたは LPAR の CPU リソースの合計の 50% を等しく分割した量をサービス・スーパークラス A に割り振ります。その結果、図 44パネル A に示すように、25% がサービス・サブクラス A1 に割り振られ、25% がサービス・サブクラス A2 に割り振られます。

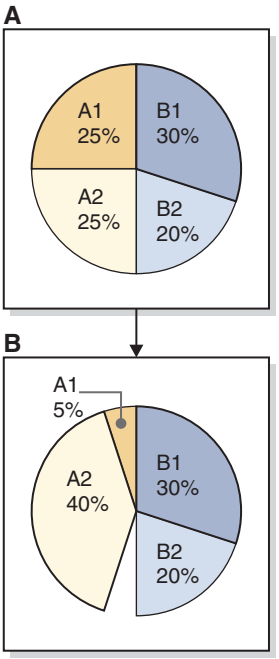


図 44. CPU リミットと複数の DB2 インスタンス: シナリオ 3

パネル B に示すように、サービス・サブクラスで実行されている作業の量が減少したため、サービス・サブクラス A1 は、CPU リソースの要求が 25% から 5% に減少しています。ソフト CPU シェアの割り当てにより、ディスパッチャーは、サービス・サブクラス A1 によって一時的に解放された未使用の CPU リソースを、サービス・サブクラス A2 によって要求された 40% まで割り振ることができます。サービス・サブクラス A2 は、ホストまたは LPAR の CPU リソースの 40% の CPU リミットを超えることはできません。この状態でインスタンス A が 50% の CPU リソース割り振りの全体を使用することはできません。

注: 同じホストまたは LPAR の複数のインスタンス内の作業を管理する、このアプローチの制限として、各インスタンスにつきサービス・スーパークラスが 1 つに制限されるというものがあります。 DB2 ワークロード管理を統合可能なワークロード・マネージャーのあるオペレーティング・システムでは、各インスタンスの DB2 サービス・クラスをオペレーティング・システム (OS) WLM (AIX WLM や Linux WLM など) のサービス・クラスにマップし、各 OS のサービス・クラスに OS WLM のハードの制限を割り当て、各インスタンスの CPU リソースの使用率に上限を設けるという、代替手段もあります。

その他の情報

以下のワークロード管理ディスパッチャーの主題について、完全な詳細が提供されています。

- ワークロード管理ディスパッチャーについては、 190 ページの『ワークロード管理ディスパッチャー』を参照してください。
- ディスパッチ並行性レベルについては、 249 ページの『ディスパッチ並行性レベル』を参照してください。
- アクティブと見なされるサービス・クラスの CPU 使用率の最小レベルについては、 245 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率』を参照してください。
- ハード CPU シェアについては、 214 ページの『ハード CPU シェア』を参照してください。
- ソフト CPU シェアについては、 222 ページの『ソフト CPU シェア』を参照してください。
- ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニングについては、 251 ページの『ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング』を参照してください。

CPU リミットの設定

CPU リミット属性を設定するには、CREATE SERVICE CLASS ステートメントを使って新しいサービス・クラスと CPU リミットを作成します。既存のサービス・クラスの CPU リミットの設定を変更するには ALTER SERVICE CLASS ステートメントを使用します。 CPU リミットを設定すると、ワークロード管理ディスパッチャー制御中のあらゆる状況で、サービス・クラスによる CPU 使用量に上限が設けられます。

始める前に

ワークロード管理ディスパッチャーをまだ使用可能にしていない場合は、**wlm_dispatcher** データベース・マネージャー構成パラメーターを構成することにより、これを使用可能にする必要があります。ワークロード管理ディスパッチャーを使用可能にすることによって、CPU リミットの属性を使用した CPU リソースの制御がデフォルトで利用可能になります。 212 ページの『ワークロード管理ディスパッチャーを使用可能にする』を参照してください。

このタスクについて

このタスクでは、コマンド行プロセッサーで CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスの CPU リミットを設定します。

制約事項

ワークロード管理ディスパッチャーが制御できるサービス・クラス、つまりユーザー・サービス・クラスおよび保守サービス・クラスにのみ、CPU リミットを割り当てることができます。システム・サービス・クラスに関する CPU リソース割り振りをワークロード管理ディスパッチャーで制御することはできません。

手順

新しいサービス・クラスを作成して CPU リミットを設定するには、次のようにします (既存のサービス・クラス CPU リミットの設定を変更する場合は、スキップしてステップ 2 に進んでください)。

1. CREATE SERVICE CLASS ステートメントを発行して新しいサービス・クラスを作成し、CPU リミットを (この例では) 25%に設定します。

   ```
   create service class service-class-name cpu limit 25
   ```

既存のサービス・クラス CPU リミットの設定を変更するには、次のようにします。

2. ALTER SERVICE CLASS ステートメントを発行して、CPU リミットを (この例では) 50% に変更します。

   ```
   alter service class service-class-name cpu limit 50
   ```

タスクの結果

これで、指定したサービス・クラスの CPU リミットが設定または変更されました。ワークロード管理ディスパッチャーは、設定された制限を超えてこのサービス・クラスに CPU リソースを割り振ることができません。

次のタスク

ワークロード管理ディスパッチャーで CPU リソースを管理できるようにして、CPU リミットを設定した後、以下のタスクを完了することを考慮してください。

- **wlm_disp_concur** データベース・マネージャー構成パラメーターを構成することにより、ディスパッチ並行性レベルを設定します。 250 ページの『ディスパッチ並行性レベルの設定』を参照してください。
- **wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターを構成することにより、ハードおよびソフト CPU シェアを有効にします。 226 ページの『CPU シェアの使用可能化および設定』のステップ 1 を参照してください。
- CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスを作成または変更し、ハードまたはソフト CPU シェアを構成します。 226 ページの『CPU シェアの使用可能化および設定』を参照してください。

- **wlm_disp_min_util** データベース・マネージャー構成パラメーターを構成することにより、ワークロード管理ディスパッチャーでサービス・クラスをアクティブと見なすための最小 CPU 使用率 (パーセンテージ) を設定します。 247 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率の設定』を参照してください。

アクティブと見なされるサービス・クラスの最小 CPU リソース使用率

データベース・マネージャーの構成パラメーター **wlm_disp_min_util** を設定することによって、サービス・クラスが実行中の作業にアクティブに従事しているとワークロード管理ディスパッチャーが見なす CPU リソースの使用率の最小レベルを制御できます。アクティブなサービス・クラスのみの CPU シェアが、ディスパッチャーによって実行される CPU リソースの割り振りのスケジューリングに計算されます。

共有ベースの CPU リソースの割り振りを管理する際、ワークロード管理ディスパッチャーはサービス・クラスをアクティブと見なし、CPU リソースに、そのサービス・クラスで実行されているデータベースの要求によって使用されている部分があれば、CPU シェアの割り当て全体について全体的な CPU リソースのスケジューリングの割り振りを計算します。特定のインスタンスでは、CPU リソースのスケジューリング中に、サービス・クラスの CPU シェアを含めるために、ワークロード管理ディスパッチャーでどれ位のサービス・クラス生成の CPU アクティビティーが必要なのかをある程度制御することをお勧めします。

246 ページの図 45 パネル A で示されているサンプルのシナリオでは、サービス・クラス A には、営業日中にのみ実行される優先順位の高いトランザクション・ワークロードが含まれ、サービス・クラス B と C には、進行中の昼と夜の、優先順位の低いバッチ・ジョブが含まれます。ハード CPU シェアをサービス・クラス B と C に割り当てることによって、サービス・クラス A は中断から保護されます。(パネル B に示すように) サービス・クラス A のトランザクション・ワークロードが存在しない夜間には、サービス・クラス B と C は CPU リソースを完全に利用できるため、(パネル C に示すように) 進行が速くなります。

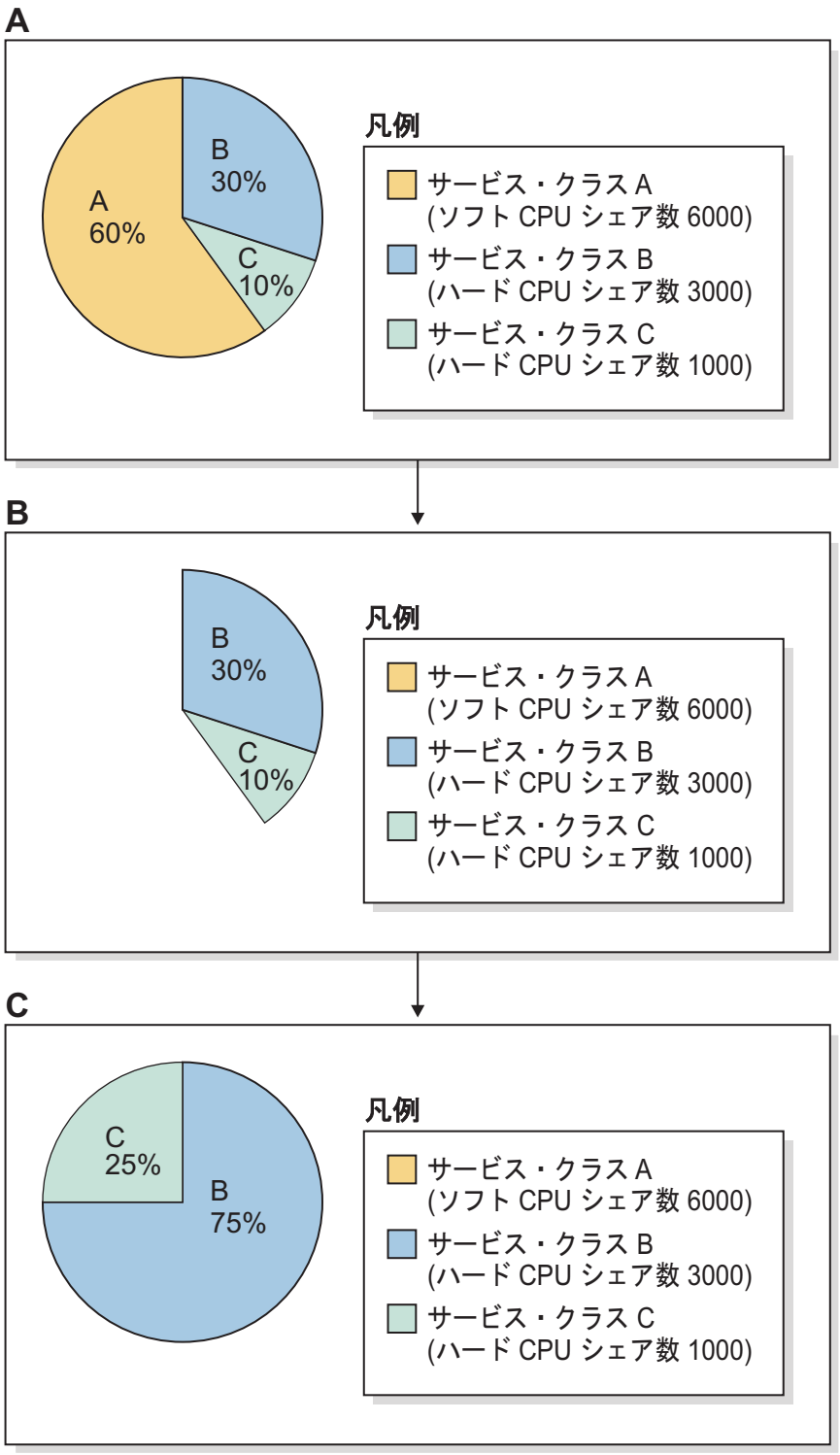


図 45. 最小 CPU 使用率の例: ハード CPU シェアとソフト CPU シェア

小さく断続的に続くトランザクションの作業がサービス・クラス A で夜間に継続して発生している場合にどうなるかを考慮してみましょう。この場合、サービス・クラス A はワークロード管理ディスパッチャーによってアクティブと見なされ、サービス・クラス A によって一時的に解放された未使用の CPU リソースは、サービス・クラス B と C で利用できません。サービス・クラス B と C の CPU リソー

ス割り振りは、パネル B のサービス・クラス A のアクティビティーを表している CPU 使用率の円グラフで、細長い部分で示されています (パネル B の円グラフには示されていません)。パネル C で示された元のシナリオでの進行より、サービス・クラス B と C の結果の夜間の進行の方がさらに遅くなります。

ホストまたは LPAR でサービス・クラスがアクティブと見なされるのと同じ、またはそれを超えた CPU 使用率パーセントを設定するオプションによって、DB2 ワークロードを管理するための最大の柔軟性がユーザーに提供されます。サービス・クラスが非アクティブと見なされる場合、CPU シェアの割り当ては、CPU リソース割り振りとして計算されないため、特にハード CPU シェアが割り当てられたサービス・クラスが、未使用の CPU リソースを要求することができます。この CPU 使用率の最小パーセントは、データベース・マネージャーの構成パラメーター **wlm_disp_min_util** のパーセント値を構成することによって指定されます。この構成パラメーターは、0 から 100 までの間でパーセント値を設定でき、デフォルト値は 5 です。この構成パラメーターの設定を有効にするには、**wlm_dispatcher** データベース・マネージャー構成パラメーターの値を YES に設定することによって、ワークロード管理ディスパッチャーを使用可能にする必要があります。

上で説明した小さく断続的に続く作業のシナリオをもう一度考えてみます。データベース・マネージャーの構成パラメーター **wlm_disp_min_util** のパーセント値を、今度はサービス・クラス A の夜間の CPU 使用率の小さな細長い部分よりわずかに高い値に設定したため、サービス・クラス B と C の夜間のバッチ・ジョブの進行が改善されるとみなすことができ、パネル C で示されていた CPU 使用率の元のシナリオと近くなります。

その他の情報

以下のワークロード管理ディスパッチャーの主題について、完全な詳細が提供されています。

- ワークロード管理ディスパッチャーについては、190 ページの『ワークロード管理ディスパッチャー』を参照してください。
- ディスパッチ並行性レベルについては、249 ページの『ディスパッチ並行性レベル』を参照してください。
- ハード CPU シェアについては、214 ページの『ハード CPU シェア』を参照してください。
- ソフト CPU シェアについては、222 ページの『ソフト CPU シェア』を参照してください。
- CPU リミットについては、228 ページの『CPU リミット』を参照してください。
- ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニングについては、251 ページの『ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング』を参照してください。

アクティブと見なされるサービス・クラスの最小 CPU リソース使用率の設定

wlm_disp_min_util データベース・マネージャー構成パラメーターの値をパーセンテージに設定することにより、サービス・クラスの最小 CPU 使用率を設定します。このパーセンテージ以上の割合で CPU リソースを使用しているサービス・ク

ラスが、ホストまたはロジカル・パーティション (LPAR) 上でアクティブ状態と見なされます。アクティブなサービス・クラスの CPU シェアが、CPU 割り振り計算に組み込まれます。

手順

アクティブと見なされるサービス・クラスの最小 CPU 使用率パーセンテージを設定するには、次のようにします。

次のようにコマンド行プロセッサー (CLP) を使って **UPDATE DATABASE MANAGER CONFIGURATION** (または **UPDATE DBM CFG**) コマンドを発行し、**wlm_disp_min_util** データベース・マネージャー構成パラメーターの値を 6% に設定します。最初に DB2 インスタンスに接続することで、パラメーター値が直ちに更新されます。

```
attach to instance-name
update dbm cfg using wlm_disp_min_util 6
detach
```

タスクの結果

アクティブと見なされるサービス・クラスの最小 CPU 使用率パーセンテージの設定が完了しました。 CPU 使用率パーセントがこの最小値より小さいサービス・クラスはアイドル状態と見なされ、それらの CPU 割り振りは、予備 CPU リソースを要求できるサービス・クラスに対して比例的に再割り振りされる可能性があります。

次のタスク

サービス・クラスがワークロード管理ディスパッチャーによってアクティブと見なされるための最小 CPU 使用率パーセンテージを設定した後、以下のタスクを完了することを考慮してください。

- 最小 CPU 使用率の設定を有効にするには、**wlm_dispatcher** データベース・マネージャー構成パラメーターを構成することでワークロード管理ディスパッチャーを使用可能にする必要があります。 212 ページの『ワークロード管理ディスパッチャーを使用可能にする』を参照してください。
- **wlm_disp_concur** データベース・マネージャー構成パラメーターを構成することにより、ディスパッチ並行性レベルを設定します。 250 ページの『ディスパッチ並行性レベルの設定』を参照してください。
- CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスを作成または変更し、CPU リミットを構成します。 243 ページの『CPU リミットの設定』を参照してください。
- **wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターを構成することにより、ハードおよびソフト CPU シェアを有効にします。 226 ページの『CPU シェアの使用可能化および設定』のステップ 1 を参照してください。
- CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスを作成または変更し、ハードまたはソフト CPU シェアを構成します。 226 ページの『CPU シェアの使用可能化および設定』を参照してください。

ディスパッチ並行性レベル

ワークロード管理ディスパッチャーは、データベース・マネージャーがオペレーティング・システム (OS) に同時にディスパッチさせることが可能なスレッド数を制御します。この数は、ディスパッチ並行性レベルと呼ばれます。ディスパッチ並行性レベルは、データベース・マネージャーの構成パラメーター **wlm_disp_concur** を使用して設定できます。詳細がここで提供されています。

フィーチャーおよび機能

ワークロード管理ディスパッチャーが管理している同時実行中のエージェントの数は、通常、システムの CPU の数の小さい倍数でなければなりません。その目標は、確立する並行性レベルを、ホストまたは LPAR の CPU リソースが完全に利用されるのに十分な大きさで、かつ大きすぎず、その一方で、スレッドがアクティブから非アクティブに切り替わる際、あるいはその逆に切り替わる際の OS のスケジューリング待ち時間を取るだけの余裕のあるものにすることです。この最適値によって、最大の効率が保証され、CPU リソース割り振りに対する最大の制御をワークロード管理ディスパッチャーに与えます。並行性の値が小さすぎる場合は、利用可能な CPU リソースを完全に利用するための、十分な作業が割り当てられません。また、並行性の値が大きすぎる場合は、ディスパッチャーはシステムで実行中である作業に対して、制御可能な余地がなくなり、リソースの競合が多発することになります。

データベース・マネージャーの構成パラメーター **wlm_disp_concur** を設定することによって、DB2 データベース・マネージャー自体が並行性レベルを設定する方法を指定するか (COMPUTED)、または並行性レベルを固定値に手動で設定できます。

一般ガイドラインとして、最適なディスパッチ並行性レベルは、サーバーのハードウェアの利用できる CPU の数の 4 倍です。このガイドラインは、通常、ほとんどのプラットフォームに対して十分当てはまり、デフォルトとして取得するレベルです (COMPUTED は、構成パラメーターのデフォルトの設定値です)。ディスパッチ並行性レベルの設定方法の完全な詳細については、250 ページの『ディスパッチ並行性レベルの設定』を参照してください。

wlm_disp_concur データベース構成パラメーターの指定された値は、各ホストまたは LPAR に適用されます。マルチメンバー・データベース環境では、指定された並行性レベルは、メンバー間で共有されます。

チューニング

ワークロード管理ディスパッチャーを使用可能にした後で、スループットと CPU 使用率のドロップについてシステムをモニターします。ドロップがいずれかのメトリックで発生している場合、構成パラメーターの値を、利用可能なプロセッサーの数の増分で増やします。

その他の情報

以下の主題について、完全な詳細が提供されています。

- ワークロード管理ディスパッチャーについては、190 ページの『ワークロード管理ディスパッチャー』を参照してください。

- ソフト CPU シェアについては、 222 ページの『ソフト CPU シェア』を参照してください。
- ハード CPU シェアについては、 214 ページの『ハード CPU シェア』を参照してください。
- CPU リミットについては、 228 ページの『CPU リミット』を参照してください。
- アクティブと見なされるサービス・クラスの CPU 使用率の最小レベルについては、 245 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率』を参照してください。
- ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニングについては、 251 ページの『ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング』を参照してください。

ディスパッチ並行性レベルの設定

wlm_disp_concur データベース・マネージャー構成パラメーターの値を COMPUTED に設定するか、特定の値を設定することにより、ディスパッチ並行性レベルを設定します。ディスパッチ並行性レベルを最適な値に設定してください。つまり最大の効率を実現し、しかも WLM ディスパッチャーで CPU リソース割り振りを最大限に制御できるようにする値です。

手順

ディスパッチ並行性レベルを設定するには、次のようにします。

コマンド行プロセッサー (CLP) を使って **UPDATE DATABASE MANAGER CONFIGURATION** (または **UPDATE DBM CFG**) コマンドを発行し、**wlm_disp_concur** データベース・マネージャー構成パラメーターの値を以下のいずれか 1 つに設定します。

- COMPUTED (デフォルト)
- *manually_set_value*

以下の例ではディスパッチ並行性レベルを 16 に設定します (システムで使用可能な CPU 数 4 の 4 倍)。最初に DB2 インスタンスにアタッチすることにより、設定が直ちに更新されます。

```
attach to instance-name
update dbm cfg using wlm_disp_concur 16
detach
```

タスクの結果

ディスパッチ並行性レベルが設定されました。これを COMPUTED に設定した場合、並行性レベルは DB2 データベース・マネージャーによって決定されます。手操作で設定した場合は、並行性レベルを最適な値に調整する必要が生じることがあります。つまり最大の効率を実現し、しかもワークロード管理ディスパッチャーで CPU リソース割り振りを最大限に制御できるようにする値です。

次のタスク

ディスパッチ並行性レベルを設定した後、以下のタスクを完了することを考慮してください。

- ディスパッチ並行性レベル設定を有効にするには、**wlm_dispatcher** データベース・マネージャー構成パラメーターを構成することにより、ワークロード管理ディスパッチャーを使用可能にする必要があります。 212 ページの『ワークロード管理ディスパッチャーを使用可能にする』を参照してください。
- CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスを作成または変更し、CPU リミットを構成します。 243 ページの『CPU リミットの設定』を参照してください。
- **wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターを構成することにより、ハードおよびソフト CPU シェアを有効にします。 226 ページの『CPU シェアの使用可能化および設定』のステップ 1 を参照してください。
- CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを使ってサービス・クラスを作成または変更し、ハードまたはソフト CPU シェアを構成します。 226 ページの『CPU シェアの使用可能化および設定』を参照してください。
- **wlm_disp_min_util** データベース・マネージャー構成パラメーターを構成することにより、ワークロード管理ディスパッチャーでサービス・クラスをアクティブと見なすための最小 CPU 使用率 (パーセンテージ) を設定します。 247 ページの『アクティブと見なされるサービス・クラスの最小 CPU リソース使用率の設定』を参照してください。

ワークロード管理ディスパッチャーを使用可能にしてディスパッチ並行性レベルを設定した後、システムをモニターして、スループットおよび CPU 使用率が低下するかどうかを確認します。いずれかのメトリックが低下した場合、**wlm_disp_concur** 構成パラメーターの値を、使用可能なプロセッサー数単位で増やしてください。ワークロード管理ディスパッチャーのパフォーマンスのモニターとチューニングについて、詳しくは、『ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング』を参照してください。

ワークロード管理ディスパッチャーのパフォーマンスのモニターおよびチューニング

ワークロード管理ディスパッチャーのパフォーマンスのモニターとチューニングは、DB2 データベース・マネージャーが提供する表関数とモニター・エレメントによって実現できます。ここでは、詳細について説明します。

概要

ワークロード管理ディスパッチャーのパフォーマンスをモニターしてチューニングし、最適な結果を実現するには、適切なツールが必要です。表関数とモニター・エレメントが提供され、ディスパッチャーのパフォーマンスのモニターに役立てることができます。ここで説明するように、収集されたモニター・データを分析した後で、ディスパッチャーのパフォーマンスをチューニングするためにサービス・クラスの CPU シェアと CPU リミットを調整した後で、ディスパッチャー並行性レベルを調整するか、または CPU 割り当てを再配布することができます。

以下のセクションでは、考慮すべきワークロードのタイプについて説明します。これは、分析するための適切なデータを提供するためにどのようにモニターするのが最もよいかが異なるためです。また、考慮されているワークロードの特定のタイプに最も適切なパフォーマンスの測定についても説明します。

ワークロードのタイプ

ディスパッチャーの構成をチューニングして、システムから可能な限り最もよいワークロードのパフォーマンスを得るという目的のためのパフォーマンスの測定の観点から、考慮すべきワークロードのタイプは、バッチとトランザクションの 2 つです。ワークロードの各タイプには、そのタイプのワークロードで実行されているシステムのパフォーマンスがどの程度かを判別するのに最も適しているパフォーマンスの特性的な測定値があります。システムで起こっているワークロードのタイプを最もよく特徴づけるパフォーマンスの測定値を使用します。

バッチ

バッチ・ワークロードには、データベースに接続する 1 つ以上のアプリケーションがあり、各アプリケーションは、アクティビティーやトランザクションを休止することなく次々にサブミットします。このワークロードのパフォーマンスの最も重要な測定値は、アクティビティーまたはトランザクションのセット全体が完了される速さです。データベース・マネージャーの処理速度は、アクティビティーまたはトランザクションのセット全体が完了される速さの主要な決定要素です。

トランザクション

トランザクション・ワークロードのユーザーは、端末において、アクティビティーまたはトランザクションをデータベースにサブミットして、応答を待ち、応答を分析して、フォローアップのアクティビティーまたはトランザクションをサブミットするかどうかを決定します。このタイプのワークロードでは、パフォーマンスの最も重要な測定は、ユーザーが個別の結果を受け取る速さです。システムの各個別のユーザーに対して、単一のアクティビティーまたはトランザクションを処理するデータベース・マネージャーの処理速度は、平均的なユーザーが個別の結果を受け取る速さの主要な決定要素です。データベース・マネージャーが一定期間にユーザーからのすべてのアクティビティーまたはトランザクションを処理できる速さは、データベース・マネージャーのパフォーマンスよりは、ユーザーの動作に多く依存しているため、関連する測定基準ではありません。

パフォーマンスの測定

以下のパフォーマンスの測定を使用して、特定のタイプのワークロードの下で、システムのパフォーマンスがどの程度かを確認できます。

平均スループット

平均スループットは、単位時間ごとのサービスの完了の平均数です。サービスがトランザクションまたは作業単位 (UOW) である場合、平均の UOW スループットは、単位時間ごとの作業単位の完了数です。通常、1 秒あたりのトランザクションまたは 1 分あたりのトランザクションとして表されます。平均スループットは、測定している作業のタイプがバッチ・ワークロードである場合のシステム・パフォーマンスの測定に役立ちます。

平均アクティビティー・スループットは、単位時間ごとのアクティビティーの完了の平均数です。多くの個別のアクティビティーを含む作業単位のほとんどが長期実行であるシステムでは、UOW スループットを測定することによってワークロード

の進行を測定するより、アクティビティー・スループットを測定することによってワークロード内のアクティビティーの進行を測定する方が容易です。

平均応答時間

平均応答時間は、サービスが要求された時間から、単一のサービスが完了するまでにかかった平均時間数です。サービスがトランザクションまたは作業単位 (UOW) である場合、平均の UOW 応答時間は、UOW が要求された時間から完了するまでにかかった時間数です。平均応答時間は、測定している作業のタイプがトランザクション・ワークロードである場合のシステム・パフォーマンスの測定に役立ちます。平均 UOW 応答時間の近似値として最も近い値は、
MON_SAMPLE_SERVICE_CLASS_METRICS 表関数と
MON_SAMPLE_WORKLOAD_METRICS 表関数、
WLM_GET_SERVICE_SUBCLASS_STATS 表関数と
WLM_GET_WORKLOAD_STATS 表関数、および WLM 統計イベント・モニターで報告される event_scstats および event_wlstats イベント・モニター論理データ・グループから利用できる **uow_lifetime_avg** 統計です。 UOW の存続期間の情報のより高度な形式は、イベント・モニターで利用可能な UowLifetime ヒストグラムで入手できます。

平均アクティビティー応答時間は、アクティビティーが開始された時間から、単一のアクティビティーが結果を返すまでにかかった平均時間数です。平均アクティビティー応答時間の近似値として最も近い値は、
WLM_GET_SERVICE_SUBCLASS_STATS 表関数と
WLM_GET_WORKLOAD_STATS 表関数、および event_scstats および event_wlstats イベント・モニター論理データ・グループから利用できる **coord_act_lifetime_avg** 統計です。この数値は、メンバーごとに測定され、メンバーが非アクティブ化されたとき、または WLM_COLLECT_STATS プロシージャーが呼び出されたときにリセットされます。この数値が近似値の可能性がある理由は、カーソル・アクティビティーという 1 つのタイプのアクティビティーについて、アクティビティー終了前に幾らかの結果が返されることがあり、結果セットの読み取りを終了するのをユーザーに依存し、アクティビティーが完了したと見なされる前にカーソルを閉じるためです。アクティビティーの存続期間のより高度な形式は、イベント・モニターで利用可能な CoordActLifetime ヒストグラムで入手できます。

CPU 使用率

ワークロードのタイプに関係なく、ワークロード管理ディスパッチャーをチューニングする場合に役立つ別の測定基準は、CPU 使用率です。CPU 使用率は、ホストまたは LPAR で CPU リソースがビジー状態である時間の短い一部分です。 CPU 使用率は、ワークロード管理ディスパッチャーが 1 つのサービス・クラスに CPU リソースを割り振るために使用する測定基準です。また、CPU 使用率は、ワークロード管理ディスパッチャーの構成が、意図したとおりに動作しているかどうかを検証するために使用できる測定基準でもあります。
MON_SAMPLE_SERVICE_CLASS_METRICS 表関数と
MON_SAMPLE_WORKLOAD_METRICS 表関数、
WLM_GET_SERVICE_SUBCLASS_STATS 表関数と
WLM_GET_WORKLOAD_STATS 表関数、および WLM 統計イベント・モニターで収集されて報告される event_scstats および event_wlstats イベント・モニター論理デ

ータ・グループを使用して、**uow_throughput**、**uow_lifetime_avg**、および **act_throughput** のモニター・エレメントと同じインターバルで CPU 使用率を測定できます。

注: CPU 使用率の測定が、作成したサービス・クラスで予想と違っていた場合は、デフォルト・ユーザー・サービス・クラスと保守サービス・クラスで実行されているワークロードが存在しないかを確認します。これは、これらのワークロードは、作成したサービス・クラスに明示的に割り当てられないからです。これらのデフォルトのサービス・クラスで実行されているワークロードを含めるのを忘れていた場合は、CPU シェアが最初に使用可能になったときに、それぞれに対してデフォルトで 1000 のハード CPU シェアが割り当てられており、CPU 使用率の測定が予想どおりにならない原因を説明するものとなります。

表関数とイベント・モニターによって報告された CPU 使用率は、ユーザー・サービス・クラスと保守サービス・クラスのみで実行されている作業によって消費された CPU リソースです。ディスパッチャーによって処理されなかった作業は、CPU 使用率としてカウントされません。

ワークロード管理ディスパッチャーによって処理されなかった作業には、以下のものが含まれます。

- DB2 データベース・マネージャーの外部で作業の一部を実行する、DB2 データベース・マネージャー以外のアプリケーションまたはミドルウェア製品によって実行される作業
- DB2 システムのサービス・クラスで実行されるエンティティーによって実行される作業
- 他の DB2 インスタンスによって実行される作業
- fenced ストアード・プロシージャーなど、fenced モード・プロセス (FMP) で実行される DB2 以外のデータベース・マネージャーの作業
- トラステッド・ルーチンで実行される DB2 以外のデータベース・マネージャーの作業

CPU リソースの他のこれらのコンシューマーに対して CPU 使用率を取得するには、OS のワークロード・マネージャーで提供されるようなオペレーティング・システム・レベル (OS レベル) のモニターを使用する必要があります。

CPU 速度

CPU 速度は、リソースに対して競合があるかどうか、およびそのような競合の程度を判別する統計です。リソースへのすべてのアクセスが相互に排他的で同時にそのリソースにアクセスしようとしている複数のリクエスターがある場合、アクセスのための何らかの形式のキューイングが存在する必要があります。存在しない場合は、それらのリクエスターを拒否する必要があります。キューへの形式の設定が許可された場合、リクエスターがリソースを取得して使用を完了するのにかかった時間は、リソースの使用のみに費やされた時間を超過する場合があります。速度は、リソースのみを使用するのに費やされた時間と、リソースの待機と使用の両方に費やされた合計時間の比率です。ゼロから 100% のスケールで測定されます。リソースの競合の量が多い場合、速度はゼロに近くなります。リソースの競合がない場合、キュー時間はありません。 CPU 速度は、最大値の 100% に近くなります。

ワークロード管理ディスパッチャーが使用可能な場合、MON_SAMPLE_SERVICE_CLASS_METRICS 表関数と MON_SAMPLE_WORKLOAD_METRICS 表関数、WLM_GET_WORKLOAD_STATS 表関数と WLM_GET_SERVICE_SUBCLASS_STATS 表関数、および WLM 統計イベント・モニターで収集されて報告される event_scstats および event_wlstats イベント・モニター論理データ・グループを使用して CPU 速度を測定できます。 CPU 速度の値が低い場合、ホストまたは LPAR の CPU リソースで競合が存在していることを示し、ワークロード管理ディスパッチャーが、CPU リソースを優先順位の低いサービス・クラスから優先順位の高いサービス・クラスに移動することが効果的である可能性があることを示しています。CPU 速度が高い場合、それは、CPU リソースの各要求が既に遅延なく処理されているため、ワークロードのパフォーマンスの改善に関してワークロード管理ディスパッチャーの影響は限られたものであることを示しています。

アクティビティーの取り消し

アクティビティーが消費するリソースが多すぎる場合、または実行時間が長すぎる場合は、そのアクティビティーをキャンセルできます。アクティビティーのキャンセルの方が、そのアクティビティーをサブミットしたアプリケーションを強制終了するのに比べ、影響範囲が小さいです。キャンセルされたアクティビティーにより、ユーザーに SQL4725N が戻されますが、接続を終了したり、その他のユーザー・アクティビティーに影響を与えたりはしません。アプリケーションを強制終了すると、接続とユーザー・アクティビティーの両方が終了します。

このタスクについて

アクティビティーを明示的にキャンセルできるのは、コーディネーター・アクティビティーが現在そのアクティビティーの要求を処理している場合のみです。IDLE 状態 (つまり、処理されている要求がない状態) でアクティビティーをキャンセルした場合、アクティビティーは CANCEL_PENDING 状態に置かれ、次に受信される要求でキャンセルされます。例えば、CURSOR アクティビティーをフェッチの間にキャンセルしようとすると、SQL4725N エラーはキャンセルの後の次のフェッチまでユーザーに戻されません。

ロード・ユーティリティーおよびストアード・プロシージャーを含むすべてのユーザー・アクティビティーはキャンセル可能です。

手順

1. キャンセルするアクティビティーを識別します。WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用すると、アプリケーションで実行されているアクティビティーを識別できます。また、アクティビティーが実行している作業を識別するために WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES の情報だけでは不十分な場合、特定のアクティビティーについての詳細をさらに表示するために MON_GET_ACTIVITY_DETAILS_COMPLETE 表関数を使用することもできます。
2. WLM_CANCEL_ACTIVITY ストアード・プロシージャーを使用してアクティビティーをキャンセルします。 ストアード・プロシージャーには、以下の引数をとります。*application_handle*、*uow_id*、および *activity_id*。このストアード・プ

ロシージャーの使用方法は、 394 ページの『シナリオ: 完了に時間がかかり過ぎているアクティビティーの識別』の例を参照してください。

第 4 章 モニターおよび介入

ワークロード管理の 3 番目のドメインはモニターです。モニターは継続的に行う必要があります。

モニターの主な目的は、システム、およびシステム上で実行される個別のワークロードの健全さと効率を確認することです。表関数を使用することより、実行中のワークロード・オカレンスおよびサービス・クラスで実行中のアクティビティーのリスト、平均応答時間など、リアルタイム運用データにアクセスできます。イベント・モニターを使用して、詳細なアクティビティー情報を収集したり、履歴分析のためにアクティビティーの統計を集約したりすることができます。

モニター方針を作成する時の最初のステップとして、通常、集約情報を調べます。集約情報はデータ・サーバー・アクティビティーの全体像を十分に反映していると共に、調べる対象となるすべてのアクティビティーの情報を収集する必要がないため、安上がりです。モニターが必要な領域の範囲が明確になってきたら、さらに詳細な情報を収集できます。

実行可能な標準的モニター・タスクは、以下のとおりです。

- 初期 DB2 ワークロード管理構成の設計を支援するための、システム上のワークロードの分析。
- 以下のタスクを可能にするタイプの操作情報を入手することによる、システムの動作のトラッキングおよび調査。
  - システム・パフォーマンス低下の分析
  - 完了に時間がかかりすぎているアクティビティーの診断
  - エージェント競合の調査
  - パフォーマンスが悪い照会の分離

  アクティビティー、サービス・クラス、ワークロード、ワーク・クラス、しきい値キュー、およびしきい値違反についての情報を入手できます。
- 問題を引き起こすと予想されるキューに入れられているアクティビティーを取り消すことによる実行環境の制御、またはシステムにマイナスの影響を与えると診断したアクティビティーの実行の取り消し。

表関数を使用したリアルタイム・モニター

リアルタイム・モニターのデータには、使用パターンやリソース割り振りの決定、および問題のある領域の識別に役立つ、システムで現在実行されている処理に関する情報と、システムで実行された処理に関する統計およびメトリックが含まれます。こうした操作情報は、DB2 表関数を使用することで取得できます。

名前の先頭が *WLM_* の表関数は、DB2 ワークロード管理の表関数です。これらの表関数を使用すると、ワークロード管理に関するデータの集合 (ワークロード管理統計など) に、SELECT ステートメントを実行できる仮想 DB2 表としてアクセスできます。これにより、あたかもデータ・サーバー上の物理表にあるかのようにデ

ータを照会して分析するアプリケーションを作成することが可能です。DB2 ワークロード管理の表関数は、SYSPROC スキーマ名で修飾されます。

名前の先頭が *MON_* の表関数は、モニター・メトリック関数です。モニター・メトリックは、DB2 データ・サーバーの正常性および照会パフォーマンスに関するモニター・データを提供します。そのデータをサード・パーティー・ツールに対する入力として使用するか、自分で作成した追加のスクリプトと組み合わせて使用することにより、返されたメトリックを分析できます。以下には、DB2 ワークロード管理に関係するモニター・メトリック関数のみ示されています。モニター・メトリック表関数は、ワークロード管理統計の表関数に似ています。どちらも、システムで生じた作業について記述したエレメントを戻します。これらのモニター・メトリック表関数と DB2 ワークロード管理表関数の主な相違点は、以下のとおりです。

- DB2 ワークロード管理表関数は、平均、高水準点、標準偏差などの算出値といった、より統計学的な性質を備えたデータを提供します。一方、それと比較してモニター・メトリック表関数は、未加工のモニター・データの集合全体を提供します。
- DB2 統計関数が報告するデータがリセットされるのは、データが統計イベント・モニターに送信されるときです。高水準点などの値を、特定の収集間隔で意味あるものとするためにこのデータのリセットが必要になります。モニター・メトリック関数によって報告されるデータは統計イベント・モニターによってもキャプチャーされますが、決してリセットされることがありません。モニター・インターフェースによって報告されるデータは、データベースが非活動化されるまでの活動化されていた期間中、累積されます。

現在システム上で実行中の作業に関する情報のセットを返す表関数もあります。

表 49. システムで現在実行されている作業を示す表関数

情報収集の対象となるオブジェクト	**関数および返される情報**
ワークロード・オカレンス	WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数は、全データベース・メンバーにおける、サービス・クラスに割り当てられたワークロード・オカレンスのリストを返します。オカレンスごとに、ワークロードをサービス・クラスに割り当てるために使用される現在の状態と接続属性に関する情報と、アクティビティー・ボリュームおよび成功率を示すアクティビティー統計に関する情報があります。この表関数の使用方法の例については、 107 ページの『例: サービス・クラスによるエージェント使用の調査』を参照してください。 推奨されていませんが、WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 表関数も使用できます。

表 49. システムで現在実行されている作業を示す表関数 (続き)

情報収集の対象となるオブジェクト	関数および返される情報
ワークロード・オカレンス・アクティビティー	WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数は、ワークロード・オカレンスに関連付けられた現在のアクティビティーのリストを返します。アクティビティーごとに、アクティビティーの現在の状態 (例えば、実行中または待機中)、アクティビティーのタイプ (例えば、LOAD、READ、または DDL)、およびアクティビティーが開始した時刻に関する情報が得られます。この表関数の使用方法の例については、 268 ページの『例: DB2 ワークロード管理の表関数を使用したデータの集約』および 394 ページの『シナリオ: 完了に時間がかかり過ぎているアクティビティーの識別』を参照してください。 推奨されていませんが、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97 表関数も使用できます。
サービス・クラス・エージェント	WLM_GET_SERVICE_CLASS_AGENTS 表関数は、サービス・クラスまたはアプリケーション・ハンドルに関連付けられたデータベース・エージェントのリストを返します。返される情報は、エージェントの現在の状態、エージェントが実行中のアクション、およびそのアクションの状況も示します。この表関数の使用方法の例については、 107 ページの『例: サービス・クラスによるエージェント使用の調査』を参照してください。 推奨されていませんが、WLM_GET_SERVICE_CLASS_AGENTS_V97 表関数も使用できます。
アクティビティー	MON_GET_ACTIVITY_DETAILS 表関数は、アプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID で識別された特定のアクティビティーに関するメトリックなどの詳細情報を XML 文書として返します。返される詳細の 1 つはアクティビティー・タイプです。そのタイプに応じて追加データのセットが返されます。例えば SQL アクティビティーの場合は、コスト見積もりのほか、ステートメント・テキスト、パッケージ・データ、戻されたまたは変更された行に関する情報が提供されます。特に分離レベルおよびプロセッサー・リソースに関する詳細も得られます。 推奨されていませんが、WLM_GET_ACTIVITY_DETAILS 表関数も使用できます。この表関数の使用方法を示す例については、 264 ページの『例: DB2 ワークロード管理の表関数を使用して、異なるレベルで現行システムの動作をモニターする』を参照してください。

いくつかの表関数は、サービス・サブクラスとワークロード・オブジェクトによって集約された、システムで実行されたすべての要求に関するモニター・データを戻します。

表 50. DB2 ワークロード管理オブジェクトによって集約されるモニター・データを示す表関数

データの集約の対象となるオブジェクト	関数および返される情報
ワークロード	MON_GET_WORKLOAD 表関数と MON_GET_WORKLOAD_DETAILS 表関数は、どちらも 1 つ以上のワークロードに関するメトリックを返します。この関数が返すメトリックは、同じワークロード定義を使用するワークロード・オカレンスすべてのメトリックをすべて累積したものとなります。 MON_GET_WORKLOAD 表関数は、最も一般的に使用されるメトリックを列ベースの形式で返します。これは、基本的なメトリックを取得するための効率のよい方法です。 MON_GET_WORKLOAD_DETAILS 表関数は、使用可能なメトリックの全セットを XML 文書形式で返します。これにより、出力をフォーマット設定する際に最大限の柔軟性が得られます。XML ベースの出力は、XML パーサーで直接に解析することもできますし、XMLTABLE 関数でリレーショナル形式に変換することもできます。
サービス・サブクラス	MON_GET_SERVICE_SUBCLASS 表関数と MON_GET_SERVICE_SUBCLASS_DETAILS 表関数は、どちらも 1 つ以上のサービス・サブクラスに関するメトリックを返します。表関数が返すメトリックは、指定されたサービス・サブクラスのもとで実行された要求のメトリックをすべて累積したものとなります。 MON_GET_SERVICE_SUBCLASS 表関数は、最も一般的に使用されるメトリックを列ベースの形式で返します。これは、基本的なメトリックを取得するための効率のよい方法です。 MON_GET_SERVICE_SUBCLASS_DETAILS 表関数は、使用可能なメトリックの全セットを XML 文書形式で返します。これにより、出力をフォーマット設定する際に最大限の柔軟性が得られます。XML ベースの出力は、XML パーサーで直接に解析することもできますし、XMLTABLE 関数でリレーショナル形式に変換することもできます。
接続	MON_GET_CONNECTION 表関数は、システムに対するユーザー接続全体で集約されるデータを返します。
作業単位	MON_GET_UNIT_OF_WORK 表関数は、ユーザー接続内の現在の作業単位で集約されるデータを返します。

統計情報

さまざまなオブジェクトについての一般統計情報も使用できます。この統計情報は、例えば DB2 ワークロード管理構成に対する変更に期待どおりの効果があったかどうかの検証など、さまざまな目的に使用できます。例えば、READ アクティビティーを分類するための新規のワーク・クラスを作成した場合、READ アクティビティーがその新規ワーク・クラスに正しく分類されているかどうかを検証できます。また、表関数を使用してシステムに関する特定の問題をすぐに認識することができます。例えば、表関数を使用して平均的なアクティビティーの存続時間の許容値を判別することや、この値が通常の範囲を超えていて、さらに調査の必要な問題を指し示している可能性がある場合に、それを認識することができます。

次の表は、表関数を使用することによって取得できる統計のリストです。すべての統計表関数は、統計が最後にリセットされたときからの累積統計を返します。

表 51. 統計情報を示す表関数

統計の対象となるオブジェクト	**関数および返される統計**
サービス・スーパークラス	WLM_GET_SERVICE_SUPERCLASS_STATS 表関数は、サービス・スーパークラス・レベルでの全データベース・メンバーにおけるサマリー統計、つまり同時接続の最高水準点を示します。これはワークロード・アクティビティーのピークを判別するときに役立ちます。
サービス・サブクラス	WLM_GET_SERVICE_SUBCLASS_STATS 表関数は、サービス・サブクラス・レベル (サービス・サブクラスで実行されるすべてのアクティビティー) での全データベース・メンバーにおけるサマリー統計を示します。統計には、完了したアクティビティー数と平均実行時間が含まれます。この情報は、サービス・クラスおよびデータベース・メンバー全体の一般的なシステム・ヘルスおよびアクティビティーの分散を調べるときに役立ちます。この表関数の使用方法の例については、 267 ページの『例: サービス・クラスからの特定時点の統計の取得』、 268 ページの『例: DB2 ワークロード管理の表関数を使用したデータの集約』、 105 ページの『例: サービス・クラス関連のシステム・スローダウンの分析』、および 390 ページの『シナリオ: ワークロード関連のシステム・スローダウンの調査』を参照してください。 推奨されていませんが、WLM_GET_SERVICE_SUBCLASS_STATS_V97 表関数も使用できます。
ワークロード	WLM_GET_WORKLOAD_STATS 表関数は、ワークロード・レベルでの全データベース・メンバーにおけるサマリー統計を示します。これには、並行ワークロード・オカレンスの最高水準点、および完了したアクティビティーの番号が含まれます。この情報は、一般的なシステム・ヘルスをモニターするときや、問題領域を特定するために詳しく調べるときに役立ちます。この表関数の使用方法の例については、 390 ページの『シナリオ: ワークロード関連のシステム・スローダウンの調査』を参照してください。 推奨されていませんが、WLM_GET_WORKLOAD_STATS_V97 表関数も使用できます。
ワーク・アクション・セット	WLM_GET_WORK_ACTION_SET_STATS 表関数は、ワーク・アクション・セット・レベルでの全データベース・メンバーにおけるサマリー統計、つまり対応するワーク・アクションが適用された各ワーク・クラス内のアクティビティーの数を示します。この情報は、ワーク・アクション・セットの有効性を理解し、システムで実行中のアクティビティーのタイプを理解するうえで役立ちます。この表関数の使用方法の例については、 71 ページの『例: アクティビティー・タイプごとのワークロードの分析』を参照してください。

表 51. 統計情報を示す表関数 (続き)

統計の対象となるオブジェクト	関数および返される統計
しきい値キュー	WLM_GET_QUEUE_STATS 表関数は、しきい値で使用される、キューの全データベース・メンバーにおけるサマリー統計を示します。統計には、キューに入れられたアクティビティーの現在数と総数、およびキューで費やされた合計時間が含まれます。この情報は、現在キューに入れられているアクティビティーを照会するときや、しきい値を正しく定義したことを検証するときに役立ちます。キューイングが過度に発生する場合は、しきい値の制限が強すぎることを示している可能性があります。また、キューイングがほとんど発生しない場合は、しきい値の制限が弱すぎるか必要ないことを示している可能性があります。

統計は、統計が収集される期間に意味がある場合にのみ役立ちます。
WLM_COLLECT_STATS ストアード・プロシージャーを使用して非常に長い期間にわたる統計を収集すると、調べる目的の期間がどうであれ、あまり役に立たない可能性があります。古いデータが多くなりすぎるために、傾向の変化や問題領域を識別するのが難しい場合があるからです。そのため、いつでも統計をリセットできるようになっています。

デフォルトのワークロードおよびデフォルトのユーザー・サービス・クラスがあるため、モニター機能は DB2 データ・サーバーをインストールした時点から存在します。この機能は、ワークロードとそれを割り当てることができるサービス・クラスを作成するために使用できるアクティビティーのソースの識別を開始するのに役立ちます。

例: DB2 ワークロード管理表関数の使用

DB2 ワークロード管理のリアルタイム・モニターで大量のデータが得られます。このトピックの例では、その情報を使えるようにする方法を示します。

ここでは、デフォルトのワークロードおよびサービス・クラスのみが使用されている状態を考えます。この例を使うと、データ・サーバーで何が実行されているかを正確に把握するためにどのように表関数を使用できるかを理解できます。以下のステップを実行してください。

1. サービス・スーパークラス統計表関数を使用してすべてのサービス・スーパークラスを表示します。 DB2 9.5 以降をインストール、またはそのバージョンにアップグレードした後、3 つのデフォルトのスーパークラスが定義されます。つまり、1 つは保守アクティビティー用、1 つはシステム・アクティビティー用、もう 1 つはユーザー・アクティビティー用です。ここで使用するサービス・クラスは SYSDEFAULTUSERCLASS です。

```
SELECT VARCHAR(SERVICE_SUPERCLASS_NAME,30) AS SUPERCLASS
   FROM TABLE(WLM_GET_SERVICE_SUPERCLASS_STATS('',-1)) AS T

SUPERCLASS
------------------------------
SYSDEFAULTSYSTEMCLASS
```

```
SYSDEFAULTMAINTENANCECLASS
SYSDEFAULTUSERCLASS

  3 record(s) selected.
```

2. サービス・サブクラス統計表関数を使用して、SYSDEFAULTUSERCLASS スーパークラスのすべてのサービス・サブクラスの統計を表示します。それぞれのサービス・サブクラスについては、現在処理中の要求の量、実行が完了したアクティビティーの数、および全メンバーにおけるアクティビティーの全体的な分散 (分散が不均一である場合、問題がある可能性があります) が示されます。オプションで、アクティビティーの平均存続時間、アクティビティーがキューで費やす平均時間などを含む追加の統計を取得することができます。ALTER SERVICE CLASS ステートメント上の COLLECT AGGREGATE ACTIVITY DATA キーワードを指定して、集約アクティビティーの統計の収集を有効にすることにより、サービス・サブクラスのオプションの統計を取得できます。

```
SELECT VARCHAR(SERVICE_SUPERCLASS_NAME, 20) AS SUPERCLASS,
   VARCHAR(SERVICE_SUBCLASS_NAME, 20) AS SUBCLASS,
   COORD_ACT_COMPLETED_TOTAL,
   COORD_ACT_ABORTED_TOTAL,
   COORD_ACT_REJECTED_TOTAL,
   CONCURRENT_ACT_TOP
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS(
   'SYSDEFAULTUSERCLASS', 'SYSDEFAULTSUBCLASS', -1))
   AS T
```

```
SUPERCLASS           SUBCLASS             COORD_ACT_COMPLETED_TOTAL COORD_ACT_ABORTED_TOTAL COORD_ACT_REJECTED_TOTAL CONCURRENT_ACT_TOP
-------------------- -------------------- ------------------------- ----------------------- ------------------------ ------------------
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS                           2                       0                        0                  1

  1 record(s) selected.
```

3. 特定のサービス・サブクラスにおいては、ワークロード・オカレンス情報表関数を使用して、サービス・サブクラスにマップされるワークロードのオカレンスをリストします。表関数は、すべての接続属性を表示します。これを使用してアクティビティーのソースを識別できます。この情報は、将来カスタム・ワークロード定義を決定する上で非常に役立つ場合があります。例えば、ここでリストされる特定のワークロード・オカレンスには、完了したアクティビティーのカウンターで示されるようにアプリケーションからの大量の作業があるかもしれません。

```
SELECT APPLICATION_HANDLE,
   VARCHAR(WORKLOAD_NAME, 30) AS WORKLOAD,
   VARCHAR(SESSION_AUTH_ID, 20) AS SESSION_AUTH_ID,
   VARCHAR(APPLICATION_NAME, 20) AS APPL_NAME
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES(
   'SYSDEFAULTUSERCLASS', 'SYSDEFAULTSUBCLASS', -1))
   AS T
```

```
APPLICATION_HANDLE   WORKLOAD                       SESSION_AUTH_ID      APPL_NAME
-------------------- ------------------------------ -------------------- --------------------
                 431 SYSDEFAULTUSERWORKLOAD         SWALKTY              db2bp

  1 record(s) selected.
```

a. そのアプリケーションでは、ワークロード・オカレンス・アクティビティー情報表関数を使用して、アプリケーションの接続により作成された、全データベース・メンバーにおける現在のアクティビティーを表示します。この情報は、データ・サーバー上で問題の原因となっている可能性があるアクティビティーの識別など、多くの目的に使用できます。

```
SELECT APPLICATION_HANDLE,
   LOCAL_START_TIME,
   UOW_ID,
```

```
   ACTIVITY_ID,
   ACTIVITY_TYPE
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(431,-1)) AS T

APPLICATION_HANDLE   LOCAL_START_TIME          UOW_ID      ACTIVITY_ID ACTIVITY_TYPE
-------------------- -------------------------- ----------- ----------- --------------------------------
                 431 2008-06-17-12.49.46.854259         11           1 READ_DML

  1 record(s) selected
```

b. アクティビティーごとに、アクティビティーの詳細表関数を使用してさらに詳細な情報を取り出します。このデータは、大量の行を返している SQL ステートメントがあること、長時間アイドル状態であるアクティビティーがあること、または極めて大きな見積コストを持つ照会が実行中であることを示している可能性があります。このような状況では、将来の損害を与える可能性のある動作を識別して防ぐために何らかのしきい値を定義することには意味があると考えられます。

```
SELECT VARCHAR(NAME, 20) AS NAME,
   VARCHAR(VALUE, 40) AS VALUE
FROM TABLE(WLM_GET_ACTIVITY_DETAILS(431,11,1,-1))
   AS T WHERE NAME IN ('UOW_ID', 'ACTIVITY_ID', 'STMT_TEXT')

NAME                 VALUE
-------------------- ----------------------------------------
UOW_ID               1
ACTIVITY_ID          1
STMT_TEXT            select * from syscat.tables

  3 record(s) selected.
```

例: DB2 ワークロード管理の表関数を使用して、異なるレベルで現行システムの動作をモニターする

DB2 ワークロード管理は、ワークロード管理構成に関するデータを取得するために使用可能な表関数をいくつか備えています。

DB2 バージョン 9.5 以降をインストールすると、デフォルトのワークロードおよびサービス・クラスのセットが作成されます。独自の DB2 ワークロード管理ソリューションを実装する方法を決定する前に、表関数を使用して、システムで実行されている作業の監視を、デフォルトのワークロード・オカレンス、サービス・クラス、およびアクティビティーの観点から行うことができます。

最初に、サービス・クラス内のワークロード・オカレンスのリストを取得することができます。これを行うには
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を使用します。以下の例では、*service_superclass_name* および *service_subclass_name* には空ストリング、および *member* には -2 (ワイルドカード文字) を渡します。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(CHAR(COORD_MEMBER),1,4) AS COORDMEMB,
       SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL,
       SUBSTR(CHAR(WORKLOAD_NAME),1,22) AS WORKLOAD_NAME,
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', -2)) AS SCINFO
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB, APPHNDL, WORKLOAD_NAME, WLO_ID
```

システムに 4 つのデータベース・メンバーがあり、照会を発行したときに 2 つのアプリケーションがデータベース上でアクティビティーを実行しているとします。結果は以下のようになります。

```
SUPERCLASS_NAME     SUBCLASS_NAME      MEMB COORDMEMB APPHNDL WORKLOAD_NAME          WLO_ID
------------------- ------------------ ---- --------- ------- ----------------------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0         1       SYSDEFAULTUSERWORKLOAD 1
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0         2       SYSDEFAULTUSERWORKLOAD 2
```

結果は、両方のワークロード・オカレンスが SYSDEFAULTUSERWORKLOAD ワークロードに割り当てられたことを示しています。さらに、この結果は、両方のワークロード・オカレンスが SYSDEFAULTUSERCLASS サービス・スーパークラスの SYSDEFAULTSUBCLASS サービス・サブクラスに割り当てられており、どちらのワークロード・オカレンスも同じコーディネーター・メンバー (メンバー 0) のものであることも示しています。

次に、WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を再び使用して、2 つのワークロード・オカレンスの接続属性を以下のように判別することもできます。

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL,
       SUBSTR(CHAR(WORKLOAD_NAME),1,22) AS WORKLOAD_NAME,
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID,
       SUBSTR(CHAR(SYSTEM_AUTH_ID),1,9) AS SYSAUTHID,
       SUBSTR(CHAR(APPLICATION_NAME),1,15) AS APPLNAME
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', 0)) AS SCINFO
ORDER BY APPHNDL, WORKLOAD_NAME, WLO_ID

APPHNDL WORKLOAD_NAME          WLO_ID SYSAUTHID APPLNAME
------- ---------------------- ------ --------- ---------------
1       SYSDEFAULTUSERWORKLOAD 1      LYNN      accountspay
2       SYSDEFAULTUSERWORKLOAD 2      KATE      businessobjects
```

次に、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用して、いずれかのワークロード・オカレンスの現在のアクティビティーを表示することができます。

```
SELECT SUBSTR(CHAR(COORD_MEMBER),1,5) AS COORD,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
       SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
       SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
       SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PARACTID,
       SUBSTR(ACTIVITY_TYPE,1,9) AS ACTTYPE,
       SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(1, -2)) AS WLOACTS
ORDER BY MEMB, UOWID, ACTID

COORD MEMB UOWID ACTID PARUOWID PARACTID ACTTYPE  NESTING
----- ---- ----- ----- -------- -------- -------- -------
0     0    1     3     -        -        CALL     0
0     0    1     5     1        3        READ_DML 1
0     1    1     5     -        -        READ_DML 1
0     2    1     5     -        -        READ_DML 1
0     3    1     5     -        -        READ_DML 1
```

照会結果は、ワークロード・オカレンス 1 が 2 つのアクティビティーを実行中であることを示しています。 1 つのアクティビティーはストアード・プロシージャー (CALL のアクティビティー・タイプで示される)、そしてもう 1 つのアクティビティーは読み取りを実行する DML アクティビティー (例、SELECT ステートメントなど) です。DML アクティビティーは、ストアード・プロシージャー呼び出しにネ

ストされています。 DML アクティビティーの親作業単位 ID および親アクティビティー ID が CALL アクティビティーの作業単位 ID およびアクティビティー ID と一致しているため、DML アクティビティーがネストされているということがわかります。さらに、DML アクティビティーがデータベース・メンバー 0、1、2、および 3 で実行されていることもわかります。親 ID の情報はコーディネーター・メンバーでのみ取得できます。

現在実行中の個別のアクティビティーに関する詳細情報を取得するには、MON_GET_ACTIVITY_DETAILS 表関数を使用できます。この表関数は XML 文書を戻します。この文書内のエレメントがアクティビティーについて記述しています。以下の例では、XMLTABLE 関数を使用して、XML 出力から結果表が戻されます。

```
SELECT D.APP_HANDLE,
       D.MEMBER,
       D.COORD_MEMBER,
       D.LOCAL_START_TIME,
       D.UOW_ID,
       D.ACTIVITY_ID,
       D.PARENT_UOW_ID,
       D.PARENT_ACTIVITY_ID,
       D.ACTIVITY_TYPE,
       D.NESTING_LEVEL,
       D.INVOCATION_ID,
       D.ROUTINE_ID
FROM TABLE(MON_GET_ACTIVITY_DETAILS(65592, 1, 1, -2)) AS ACTDETAILS,
XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
           '$details/db2_activity_details' PASSING XMLPARSE(DOCUMENT
    ACTDETAILS.DETAILS) as "details"
COLUMNS "APP_HANDLE"         BIGINT    PATH 'application_handle',
        "MEMBER"             BIGINT    PATH 'member',
        "COORD_MEMBER"       BIGINT    PATH 'coord_member',
        "LOCAL_START_TIME"   VARCHAR(26) PATH 'local_start_time',
        "UOW_ID"             BIGINT    PATH 'uow_id',
        "ACTIVITY_ID"        BIGINT    PATH 'activity_id',
        "PARENT_UOW_ID"      BIGINT    PATH 'parent_uow_id',
        "PARENT_ACTIVITY_ID" BIGINT    PATH 'parent_activity_id',
        "ACTIVITY_TYPE"      VARCHAR(10) PATH 'activity_type',
        "NESTING_LEVEL"      BIGINT    PATH 'nesting_level',
        "INVOCATION_ID"      BIGINT    PATH 'invocation_id',
        "ROUTINE_ID"         BIGINT    PATH 'routine_id'
) AS D;

APP_HANDLE MEMBER COORD_MEMBER LOCAL_START_TIME            UOW_ID ACTIVITY_ID
---------- ------ ------------ -------------------------- ------ -----------
65592      1      1            2009-04-07-18.39.42.549197  1      1
65592      0      1            2009-04-07-18.39.42.552763  1      1

 PARENT_UOW_ID PARENT_ACTIVITY_ID ACTIVITY_TYPE NESTING_LEVEL INVOCATION_ID ROUTINE_ID
 ------------- ------------------ ------------- ------------- ------------- ----------
 -             -                  READ_DML      0             0             0
 -             -                  READ_DML      0             0             0

  2 record(s) selected.
```

注: 照会結果は読みやすくするために 2 つに分割されています。

前述の表関数は、システムで実行中の作業に関する概要的な記述を提供します。これらの表関数が作業の状況について提供する情報は、EXECUTING などのアクティビティー状態に制限されます。ある時点で、サービス・クラスで何が起きているかを正確に知るために詳しく調査するために、WLM_GET_SERVICE_CLASS_AGENTS 表関数を実行できます。

以下の例では、*application_handle* に 1、*member* に -2 (ワイルドカード文字) を渡すことによって WLM_GET_SERVICE_CLASS_AGENTS を呼び出します。

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(CHAR(AGENT_TID),1,9) AS AGENT_TID,
       SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
       SUBSTR(AGENT_STATE,1,10) AS AGENTSTATE,
       SUBSTR(REQUEST_TYPE,1,14) AS REQTYPE,
       SUBSTR(CHAR(UOW_ID),1,6) AS UOW_ID,
       SUBSTR(CHAR(ACTIVITY_ID),1,6) AS ACT_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', 1, -2)) AS SCDETAILS
ORDER BY APPHANDLE, MEMB, AGENT_TID

APPHANDLE MEMB AGENT_TID AGENTTYPE   AGENTSTATE REQTYPE      UOW_ID ACT_ID
--------- ---- --------- ----------- ---------- ------------------------
1         0    3         COORDINATOR ACTIVE     FETCH             1      5
1         0    4         PDBSUBAGENT ACTIVE     SUBSECTION:1      1      5
1         1    2         PDBSUBAGENT ACTIVE     SUBSECTION:2      1      5
```

結果は、メンバー 0 のコーディネーター・エージェントおよびサブエージェント、およびメンバー 1 のサブエージェントが、作業単位 ID が 1 でアクティビティー ID が 5 のアクティビティーのために作動していることを示しています。コーディネーター・エージェント情報は、要求がフェッチ要求であることを示しています。

例: サービス・クラスからの特定時点の統計の取得

すべてのアクティビティーは、実行される前にサービス・クラスにマップされます。サービス・クラス統計の表関数を使用し、すべてのデータベース・メンバー上のすべてのサービス・クラスを照会して、特定時点の統計を取得することにより、システムをモニターできます。

以下のステートメントを使用して、アクティビティーの平均存続期間などのサービス・クラス統計を取得することができます。
WLM_GET_SERVICE_SUBCLASS_STATS 表関数の引数に空ストリングを渡した場合、結果はその引数によって制限されません。最後の引数 *member* の値は -2 (ワイルドカード文字) であり、これはすべてのデータベース・メンバーからのデータが返されることを意味します。

注: 存続時間情報が返されるのは、COLLECT AGGREGATE ACTIVITY DATA で定義されているサービス・クラスについてのみです。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3)) AS AVGLIFETIME,
       CAST(COORD_ACT_LIFETIME_STDDEV / 1000 AS DECIMAL(9,3)) AS STDDEVLIFETIME,
       SUBSTR(CAST(LAST_RESET AS VARCHAR(30)),1,16) AS LAST_RESET
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '', -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB

SUPERCLASS_NAME     SUBCLASS_NAME      MEMB AVGLIFETIME STDDEVLIFETIME LAST_RESET
------------------- ------------------ ---- ----------- ------------- ----------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0        691.242         34.322 2006-07-24-11.44
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1        644.740         22.124 2006-07-24-11.44
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2        612.431         43.347 2006-07-24-11.44
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3        593.451         28.329 2006-07-24-11.44
```

さらに、WLM_GET_SERVICE_SUBCLASS_STATS 表関数を次のように使用して、メンバーごとにサービス・クラスで実行される、コーディネーター・アクティビティーの並行性の最高水準点を取得することもできます。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       CONCURRENT_ACT_TOP AS ACTHIGHWATERMARK
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '',  -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB

SUPERCLASS_NAME     SUBCLASS_NAME      MEMB ACTHIGHWATERMARK
------------------- ------------------ ---- ----------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0                  10
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1                   0
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2                   0
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3                   0
```

完了したアクティビティーの平均存続期間と数を検討することにより、WLM_GET_SERVICE_SUBCLASS_STATS 表関数の出力を使用して、データベースの各メンバーのワークロードのロールアップ・ビューを取得することができます。表関数によって返された最高水準点と平均に大きなばらつきがある場合、システム上でワークロードが変更されている可能性があります。

例: DB2 ワークロード管理の表関数を使用したデータの集約

DB2 ワークロード管理構成では、表データでさまざまな集約を実行して、システムをモニターしたり、起こりうる問題を識別したりすることができます。

以下は、問題を識別するために実行可能なデータ集約の例です。

ワークロードで実行されている照会数の突然の増加を識別する

WL1 というワークロードがあると想定します。システム全体を通じて、ワークロードに対して実行中のネストなしコーディネーター・アクティビティーの合計数を表示することによって、ワークロードで非常に多くの照会が実行されている状態を識別することができます。

```
SELECT SUBSTR(WORKLOAD_NAME,1,22) AS WLNAME,
COUNT(*) AS TOTAL_EXE_ACT
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', -2)) AS APPS,
TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(APPS.APPLICATION_HANDLE, -2)) AS APPACTS
WHERE WORKLOAD_NAME = 'WL1' AND
APPS.DBPARTITIONNUM = APPS.COORD_PARTITION_NUM AND
ACTIVITY_STATE = 'EXECUTING' AND
NESTING_LEVEL = 0
GROUP BY WORKLOAD_NAME

WLNAME               TOTAL_EXE_ACT
-------------------- -------------
WL1                  5
```

例: WLM しきい値によってキューに入れられるアクティビティーと、そのキュー順序の判別

DB2 ワークロード・マネージャー (WLM) 表関数
WLM_GET_SERVICE_CLASS_AGENTS を使用すると、WLM しきい値によってキューに入れられるアプリケーションまたはアクティビティー、およびそのキュー内におけるアプリケーションまたはアクティビティーの順序を表示できます。

WLM しきい値によってキューに入れられるエージェントには、そのエージェントに戻される対応する行の EVENT_OBJECT 列に値 WLM_QUEUE が入ります。また、AGENT_STATE_LAST_UPDATE_TIME 列には、エージェントが WLM_QUEUE 状態になった時刻、つまりエージェントがキューに入れられた時刻が入ります。この情報を使用すると、WLM しきい値によってキューに入れられたすべてのアプリケーションまたはアクティビティーをリストし、そのキュー・エントリー時刻を取得するための簡単なビューを構成できます。

以下に記されている例において、WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用して、WLM キュー情報を取得するためのビューを作成するには、以下のステートメントを実行します。

```
CREATE VIEW WLM_QUEUE_INFO (APPLICATION_HANDLE, UOW_ID, ACTIVITY_ID,
    THRESHOLD_NAME, QUEUE_ENTRY_TIME, MEMBER)
    AS SELECT APPLICATION_HANDLE, UOW_ID, ACTIVITY_ID,
    VARCHAR(EVENT_OBJECT_NAME, 128), AGENT_STATE_LAST_UPDATE_TIME,
    MEMBER FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS(NULL,NULL,NULL,-2))
    AS T WHERE EVENT_OBJECT = 'WLM_QUEUE'
```

注: TOTALSCPARTITIONCONNECTIONS しきい値に関しては、**uow_id** と **activity_id** は適用外 (NA) で、NULL として表示されます。

このビューを使用すると、以下のような質問に簡単に答えられます。

- WLM しきい値によって現在キューに入れられているアプリケーションまたはアクティビティーは幾つありますか。
- WLM しきい値キュー内におけるアプリケーションまたはアクティビティーの順序はどのようになっていますか。

例 1

それぞれのキューイングしきい値によってキューに入れられているアプリケーション数を数えるには、以下のステートメントを実行します。

```
SELECT VARCHAR(THRESHOLD_NAME, 30) AS THRESHOLD, COUNT(*)
     AS QUEUED_ENTRIES FROM WLM_QUEUE_INFO GROUP BY THRESHOLD_NAME
```

前述のステートメントを実行して取得される出力の例を、以下に記します。

```
THRESHOLD                      QUEUED_ENTRIES
------------------------------ --------------
TH1                                         3

  1 record(s) selected.
```

例 2

TH1 CONCURRENTDBCOORDACTIVITIES しきい値によってキューに入れられるすべてのアクティビティーをリスト表示し、それらのキュー・エントリー時刻によって判別されるキューにおけるアクティビティー順序を表示するには、以下のステートメントを実行します。

```
SELECT QUEUE_ENTRY_TIME, APPLICATION_HANDLE, UOW_ID, ACTIVITY_ID FROM
     WLM_QUEUE_INFO WHERE THRESHOLD_NAME = 'TH1' ORDER BY QUEUE_ENTRY_TIME ASC
```

前述のステートメントを実行して取得される出力の例を、以下に記します。

```
QUEUE_ENTRY_TIME           APPLICATION_HANDLE   UOW_ID      ACTIVITY_ID
-------------------------- -------------------- ----------- -----------
2009-11-09-18.08.32.583286                  145           1           2
2009-11-09-18.08.42.589623                  146           1           1
2009-11-09-18.08.54.607083                  147           1           1

  3 record(s) selected.
```

WLM イベント・モニターを使用した履歴モニター

DB2 ワークロード管理は、イベント・モニターを使用して将来または履歴分析のために活用できる可能性がある情報をキャプチャーします。

使用できるイベント・モニターは 3 つあります。それぞれのイベント・モニターは、それぞれ異なる目的で使用されます。

アクティビティー・イベント・モニター

このモニターは、サービス・クラス、ワークロード、またはワーク・クラスにおける個々のアクティビティーに関する情報またはしきい値に違反したアクティビティーに関する情報をキャプチャーします。アクティビティーごとにキャプチャーされるデータの量は構成可能で、ディスク・スペースの容量およびモニター・データを維持しなければならない時間の長さを決定する際に考慮する必要があります。アクティビティー・データの一般的な使用法としては、**db2advis** などのツールへの入力データとして、または一連の照会における表、列、および索引の使用量の判別に役立つアクセス・プランを (Explain ユーティリティーから) 使用するために活用するというものです。

アクティビティーに関する情報を収集するには、該当するアクティビティーが属するサービス・クラス、ワークロード、またはワーク・アクションに対して、あるいは該当するアクティビティーが違反する可能性のあるしきい値に対して、COLLECT ACTIVITY DATA を指定します。アクティビティーが完了すると、アクティビティーが正常に完了したかどうかにかかわらず、情報が収集されます。

データベースが非活動化されるときにアクティビティー・イベント・モニターがアクティブになっていると、キューのバックログ・アクティビティー・レコードはすべて破棄されることに留意してください。アクティビティー・イベント・モニター・レコードをすべて取得し、どれも破棄されないようにするには、データベースを非活動化する前にまず、アクティビティー・イベント・モニターを明示的に非活動化します。アクティビティー・イベント・モニターが明示的に非活動化されると、イベント・モニターが非活動化される前にキュー内のすべてのバックログ・アクティビティー・レコードが処理されます。

しきい値違反イベント・モニター

このモニターは、しきい値に違反した場合に情報をキャプチャーします。これは、どのしきい値に違反したのか、違反の原因となったアクティビティー、およびそれが発生した時に取られたアクションを知らせます。

しきい値に COLLECT ACTIVITY DATA を指定した場合、アクティビティー・イベント・モニターが作成されてアクティブになると、しきい値に違反

しているアクティビティーに関する情報も収集されますが、この情報が収集されるのはアクティビティーが終了 (正常に完了あるいは失敗) したときです。

しきい値に関する詳細を入手するには SYSCAT.THRESHOLDS ビューを照会します。

統計イベント・モニター

このモニターは、詳細なアクティビティー情報をキャプチャーする代わりに、集約データ (完了したアクティビティーの数、平均実行時間など) を収集することによりオーバーヘッドを少なくします。集約データには、存続時間、キュー時間、実行時間および見積コストを含むさまざまなアクティビティー測定のヒストグラムが含まれます。ヒストグラムを使用して値の分布を理解し、異常値を識別し、そして平均および標準偏差のような追加統計を計算することができます。例えば、ヒストグラムはユーザーの体感する応答時間の変化を理解するのに役立ちます。変動性が高い場合、平均の存続時間そのものは、ユーザーの体感する応答時間を反映するものとはなりません。
イベント・モニターに統計を送信する方法について詳しくは、 302 ページの『統計イベント・モニターを使用したワークロード管理統計の収集』を参照してください。

以下の図は、ワークロード情報にアクセスするために使用できるさまざまなモニター・オプションを示しています。このように、リアルタイム統計にアクセスする表関数と、個々のアクティビティーの効率的集約または詳細としてキャプチャーされるアクティビティーの詳細および履歴情報を使用できます。

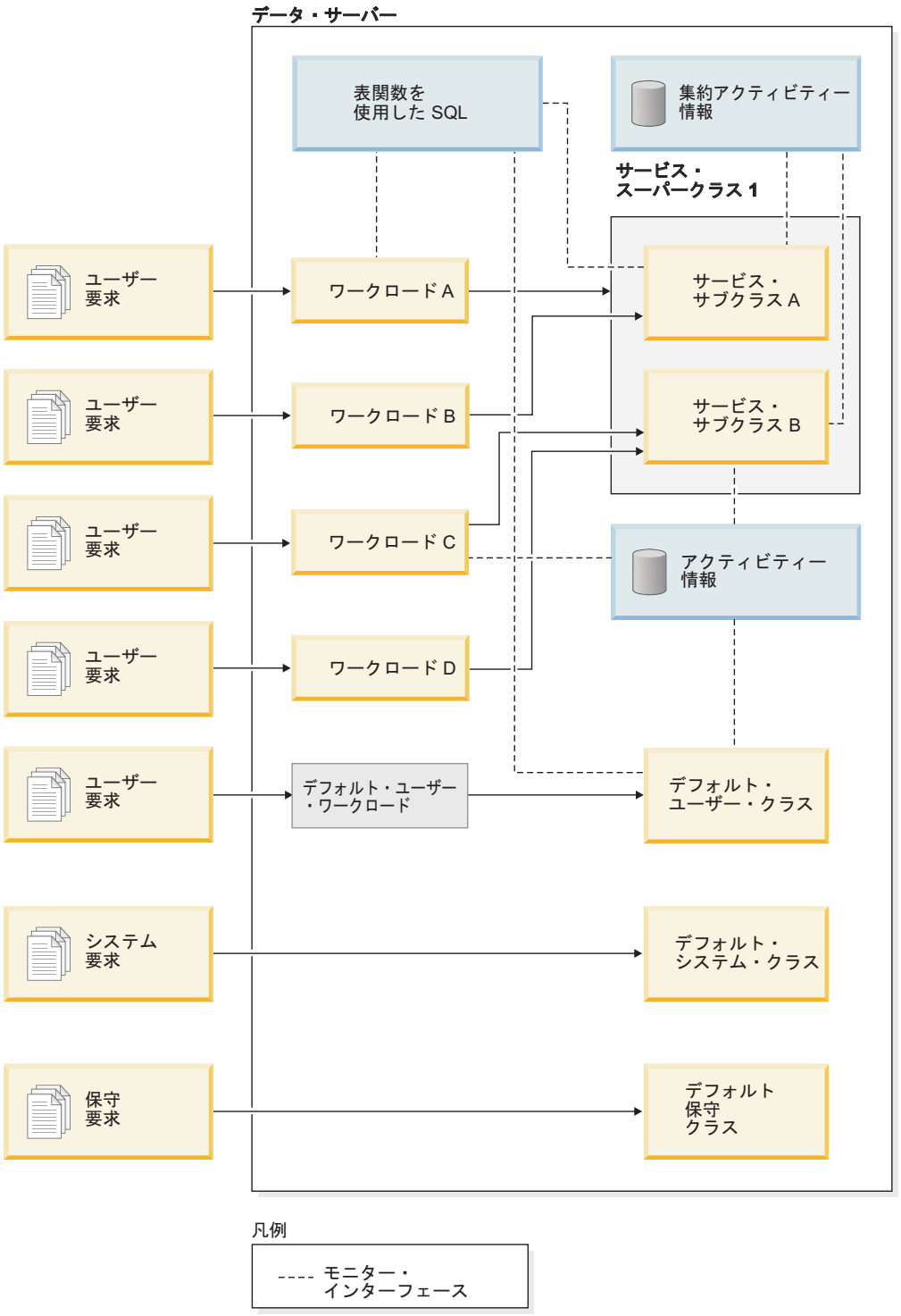


図 46. モニターを使用したワークロード管理

ステートメント、接続、およびトランザクション・イベント・モニターとは異なり、アクティビティー、統計、およびしきい値違反のイベント・モニターにはイベント条件 (つまり、CREATE EVENT MONITOR ステートメントの WHERE キーワードで指定された条件) がありません。代わりに、これらのイベント・モニターは、サービス・クラス、ワークロード、ワーク・クラス、およびしきい値の属性に

依存することによって、これらのオブジェクトが自分のアクティビティー情報または集約情報を各モニターに送るかどうかを決めます。

通常、イベント・モニターは表またはファイルのいずれかにデータを書き込みます。これらの表またはファイルから自動的にデータが削除されるわけではないため、自分で定期的にデータを削除する必要があります。

sqllib/misc ディレクトリーの **wlmevmon.ddl** スクリプトを使用して、DB2ACTIVITIES、DB2STATISTICS、および DB2THRESHOLDVIOLATIONS という 3 つのイベント・モニターを作成して使用可能にすることができます。必要に応じて、スクリプトを変更して表スペースまたは他のパラメーターを変更してください。

例

例: 統計イベント・モニターを使用して見積コストの大きい照会を識別する: おそらく照会自体の最適化が不十分なために、高コストの大規模な照会がデータベース・ワークロードにときどき含まれているように思われます。こうした照会を識別し、これらの照会がシステム上のリソースを過度に消費することがないようにする必要があります。パフォーマンスを改善するためにはおそらく照会のいくつかを書き直すことになるという長期目標で臨みます。統計イベント・モニターを使用すると、照会の見積コストを低オーバーヘッドで測定できます。この見積コストを使用することで、データ・サーバーに対する照会の最大許容見積コストを決定できます。最適化が不十分な照会は通常、見積コストが他のほとんどの照会の見積コストよりも何倍も大きいことで区別できます。

始めに、次のように統計イベント・モニターを作成して活動化し、照会が実行されるサービス・クラスの拡張集約アクティビティー・データの収集を開始する必要があります。

```
CREATE EVENT MONITOR DB2STATISTICS
  FOR STATISTICS WRITE TO TABLE

SET EVENT MONITOR DB2STATISTICS STATE 1
```

この例では、すべての照会が SYSDEFAULTUSERCLASS サービス・クラスの SYSDEFAULTSUBCLASS サブクラスで実行されますが、次のように必要なデータを収集するように変更できます。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
   COLLECT AGGREGATE ACTIVITY DATA EXTENDED
```

丸一日かけると、データ・サーバーが通常処理する照会の範囲の妥当な概算が得られる可能性があります。 1 日の終わりに、メモリーから収集された統計を統計イベント・モニターにコピーします。そのためには、次のように WLM_COLLECT_STATS ストアード・プロシージャーを実行します。

```
CALL WLM_COLLECT_STATS()
```

イベント・モニター表に書き込まれたさまざまな統計とともに、照会の見積コスト統計があります。これを調べるには、次のようにサービス・クラス統計表 SCSTATS_DB2STATISTICS を照会します。

```
SELECT STATISTICS_TIMESTAMP,
       COORD_ACT_EST_COST_AVG,
       COST_ESTIMATE_TOP
```

```
FROM SCSTATS_DB2STATISTICS
WHERE SERVICE_SUPERCLASS_NAME = 'SYSDEFAULTUSERCLASS'
  AND SERVICE_SUBCLASS_NAME = 'SYSDEFAULTSUBCLASS'

STATISTICS_TIMESTAMP       COORD_ACT_EST_COST_AVG COST_ESTIMATE_TOP
-------------------------- ---------------------- --------------------
2008-09-03-09.49.04.455979                 169440             13246445

  1 record(s) selected.
```

この出力は、平均的な照会の見積コストは数十万 timeron の範囲であることと、最大の照会の見積コストは 1 千万 timeron より大きいことを示しています。 1 千万 timeron 以上の照会が異常値であることは、見積コスト・ヒストグラムを調べることによって確認できます。このヒストグラムは、出力で示される平均と最高水準点がイベント・モニター表に書き込まれるのと同時に生成されたものです。ヒストグラムを調べるには、次のように HISTOGRAMBIN_DB2STATISTICS 表を照会します。

```
SELECT STATISTICS_TIMESTAMP,
       TOP,
       NUMBER_IN_BIN
FROM HISTOGRAMBIN_DB2STATISTICS HIST,
     SYSCAT.SERVICECLASSES SC
WHERE HIST.SERVICE_CLASS_ID = SC.SERVICECLASSID
  AND SC.PARENTSERVICECLASSNAME = 'SYSDEFAULTUSERCLASS'
  AND SC.SERVICECLASSNAME = 'SYSDEFAULTSUBCLASS'
  AND HISTOGRAM_TYPE = 'COORDACTESTCOST'"

STATISTICS_TIMESTAMP       TOP                  NUMBER_IN_BIN
-------------------------- -------------------- --------------------
2008-09-03-09.49.04.455979                    1                    0
2008-09-03-09.49.04.455979                    2                    0
2008-09-03-09.49.04.455979                    3                    0
2008-09-03-09.49.04.455979                    5                    0
2008-09-03-09.49.04.455979                    8                    0
2008-09-03-09.49.04.455979                   12                    1
2008-09-03-09.49.04.455979                   19                    0
2008-09-03-09.49.04.455979                   29                    0
2008-09-03-09.49.04.455979                   44                    2
2008-09-03-09.49.04.455979                   68                    5
2008-09-03-09.49.04.455979                  103                   22
2008-09-03-09.49.04.455979                  158                   14
2008-09-03-09.49.04.455979                  241                   54
2008-09-03-09.49.04.455979                  369                    2
2008-09-03-09.49.04.455979                  562                  142
2008-09-03-09.49.04.455979                  858                   21
2008-09-03-09.49.04.455979                 1309                  123
2008-09-03-09.49.04.455979                 1997                  512
2008-09-03-09.49.04.455979                 3046                  643
2008-09-03-09.49.04.455979                 4647                  201
2008-09-03-09.49.04.455979                 7089                  875
2008-09-03-09.49.04.455979                10813                 1445
2008-09-03-09.49.04.455979                16493                 5386
2008-09-03-09.49.04.455979                25157                 2409
2008-09-03-09.49.04.455979                38373                 8940
2008-09-03-09.49.04.455979                58532                 9820
2008-09-03-09.49.04.455979                89280                 2149
2008-09-03-09.49.04.455979               136181                  798
2008-09-03-09.49.04.455979               207720                 2411
2008-09-03-09.49.04.455979               316840                14989
2008-09-03-09.49.04.455979               483283                 9831
2008-09-03-09.49.04.455979               737162                 1451
2008-09-03-09.49.04.455979              1124409                  213
2008-09-03-09.49.04.455979              1715085                   24
2008-09-03-09.49.04.455979              2616055                    1
2008-09-03-09.49.04.455979              3990325                    0
2008-09-03-09.49.04.455979              6086529                    0
```

```
2008-09-03-09.49.04.455979              9283913                    0
2008-09-03-09.49.04.455979             14160950                    3
2008-09-03-09.49.04.455979             21600000                    0
2008-09-03-09.49.04.455979                   -1                    0
```

このヒストグラムでは、top が 2616055 より大きい照会の number_in_bin 列の値は、top が 14160950 に達する (number_in_bin が 3 になる) まではゼロです。この 3 つの照会は異常値であり、照会の見積コストが 1 千万 timeron を超えたら起動するように ESTIMATEDSQLCOST しきい値を設定することによって制御できます。このようにすることで、こうしたアクティビティーが実行されないようにし、それらをより綿密にモニターできます。

例: しきい値違反イベント・モニターの使用: 特定の見積コストのアクティビティーを制御するために、特定の見積コストを超える合計ワークロードのサブセットのみに適用されるワークロードの ESTIMATEDSQLCOST しきい値を定義します。見積コスト・ヒストグラムを調べた結果、見積コストが 0 から 3 百万 timeron 未満の範囲のアクティビティーが頻繁に発生することと、見積コストが 1 千万 timeron を超えるアクティビティーはまれにしか発生しないこと (おそらく 1 日に数回のみで、デカルト結合の使用など照会での何らかの不備がいつも原因のようである) が分かっています。

1 日にごくわずかしか発生しないが実行を許可すべきでないこうしたアクティビティーを停止する場合に 1 千万 timeron というしきい値が有効かどうかを検証するには、次のようにしきい値イベント・モニターを作成して活動化します。

```
CREATE THRESHOLD TH1
  FOR DATABASE ACTIVITIES
  ENFORCEMENT DATABASE
  WHEN ESTIMATEDSQLCOST > 10000000
  STOP EXECUTION

CREATE EVENT MONITOR DB2THRESHOLDVIOLATIONS
  FOR THRESHOLD VIOLATIONS
  WRITE TO TABLE

SET EVENT MONITOR DB2THRESHOLDVIOLATIONS STATE 1
```

1 日が終わったら、発生したしきい値違反を調べます。そのためには、次のようにしきい値違反表を照会します。

```
SELECT THRESHOLDID,
       SUBSTR(THRESHOLD_PREDICATE, 1, 20) PREDICATE,
       TIME_OF_VIOLATION,
       THRESHOLD_MAXVALUE,
       THRESHOLD_ACTION
FROM THRESHOLDVIOLATIONS_DB2THRESHOLDVIOLATIONS
ORDER BY TIME_OF_VIOLATION, THRESHOLDID
```

```
THRESHOLDID PREDICATE            TIME_OF_VIOLATION          THRESHOLD_MAXVALUE   THRESHOLD_ACTION
----------- -------------------- -------------------------- -------------------- ----------------
          1 EstimatedSQLCost     2008-09-02-22.39.10.000000             10000000 Stop

  1 record(s) selected.
```

例: アクティビティー・イベント・モニターの使用

先ほどの例では、イベント・モニター表のしきい値情報を収集して、見積コストの大きいアクティビティーがしきい値によって実行を阻止されていることを確認する方法を示しました。こうしたしきい値違反を調べた後、大規模なこれらの照会を生

み出している SQL ステートメント・テキストを判別します。こうすることで、照会先の表の索引が必要かどうかを Explain 機能を使用して判断できます。

この追加情報を収集するには、次のようにアクティビティー・イベント・モニターを作成して活動化し、しきい値を変更して詳細情報を含んだアクティビティー・コレクションをオンにする必要があります。

```
CREATE EVENT MONITOR DB2ACTIVITIES
  FOR ACTIVITIES WRITE TO TABLE

SET EVENT MONITOR DB2ACTIVITIES STATE 1

ALTER THRESHOLD TH1
  WHEN EXCEEDED
  COLLECT ACTIVITY DATA WITH DETAILS
```

もう 1 日営業日が終わってしきい値違反表を再び照会するときは、次のように ACTIVITYSTMT_DB2ACTIVITIES 表との結合を実行して、しきい値に違反したアクティビティーの SQL ステートメント・テキストを調べることができます。

```
SELECT THRESHOLDID,
       SUBSTR(THRESHOLD_PREDICATE, 1, 20) PREDICATE,
       TIME_OF_VIOLATION,
       SUBSTR(STMT_TEXT,1,70) STMT_TEXT
FROM THRESHOLDVIOLATIONS_DB2THRESHOLDVIOLATIONS TV,
     ACTIVITYSTMT_DB2ACTIVITIES A
WHERE TV.APPL_ID = A.APPL_ID
  AND TV.UOW_ID = A.UOW_ID
  AND TV.ACTIVITY_ID = A.ACTIVITY_ID
```

```
THRESHOLDID PREDICATE            TIME_OF_VIOLATION          STMT_TEXT
----------- -------------------- -------------------------- --------------------------------------------------------------------------
          1 EstimatedSQLCost     2008-09-02-23.04.49.000000 select count(*) from syscat.tables,syscat.tables,syscat.tables

  1 record(s) selected.
```

使用可能なモニター・データ

モニター・データは、ワークロード、サービス・サブクラスとサービス・スーパークラス、ワーク・クラス、およびしきい値キューから使用可能です。このデータを使用して、問題を診断および修正したり、パフォーマンス調整を行ったりすることができます。

ワークロード・モニター・データ

以下の図は、ワークロードで使用可能なモニター情報を示しています。イベント・モニターを使用して、ワークロード内で実行されるアクティビティーに関するワークロード統計および情報を収集することができます。ワークロードの場合は、集約アクティビティー統計も取得できます。表関数を使用して、ワークロード統計およびワークロード・オカレンスに関する情報にリアルタイムでアクセスすることができます。

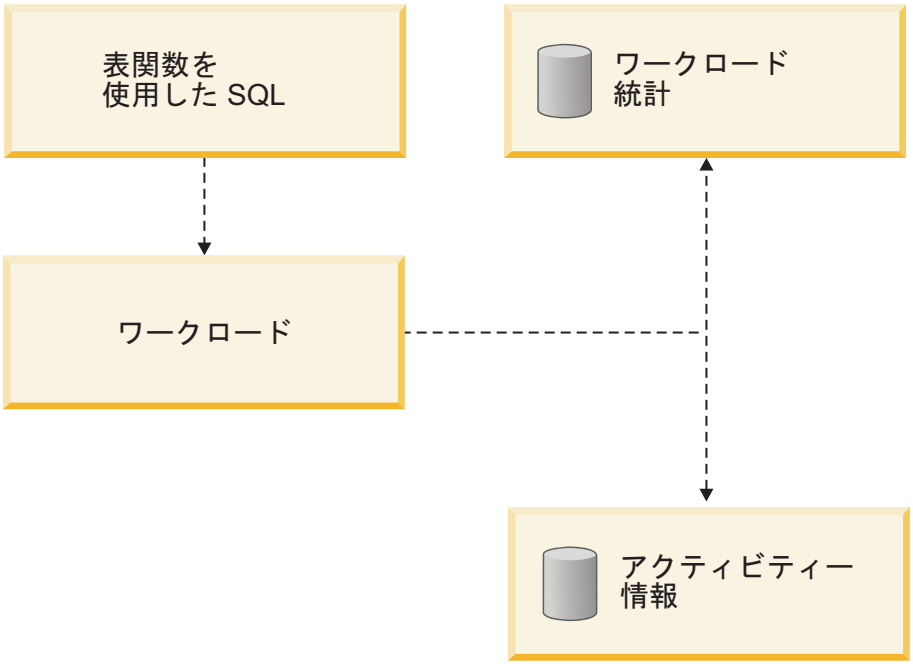


図 47. ワークロードで使用可能なモニター・データ

サービス・クラス・モニター・データ

以下の図は、サービス・クラスで使用可能なモニター情報を示しています。サービス・サブクラスおよびサービス・スーパークラスの統計を収集することができます。サービス・サブクラスの場合、集約アクティビティーおよび要求統計、およびサービス・サブクラスで実行されるアクティビティーに関する情報を入手することもできます。表関数を使用して、サービス・スーパークラスおよびサービス・サブクラスの統計、および特定のサービス・クラスで実行中のエージェントに関する情報にリアルタイムでアクセスできます。

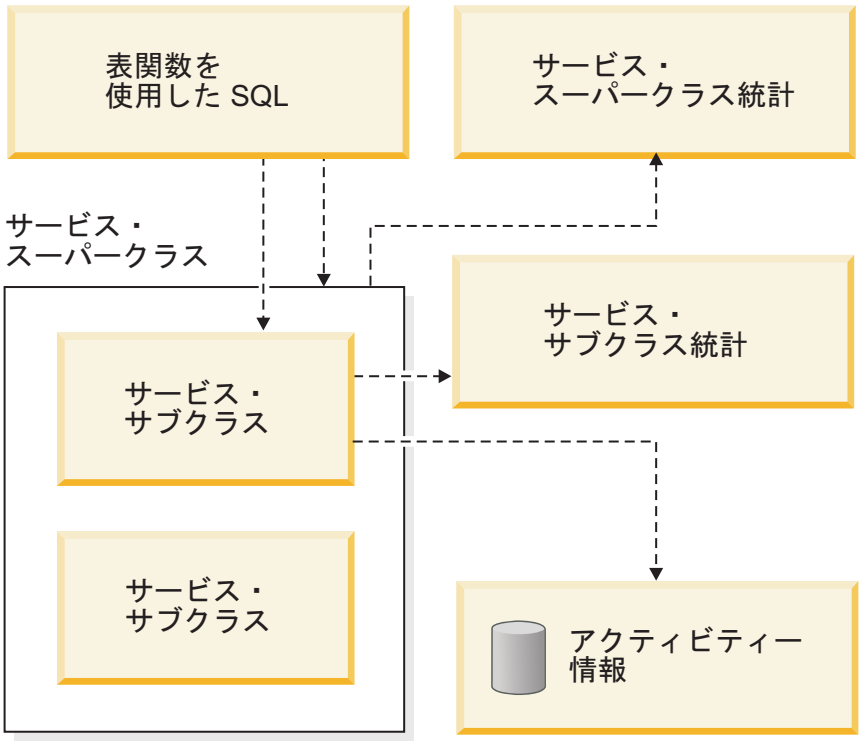


図 48. サービス・クラスで使用可能なモニター・データ

ワーク・クラス・モニター・データ

以下の図は、ワーク・クラスで使用可能なモニター情報を示しています。ワーク・クラスの統計、および特定のワーク・クラスに関連付けられたアクティビティーに関する情報を収集することができます。表関数を使用して、ワーク・クラスの統計にリアルタイムでアクセスすることができます。

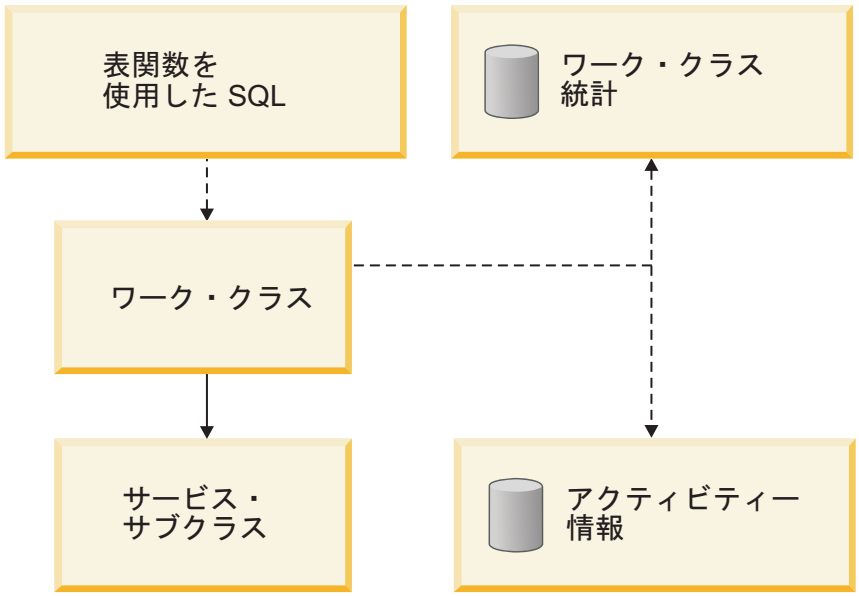


図 49. ワーク・クラスで使用可能なモニター・データ

しきい値モニター・データ

以下の図は、しきい値で使用可能なモニター情報を示しています。しきい値の違反、しきい値の違反の原因となったアクティビティー、およびキューイング統計 (キューイングしきい値用) に関する情報を入手できます。表関数を使用して、キューイングしきい値の統計にリアルタイムでアクセスすることができます。

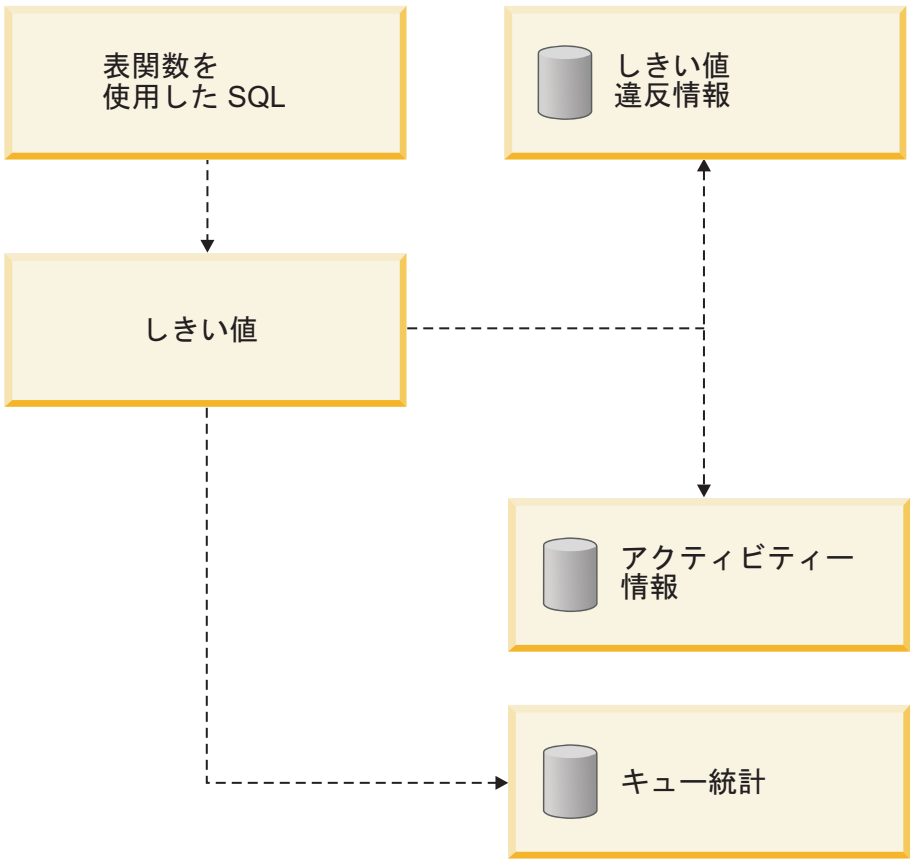


図 50. しきい値で使用可能なモニター・データ

DB2 ワークロード管理用ストアード・プロシージャー

アクティビティーのキャンセル、アクティビティーに関する詳細情報のキャプチャー、DB2 ワークロード管理オブジェクトに関する統計のリセット、およびデータ・サーバーで使用されるクライアント情報の設定に、ストアード・プロシージャーを使用できます。

以下のストアード・プロシージャーを DB2 ワークロード管理と共に使用できます。

WLM_CANCEL_ACTIVITY(*application_handle*, *uow_id*, *activity_id*)。
このストアード・プロシージャーは、実行中またはキューに入れられたアクティビティーをキャンセルするために使用します。アクティビティーは、そのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID

で識別されます。どのタイプのアクティビティーもキャンセルすることができます。キャンセルされたアクティビティーが含まれるアプリケーションはエラー SQL4725N を受け取ります。

WLM_CAPTURE_ACTIVITY_IN_PROGRESS(*application_handle***, *uow_id*, *activity_id*)。**

このストアード・プロシージャーを使用して、現在実行中の個別のアクティビティーに関する情報をアクティビティー・イベント・モニターに送信します。このストアード・プロシージャーは、アクティビティーが完了するまで待機するのではなく、情報を即時に送信します。

WLM_COLLECT_STATS()

このストアード・プロシージャーは、DB2 ワークロード管理オブジェクトの統計を収集およびリセットするために使用します。サービス・クラス、ワークロード、しきい値キュー、およびワーク・アクション・セット用に追跡されたすべての統計は、アクティブな統計イベント・モニターに送信され (存在する場合)、リセットされます。アクティブな統計イベント・モニターが存在しない場合、統計のリセットのみが行われ、収集は行われません。

WLM_SET_CLIENT_INFO(*client_userid***, *client_wrkstnname*,*client_applname*, *client_acctstr*,*client_workload*)。**

このプロシージャーは、現在接続を使用中のアプリケーションまたはエンド・ユーザーの ID を記録するのにデータ・サーバーで使用されるクライアント情報属性を設定するために使用します。アプリケーションまたはユーザーとデータ・サーバーの間にミドルウェアが存在する場合には、WLM_SET_CLIENT_INFO プロシージャーを使用して特徴的な接続属性を明示的に設定してください。

DB2 ワークロード管理オブジェクトの統計

サービス・クラス、ワーク・クラス、ワークロード、およびしきい値キューを含む、DB2 ワークロード管理オブジェクトの統計が維持されます。これらの統計はメモリー内に常駐しており、DB2 ワークロード管理統計の表関数を使用してリアルタイムで表示することが可能です。あるいは、統計を収集して統計イベント・モニターに送信し、後で履歴分析を行うときに表示することもできます。

なお、統計イベント・モニターによってモニター・メトリックを取得することもできます。このトピックでは DB2 ワークロード管理に固有の統計だけが取り上げられているので、それらについては説明されていません。

統計がイベント・モニターに送信されると、メモリー内の値はリセットされ、重複したデータが後続の収集間隔において収集されないようにします。 DB2 ワークロード管理統計の表関数は現在のメモリー内の値を報告するため、収集の後はリセットされた値が報告されます。DB2 ワークロード管理の表関数は統計のサブセットのみを報告します。完全な統計のセットを表示するには、統計を収集して、それらを統計イベント・モニターに送信する必要があります。

集約アクティビティー・データ統計の収集

以下の統計は、各メンバーの所定のオブジェクトに関して維持されます。これは、そのオブジェクトを作成または変更したときに指定した COLLECT AGGREGATE ACTIVITY DATA オプションの値に関わりなく、そのようになります。

表 52. COLLECT AGGREGATE ACTIVITY DATA 設定に関係なく収集されるデータベース・オブジェクトの統計

データベース・オブジェクト	統計	説明
サービス・サブクラス	1. 並行アクティビティーの上限 (concurrent_act_top)。 2. 完了したコーディネーター・アクティビティーの合計 (coord_act_completed_total)。 3. 打ち切られたコーディネーター・アクティビティーの合計 (coord_act_aborted_total)。 4. 拒否されたコーディネーター・アクティビティーの合計 (coord_act_rejected_total)。 5. アクティブな要求の数 (num_requests_active)。 6. マッピングされるアクティビティーの数およびマッピングの際に除外されるアクティビティーの数 (act_remapped_in および act_remapped_out)。	1. このアクティビティー並行性最高水準点は、統計を収集した時間間隔中に、特定のメンバーで特定のサービス・クラスが達した、アクティビティーの最高の並行性 (ネストされたアクティビティーを含む) を判別するために使用します。 2. この統計は、サービス・クラスで実行中の作業の量を判別するために使用します。 3. この統計は、正常に完了しなかったアクティビティーを測定し、システムがどれほど正常稼働しているかを判別するために使用します。アクティビティーは、取り消し、エラー、または再アクティブしきい値が原因で打ち切られる可能性があります。 4. この拒否されたネストなしコーディネーター・アクティビティー・カウントは、アクティビティーの拒否を測定し、拒否ポリシーの有効性の指標を入手するために使用します。アクティビティーが STOP EXECUTION アクションを持つ予測しきい値に違反した場合、あるいはワーク・アクションによって実行を妨げられた場合、そのアクティビティーは拒否されたものとして数えられます。 5. この統計は、サービス・クラスで現在実行中の要求の数を判別するために使用します。 6. これらの統計は、継続中のアクティビティーの優先度変更の一部としてサービス・サブクラスに対して再マッピングまたは再マッピングの際に除外されるアクティビティーの数を判別するために使用します。
サービス・スーパークラス	同時接続の上限 (concurrent_connection_top)。	このコーディネーター接続並行性最高水準点は、接続の並行性しきい値を調整するために使用します。

表 52. COLLECT AGGREGATE ACTIVITY DATA 設定に関係なく収集されるデータベース・オブジェクトの統計 (続き)

データベース・オブジェクト	統計	説明
ワークロード	1. 並行ワークロード・オカレンスの上限 (concurrent_wlo_top)。 2. 並行ワークロード・オカレンス・アクティビティーの上限 (concurrent_wlo_act_top)。 3. 完了したコーディネーター・アクティビティーの合計 (coord_act_completed_total)。 4. 打ち切られたコーディネーター・アクティビティーの合計 (coord_act_aborted_total)。 5. 拒否されたコーディネーター・アクティビティーの合計 (coord_act_rejected_total)。 6. 完了したワークロード・オカレンスの合計 (wlo_completed_total)。 7. アクティビティーの合計 (act_total)。	1. このワークロード・オカレンス最高水準点は、並行ワークロード・オカレンスの最大数を識別するために、あるいは並行して実行中のワークロード・オカレンス数が多すぎる (つまり、同じワークロード定義に関連付けられ、システム上で同時に実行しているアプリケーションが多すぎる) 場合にワークロード・オカレンスの並行性しきい値を設定または調整するために使用します。 2. このエレメントを使用して、収集された時間間隔にこのワークロードの任意のオカレンス用のメンバーで到達した並行アクティビティーの最大数を調べることができます。 3. この統計は、アクティビティーの正常完了率を測定し、システムの正常性の指標を入手するために使用します。 4. この統計は、正常に完了しなかったアクティビティーを測定し、システムがどれほど正常稼働しているかを判別するために使用します。アクティビティーは、取り消し、エラー、または再アクティブしきい値が原因で打ち切られる可能性があります。 5. この統計は、アクティビティーの拒否率を測定し、拒否ポリシーの有効性を判別するために使用します。アクティビティーが STOP EXECUTION アクションを持つ予測しきい値に違反した場合、あるいはワーク・アクションによって実行を妨げられた場合、そのアクティビティーは拒否されたものとして数えられます。 6. この統計は、特定の期間に完了したワークロードのオカレンスの数を判別するために使用します。 7. この統計は、ワーク・アクション・セットの有効性を判別したり、システム上の各アクティビティー・タイプの相対的な割合を判別したりするために使用します。
ワーク・クラス (ワーク・アクションを介した)	キューの割り当ての合計 (queue_assignments_total)。	この統計は、過剰なキューイングが発生していないかどうか、あるいは適正な数のアクティビティーがキューイングされているかどうか (つまり、並行性しきい値による制限が過度または不十分ではないか) を判別するために使用します。

表 52. COLLECT AGGREGATE ACTIVITY DATA 設定に関係なく収集されるデータベース・オブジェクトの統計 (続き)

データベース・オブジェクト	統計	説明
しきい値キュー	1. キュー・サイズの上限 (queue_size_top)。 2. キュー時間の合計 (queue_time_total)。	1. この統計は、最大キュー・サイズを決定したり、キュー・サイズが十分であるかどうかを識別したりするために使用します。 2. この統計は、アクティビティーがキューで費やす時間や、その時間が過剰でないかを判別するために使用します。

サービス・サブクラス、ワークロード、またはワーク・クラス (ワーク・アクションを介した) に対して COLLECT AGGREGATE ACTIVITY DATA オプションの値を BASE に設定すると、メンバーごとに、以下の統計の一部も収集されるか、対応するヒストグラムが生成されます。平均を使用すると、アクティビティーがどこで時間の大半を費やすか (例えば、キューで、あるいは実行時に)、およびその応答時間 (存続時間) を素早く把握することができます。さらに、平均を使用してヒストグラム・テンプレートを調整することもできます。つまり、真の平均とヒストグラムから計算された平均とを比較して、ヒストグラムからの平均が真の平均から外れている場合、自分のデータにとってより適切なビン値のセットを使用して、対応するヒストグラムのヒストグラム・テンプレートを変更することを考慮できます。

表 53. COLLECT AGGREGATE ACTIVITY DATA を BASE に設定した場合に収集される統計またはヒストグラム

統計またはヒストグラム	説明
平均要求実行時間 (request_exec_time_avg)。	この統計は、サービス・クラスに関連付けられた要求の実行時間の算術平均を判別するために使用します。
コーディネーター・アクティビティーの平均存続期間 (coord_act_lifetime_avg)。	この統計は、サービス・クラス、ワークロード、またはワーク・クラスに関連付けられた、ネストなしコーディネーター・アクティビティーの存続期間の算術平均を判別するために使用します。
コーディネーター・アクティビティーの平均実行時間 (coord_act_exec_time_avg)。	この統計は、サービス・クラス、ワークロード、またはワーク・クラスに関連付けられた、ネストなしコーディネーター・アクティビティーの実行時間の算術平均を判別するために使用します。
コーディネーター・アクティビティー・キューの平均時間 (coord_act_queue_time_avg)。	この統計は、サービス・クラス、ワークロード、またはワーク・クラスに関連付けられた、ネストなしコーディネーター・アクティビティーのキュー時間の算術平均を判別するために使用します。
コスト見積もりの上限 (cost_estimate_top)。	この統計は、見積コストのしきい値を調整するために使用します。
返される実際の行数の上限 (rows_returned_top)。	この情報は、返される実際の行数のしきい値を調整するために使用します。

表 53. COLLECT AGGREGATE ACTIVITY DATA を BASE に設定した場合に収集される統計またはヒストグラム (続き)

統計またはヒストグラム	説明
集約 TEMPORARY 表スペースの上限 (agg_temp_tablespace_top)。	この統計は、集約 SYSTEM TEMPORARY 表スペースの使用量を調整するために使用します。 この統計は、集約 TEMPORARY 表スペース使用量のしきい値を定義した場合にのみモニターされます。サービス・サブクラスの場合は、サービス・サブクラス自体に AGGSQLTEMPSPACE しきい値を定義すると、この統計が必ずモニターされます。また、同じスーパークラス内のサービス・サブクラスに同様のしきい値を定義したときもモニターされます。
TEMPORARY 表スペースの上限 (temp_tablespace_top)。	この統計は、TEMPORARY 表スペース使用量のしきい値を調整するために使用します。 この統計は、TEMPORARY 表スペース使用量のしきい値を定義した場合にのみモニターされます。サービス・サブクラスの場合は、サービス・サブクラス自体に AGGSQLTEMPSPACE しきい値を定義すると、このしきい値も必ずモニターされます。また、同じスーパークラス内のサービス・サブクラスに同様のしきい値を定義したときもモニターされます。
コーディネーター・アクティビティー存続期間 (CoordActLifetime) ヒストグラム。	このヒストグラムは、システム・パフォーマンスの全体像を把握するために使用します。 このヒストグラムは、ネストなしコーディネーター・アクティビティーのアクティビティー発生時刻から終了時刻までの時間 (ミリ秒) を収集します。 アクティビティーが、終了後もカーソルを開いたままにするルーチンの場合、存続期間ヒストグラムは、カーソルの存続期間を、カーソルの親であるルーチンの存続期間にカウントしません。

表 53. COLLECT AGGREGATE ACTIVITY DATA を BASE に設定した場合に収集される統計またはヒストグラム (続き)

統計またはヒストグラム	説明
コーディネーター・アクティビティー実行時間 (CoordActExecTime) ヒストグラム。	このヒストグラムは、実行時間に影響する、システムに対する変更の影響を測定するために使用します。 このヒストグラムは、ネストなしコーディネーター・アクティビティーの実行時間 (ミリ秒) を収集します。 実行時間は以下のように計算されます。 • カーソルの場合、実行時間はオープン・カーソル要求、フェッチ、およびクローズ・カーソル要求を組み合わせた時間になります。カーソルがアイドル中の時間は実行時間にカウントされません。 • ルーチンの場合、実行時間はルーチン呼び出しの開始から終了までの時間です。ルーチンの終了後もカーソルが開かれたままの場合、これらのカーソルの存続期間はルーチンの実行時間にカウントされません。 • 他のすべてのアクティビティーの場合、実行時間は、アクティビティーの存続期間とアクティビティーがキューで費やした時間との差になります。
コーディネーター・アクティビティー・キュー時間 (CoordActQueueTime) ヒストグラム。	このヒストグラムを使用して、アクティビティーに対するキューイングしきい値の影響を測定します。 このヒストグラムは、ネストなしコーディネーター・アクティビティーがキューで費やす時間 (ミリ秒) を収集します。

サービス・サブクラス、ワークロード、またはワーク・クラスに対して COLLECT AGGREGATE ACTIVITY DATA オプションの値を EXTENDED に設定すると、対応するサービス・クラスまたはワーク・クラス (ワーク・アクションを介した) に関して、メンバーごとに、以下のシステム統計が収集されるか、以下のヒストグラムが生成されます。平均を使用すると、アクティビティーの発生の平均比率 (発生比率は発生間隔時間の反対) およびアクティビティーのコスト (見積コスト) について素早く理解することができます。さらに、平均を使用してヒストグラム・テンプレートを調整することもできます。つまり、真の平均とヒストグラムから計算された平均とを比較して、ヒストグラムからの平均が真の平均から外れている場合、自分のデータにとってより適切なビン値のセットを使用して、対応するヒストグラムのヒストグラム・テンプレートを変更することを考慮できます。 EXTENDED 統計は、より詳細なパフォーマンスのモデル化に役立ちます。 318 ページの『ワークロード管理のパフォーマンスのモデル化』も参照してください。

表 54. COLLECT AGGREGATE ACTIVITY DATA を EXTENDED に設定した場合に収集される統計またはヒストグラム

統計またはヒストグラム	説明
コーディネーター・アクティビティーの平均見積コスト (coord_act_est_cost_avg)。	この統計は、このサービス・サブクラス、ワークロード、またはワーク・クラスに関連付けられた、ネスト・レベル 0 のコーディネーター DML アクティビティーの、最後に統計がリセットされた時以降の見積もりコストの算術平均を判別するために使用します。
ネストなしコーディネーター・アクティビティーの発生間隔時間の平均 (coord_act_interarrival_time_avg)。	この統計は、このサービス・クラス、ワークロード、またはワーク・クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーが発生してから、次のコーディネーター・アクティビティーが発生するまでの時間の算術平均を判別するために使用します。統計が最後にリセットされた時からの平均が計算されます。
コーディネーター・アクティビティー見積コスト (CoordActEstCost) ヒストグラム。	このヒストグラムを使用して、概算のサービス時間分布を入手します。 このヒストグラムは、ネストなしコーディネーター・アクティビティーの見積コスト (timeron 単位) を収集します。このデータは、システムのモデル化に、あるいはパフォーマンス・モデル化アプリケーションへの入力に役立ちます。
コーディネーター・アクティビティー発生間隔時間 (CoordActInterArrivalTime) ヒストグラム。	このヒストグラムを使用して、ネストなしコーディネーター・アクティビティーの発生間隔時間分布を入手します。 このヒストグラムは、ネストなしコーディネーター・アクティビティーの発生間隔時間 (ミリ秒) を収集します。このデータは、システムのモデル化に、あるいはパフォーマンス・モデル化アプリケーションへの入力に役立ちます。

次の表は、収集されるアクティビティー統計を DB2 ワークロード管理オブジェクトごとにまとめたリファレンスです。表関数とイベント・モニターの両方から使用できるすべての集約統計が含まれています。オブジェクトによっては、常に収集される統計もあります。他の統計は、特定の COLLECT AGGREGATE オプションが指定された場合にのみ収集されます。集約アクティビティー統計の場合、COLLECT AGGREGATE ACTIVITY DATA EXTENDED が指定されると、すべての BASE 集約アクティビティー統計も収集されます。

表 55. DB2 ワークロード管理オブジェクトの集約アクティビティー統計の収集

オブジェクト・タイプ	デフォルトで常に収集されるアクティビティー統計	COLLECT AGGREGATE ACTIVITY DATA BASE を指定した場合に収集されるアクティビティー統計	COLLECT AGGREGATE ACTIVITY DATA EXTENDED を指定した場合に収集されるアクティビティー統計
サービス・サブクラス	act_remapped_in act_remapped_out concurrent_act_top coord_act_completed_total coord_act_rejected_total coord_act_aborted_total	agg_temp_tablespace_top coord_act_exec_time_avg coord_act_lifetime_avg coord_act_lifetime_top coord_act_queue_time_avg coord_act_lifetime_stddev coord_act_exec_time_stddev coord_act_queue_time_stddev CoordActLifetime histogram CoordActExecTime histogram CoordActQueueTime histogram cost_estimate_top rows_returned_top temp_tablespace_top	coord_act_est_cost_avg coord_act_interarrival_time_avg CoordActEstCost histogram CoordActInterArrivalTime histogram
サービス・スーパークラス	concurrent_connection_top	N/A	N/A
ワークロード	concurrent_wlo_act_top concurrent_wlo_top coord_act_aborted_total coord_act_completed_total coord_act_rejected_total wlo_completed_total	coord_act_exec_time_avg coord_act_lifetime_top coord_act_lifetime_avg coord_act_queue_time_avg coord_act_lifetime_stddev coord_act_exec_time_stddev coord_act_queue_time_stddev CoordActLifetime histogram CoordActExecTime histogram CoordActQueueTime histogram cost_estimate_top rows_returned_top temp_tablespace_top	coord_act_est_cost_avg coord_act_interarrival_time_avg CoordActEstCost histogram CoordActInterArrivalTime histogram
ワーク・クラス (ワーク・アクションを介した)	act_total	agg_temp_tablespace_top coord_act_lifetime_top coord_act_lifetime_avg coord_act_exec_time_avg coord_act_queue_time_avg CoordActLifetime histogram CoordActExecTime histogram CoordActQueueTime histogram cost_estimate_top rows_returned_top temp_tablespace_top	coord_act_est_cost_avg coord_act_interarrival_time_avg CoordActEstCost histogram CoordActInterArrivalTime histogram

表 55. DB2 ワークロード管理オブジェクトの集約アクティビティー統計の収集 (続き)

オブジェクト・タイプ	デフォルトで常に収集されるアクティビティー統計	COLLECT AGGREGATE ACTIVITY DATA BASE を指定した場合に収集されるアクティビティー統計	COLLECT AGGREGATE ACTIVITY DATA EXTENDED を指定した場合に収集されるアクティビティー統計
しきい値	N/A	N/A	N/A
しきい値キュー	queue_assignments_total queue_size_top queue_time_total	N/A	N/A

集約要求データ統計の収集

サービス・サブクラスの COLLECT AGGREGATE REQUEST DATA オプションの値を BASE に設定すると、以下の統計がサービス・サブクラス用に維持されます。

表 56. COLLECT AGGREGATE REQUEST DATA を BASE に設定した場合に収集される統計またはヒストグラム

統計またはヒストグラム	説明
要求実行時間の平均 (request_exec_time_avg)。	この統計は、メンバーでの各要求の処理に費やされる平均時間を把握し、対応する要求実行時間のヒストグラムのヒストグラム・テンプレートを調整するために使用します。

表 56. COLLECT AGGREGATE REQUEST DATA を BASE に設定した場合に収集される統計またはヒストグラム (続き)

統計またはヒストグラム	説明
要求実行時間 (ReqExecTime) ヒストグラム	このヒストグラムは、どこで作業が実行されているか、およびメンバー間で作業の分散が均等であるかどうかを把握するために使用します。 このヒストグラムは、サービス・クラスで実行されている作業のボリュームと、データベース・メンバー全体におけるこの作業の分布を示します。要求実行時間 (ミリ秒) は、メンバーごとにすべての要求に関して収集され、ヒストグラムにまとめられます。 このヒストグラムには、コーディネーター・メンバーの要求と、存在する場合はコーディネーター・メンバーと非コーディネーター・メンバー両方の (RPC 要求または SMP サブエージェント要求のような) 副要求が含まれます。含まれる要求には、アクティビティーに関連付けられるものとそうでないものがあります。例えば、このヒストグラムには PREPARE 要求と OPEN 要求の両方が含まれています。しかし、OPEN 要求は常にカーソル・アクティビティーと関連付けられるのに対し、PREPARE 要求はアクティビティーの一部ではありません。 要求実行時間は、サービス・クラスで作業するエージェントが費やす時間を概算したものです。例えば、コーディネーター・アクティビティー・カウントは、大半のユーザー・アクティビティーが 1 つのメンバーで発生していることを示しているかもしれませんが、コーディネーター・エージェントは、アクティビティーの処理の一環として、作業の大半が実行される別のメンバーに副要求を送る場合があります。 要求実行時間のヒストグラムは、メンバーに送信される要求のサイズ、つまり、メンバーに送信される作業の大半を構成しているのが小さな要求あるいは大きな要求のどちらか、または特定の分散がないかどうかを判別するのに役立ちます。 要求実行時間のヒストグラムをアクティビティー応答時間の分析で使用すべきではありません。アクティビティーはいくつかの要求および副要求で構成されている場合があること、要求とアクティビティーの実行時間の間には 1 対 1 のマッピングがないこと、そしてすべての要求がアクティビティーに関連付けられているわけではないことがその理由です。

次の表は、収集される要求統計を DB2 ワークロード管理オブジェクトごとにまとめたリファレンスです。表関数とイベント・モニターの両方から使用できるすべての集約統計が含まれています。オブジェクトによっては、常に収集される統計もあります。他の統計は、COLLECT AGGREGATE REQUEST DATA オプションが指定された場合にのみ収集されます。

表 57. DB2 ワークロード管理オブジェクトの集約要求統計の収集

オブジェクト・タイプ	デフォルトで常に収集される要求統計	COLLECT AGGREGATE REQUEST DATA BASE を指定した場合に収集される要求統計
サービス・サブクラス	num_requests_active	request_exec_time_avg request_exec_time_stddev request_exec_time_total ReqExecTime histogram
サービス・スーパークラス	N/A	N/A
ワークロード	N/A	N/A
ワーク・クラス (ワーク・アクションを介した)	N/A	N/A
しきい値	N/A	N/A
しきい値キュー	N/A	N/A

再マップされたアクティビティーがある場合の統計収集とモニター

統計を収集する方法とモニターする方法はどちらも、サービス・サブクラス間のアクティビティーの動的再マップの影響を受けます。

1 つのサービス・クラスで実行されているアクティビティーが、実行を続行するために別のサービス・クラスに移動されたときに、再マップが行われます。優先度変更のアプローチと同じように、この再マップは、CPUTIMEINSC などのしきい値を使用して実行でき、ワークロード管理構成の不可欠な部分にできます。

再マップされたアクティビティーの影響を受ける統計

この規則の例外として、まず、アクティビティーの発生間隔時間、見積コスト、キュー時間はすべて、アクティビティーが実行を終了するサブクラスではなく、アクティビティーが実行を開始するサブクラスに関連付けられます。再マップされたアクティビティーは両方のサブクラスの統計収集に影響を与えるため、発生間隔時間、見積コスト、またはキュー時間のヒストグラムでカウントされるアクティビティーの数は、存続期間または実行時間のヒストグラムにおける数と異なる場合があります。

例えば、サービス・サブクラス A で実行を開始し、その後サービス・サブクラス B に再マップされ、そこで実行を終了するアクティビティーがあるとします。このアクティビティーの見積コストはサービス・サブクラス A に関連付けられますが、その存続期間はサービス・サブクラス B に関連付けられます。その結果として、サブクラス A では、見積コスト・ヒストグラムでカウントされるエレメント数は存続期間ヒストグラムでカウントされる数より 1 つ多く、サービス・サブクラス B では、存続期間ヒストグラムでカウントされるエレメント数は見積コスト・ヒストグラムでカウントされる数より 1 つ多くなります。

規則のもう 1 つの例外として、モニター・エレメント **concurrent_act_top** はアクティビティーが通るサブクラスで更新可能であり、そこに帰属することができます。アクティビティーの開始時に増分されてアクティビティーの終了時に減分される以外に、このモニター・エレメントはアクティビティーがサブクラスにマップされると増分され、アクティビティーがサブクラスからマップアウトされる (別のサ

ブクラスにマップされる) と減分されます。

アクティビティーの再マップに関する統計

act_remapped_in と **act_remapped_out** の 2 つのモニター・エレメントを使用して、再マップ・アクションによってサービス・サブクラスに出入りするアクティビティーの数をカウントできます。 **act_remapped_in** および **act_remapped_out** モニター・エレメントは、あるパーティションのある特定のサブクラスに関して、最後のリセット以降そのサブクラスに対してマップインまたはマップアウトされたアクティビティーの数をカウントします。これらのモニター・エレメントを使用して、サービス・サブクラス間のアクティビティーの再マップが正しく行われているかどうかを確認できます。

再マップ・アクションのターゲットであるソース・サービス・サブクラスと宛先サービス・サブクラスを判別するには、しきい値違反イベント・モニター・レコードを参照します。このレコードには、宛先サービス・クラス ID (destination_service_class_id) が含まれています。ソース・サービス・クラスについても、このしきい値違反レコードを使用して判別できます。

アクティビティーの再マップが行われる場合のモニター

別のサブクラスへのアクティビティーの再マップは、そのアクティビティーをモニターする方法に影響を与えます。再マップのために開始サービス・クラスと終了サービス・クラスが異なるアクティビティーの統計がすべて収集されるようにするには、サービス・クラスを作成または変更するときに、アクティビティーが実行を開始するサービス・サブクラスとアクティビティーが実行を終了するサービス・サブクラスの両方とも集約アクティビティー・データ収集をオンにします。アクティビティーが開始されるサービス・サブクラスの集約アクティビティー・データ収集のみをオンにすると、アクティビティーが寄与するのはキュー時間統計のみになります。拡張統計の場合には、見積コスト統計と発生間隔時間統計にも寄与します。アクティビティーが実行を終了するサービス・サブクラスの集約アクティビティー・データ収集のみをオンにすると、CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントを発行するときに指定したオプションが COLLECT AGGREGATE DATA BASE であるか COLLECT AGGREGATE DATA EXTENDED であるかにかかわらず、アクティビティーが寄与するのは存続期間統計と実行時間統計のみになります。

以下の表は、再マップが統計収集に与える影響と収集の設定についての要約です。

表 58. 再マップに関係するサブクラスの集約統計収集に COLLECT AGGREGATE DATA BASE オプションが与える影響

統計	開始サブクラスの収集設定および終了サブクラスの収集設定			
	NONE および NONE	BASE および NONE	NONE および BASE	BASE および BASE
存続期間	非収集	非収集	収集	収集
キュー時間	非収集	収集	非収集	収集
実行時間	非収集	非収集	収集	収集

表 59. 再マップに関係するサブクラスの集約統計収集に COLLECT AGGREGATE DATA EXTENDED オプションが与える影響

統計	開始サブクラスの収集設定および終了サブクラスの収集設定			
	NONE および NONE	EXTENDED および NONE	NONE および EXTENDED	EXTENDED および EXTENDED
存続期間	非収集	非収集	収集	収集
キュー時間	非収集	収集	非収集	収集
実行時間	非収集	非収集	収集	収集
発生間隔時間	非収集	収集	非収集	収集
見積コスト	非収集	収集	非収集	収集

表 60. 再マップに関係するサブクラスの集約統計収集に COLLECT AGGREGATE DATA BASE および COLLECT AGGREGATE DATA EXTENDED オプションの混在が与える影響

統計	開始サブクラスの収集設定および終了サブクラスの収集設定	
	BASE および EXTENDED	EXTENDED および BASE
存続期間	収集	収集
キュー時間	収集	収集
実行時間	収集	収集
発生間隔時間	非収集	収集
見積コスト	非収集	収集

ワークロード管理のヒストグラム

ヒストグラムはビンのコレクションです。ビンは、データの離散的範囲を収集するためのコンテナーです。ヒストグラムは、様々なワークロード分析およびパフォーマンスのチューニング・タスクに役立ちます。

DB2 ワークロード管理のヒストグラムには 41 のビンがあり、この数は固定です。40 番目のビンにはヒストグラムの最高の定義値が含まれますが、41 番目のビンは最高の定義値を超える値のためのものです。それぞれのビンは特定の範囲の値を表し、ビンの範囲は対数目盛りに従います。それぞれのビンが表す範囲は、1 番目のビンから 40 番目のビンに進むにつれて徐々に大きくなります。以下の図は、アクティビティー存続時間を棒グラフで作図したヒストグラムを示しています。

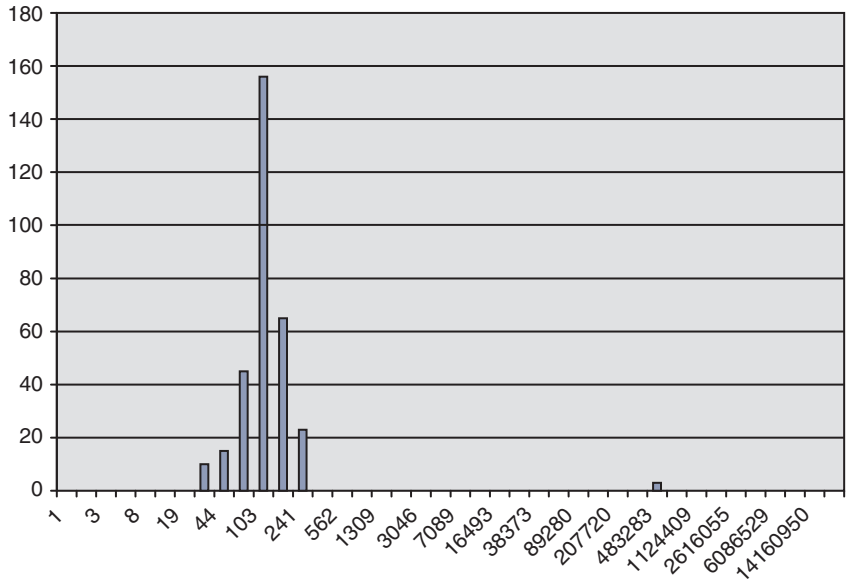


図 51. アクティビティー存続時間を棒グラフで作図したヒストグラム

アクティビティー存続時間ヒストグラムは、以下のデータに対応しています。それぞれのカウントは、存続時間 (ミリ秒) がビンの下限値からビンの上限値までの範囲に含まれるアクティビティーの数を表しています。例えば、156 個のアクティビティーは、存続時間が 68 ミリ秒から 103 ミリ秒の範囲内でした。

ビンの下限	ビンの上限	カウント
0	1	0
1	2	0
2	3	0
3	5	0
5	8	0
8	12	0
12	19	0
19	29	10
29	44	15
44	68	45
68	103	156
103	158	65
158	241	23
241	369	0
369	562	0
562	858	0
858	1309	0
1309	1997	0
1997	3046	0
3046	4647	0
4647	7089	0
7089	10813	0
10813	16493	0
16493	25157	0
25157	38373	0
38373	58532	0
58532	89280	0
89280	136181	0
136181	207720	0
207720	316840	0
316840	483283	3
483283	737162	0
737162	1124409	0
1124409	1715085	0

```
 1715085   2616055     0
 2616055   3990325     0
 3990325   6086529     0
 6086529   9283913     0
 9283913  14160950     0
14160950  21600000     0
21600000  無限大        0
```

ヒストグラムを多数のさまざまな目的に使用することができます。例えば、ヒストグラムを使用して値の分布を調べたり、範囲外にある値を識別したり、または平均および標準偏差を計算したりすることができます。ワークロードのより良い理解と特徴付けにヒストグラムを使用する方法の例については、405 ページの『シナリオ: キャパシティー・プランニング情報が使用できない場合の DB2 ワークロード管理構成のチューニング』および 300 ページの『例: DB2 ワークロード管理構成におけるヒストグラムからの平均および標準偏差の計算』を参照してください。

マルチメンバー・データベース環境では、ヒストグラムはメンバーごとに収集されます。ヒストグラムの各ビンの値の範囲はすべてのデータベース・メンバーで同じですが、ビンあたりのカウントはメンバーごとに固有の数になります。ビンを使用して、メンバーごとに情報を分析することができます。対応するビンのカウントを加算することによってすべてのデータベース・メンバーからのヒストグラムを結合し、この単一ヒストグラムを使用してデータの全体像を得ることもできます。これを、全体的な平均や標準偏差の計算などの作業に使用できます。

ヒストグラムは、サービス・サブクラス、ワークロード、およびワーク・クラスに、ワーク・アクションを介して使用できます。オブジェクトの作成または変更時に COLLECT AGGREGATE ACTIVITY DATA、COLLECT AGGREGATE REQUEST DATA、または COLLECT AGGREGATE UNIT OF WORK DATA 節の 1 つを指定すると、これらのオブジェクトのためにヒストグラムが収集されます。ワーク・クラスの場合は、COLLECT AGGREGATE ACTIVITY DATA ワーク・アクションをワーク・クラスに適用するとヒストグラムも収集されます。以下のヒストグラムが使用可能です。

- ネストなしコーディネーター・アクティビティー存続時間 (サービス・サブクラス、ワークロード、またはワーク・クラスに適用されるワーク・アクションに対して、AGGREGATE ACTIVITY DATA BASE または AGGREGATE ACTIVITY DATA EXTENDED を指定した場合)。
- ネストなしコーディネーター・アクティビティー実行時間 (サービス・サブクラス、ワークロード、またはワーク・クラスに適用されるワーク・アクションに対して、AGGREGATE ACTIVITY DATA BASE または AGGREGATE ACTIVITY DATA EXTENDED を指定した場合)。
- ネストなしコーディネーター・アクティビティー・キュー時間 (サービス・サブクラス、ワークロード、またはワーク・クラスに適用されるワーク・アクションに対して、AGGREGATE ACTIVITY DATA BASE または AGGREGATE ACTIVITY DATA EXTENDED を指定した場合)。
- 要求実行時間 (サービス・サブクラスに対して AGGREGATE REQUEST DATA BASE を指定した場合)。このヒストグラムは、ワークロードまたはワーク・クラスには適用されません。
- ネストなしアクティビティー到着間隔時間ヒストグラム (サービス・サブクラス、ワークロード、またはワーク・クラスに適用されるワーク・アクションに対して、AGGREGATE ACTIVITY DATA EXTENDED を指定した場合)。

- ネストなし DML アクティビティー見積コスト (サービス・サブクラス、ワークロード、またはワーク・クラスに適用されるワーク・アクションに対して、AGGREGATE ACTIVITY DATA EXTENDED を指定した場合)。
- 作業単位の存続期間 (サービス・クラスに対して、 AGGREGATE UNIT OF WORK DATA BASE を指定した場合)。

アクティビティー関連のすべてのヒストグラムに、完了したアクティビティー、取り消されたアクティビティー、拒否されたアクティビティーに関する情報が含まれます。

ヒストグラム・テンプレート

オプションでヒストグラム・テンプレートを指定することができます。このテンプレートを使用して、ビンの上限値など、特定のヒストグラムの外観を決定できます。ヒストグラム・テンプレートは単位なし オブジェクトです。つまり、定義済みの計算単位が割り当てられているわけではありません。ヒストグラム・テンプレートが使用されるコンテキストに応じて、計算単位がサービス・クラス、ワークロード、またはワーク・アクションの作成または変更時に、ヒストグラムに割り当てられます。時間ベースのヒストグラム (例えば、ACTIVITY LIFETIME HISTOGRAM 節によって指定されるものなど) は、ミリ秒単位を使用します。一方、コスト・ベースのヒストグラム (例えば、ACTIVITY ESTIMATED COST HISTOGRAM 節によって指定されるものなど) は、timeron 単位を使用します。

CREATE HISTOGRAM TEMPLATE ステートメントを使用してビンの最大上限値を指定することによって、ヒストグラム・テンプレートを作成することができます。その他のビンはすべて、ビンの上限値に向かって指数関数的に増加する値として、自動的に定義されます。例えば、ビンの上限値を 3 000 000 としてヒストグラム・テンプレートを作成するには、以下のようなステートメントを発行します。

```
CREATE HISTOGRAM TEMPLATE TEMPLATE1 HIGH BIN VALUE 3000000
```

このステートメントは、以下のビンの値を使用してヒストグラム・テンプレートを作成します。

```
ビンの下限  ビンの上限
      0         1
      1         2
      2         3
      3         4
      4         6
      6         9
      9        13
     13        19
     19        28
     28        41
     41        60
     60        87
     87       127
    127       184
    184       268
    268       389
    389       565
    565       821
    821      1192
   1192      1732
   1732      2514
   2514      3651
   3651      5300
```

```
   5300     7696
   7696    11173
  11173    16222
  16222    23553
  23553    34196
  34196    49649
  49649    72084
  72084   104657
 104657   151948
 151948   220609
 220609   320297
 320297   465030
 465030   675163
 675163   980250
 980250  1423197
1423197  2066299
2066299  3000000
3000000   無限大
```

サービス・サブクラス、ワークロード、またはワーク・アクションを作成または変更する時、適切な HISTOGRAM TEMPLATE キーワードを使用することにより、ヒストグラム・テンプレートを適用することができます。ヒストグラム・テンプレートを指定しない場合、デフォルトのテンプレート SYSDEFAULTHISTOGRAM が使用されます。AGGREGATE ACTIVITY DATA コレクションをオブジェクトが使用できるようにしていない場合、ヒストグラム・テンプレートは無視されます。

例えば、サービス・スーパークラス MYSUPERCLASS 下のサービス・サブクラス MYSUBCLASS の既存のアクティビティー存続時間ヒストグラムに TEMPLATE1 ヒストグラム・テンプレートを使用するには、次のステートメントを発行します。

```
ALTER SERVICE CLASS MYSUBCLASS UNDER MYSUPERCLASS
ACTIVITY LIFETIME HISTOGRAM TEMPLATE TEMPLATE1
```

ALTER SERVICE CLASS ステートメントをコミットした後、MYSUBCLASS サービス・サブクラスに関して収集されるアクティビティー存続時間ヒストグラムのビンの上限値は、SYSDEFAULTHISTOGRAM ヒストグラム・テンプレートではなく TEMPLATE1 ヒストグラム・テンプレートによって決定されます。

別のヒストグラム・テンプレートを使用するようサービス・クラスまたはワークロードを変更する場合、あるいはヒストグラム・テンプレートを変更する場合、その変更は統計をリセットするまで有効になりません。

DROP HISTOGRAM TEMPLATE ステートメントを使用してヒストグラム・テンプレートをドロップすることができます。

SYSCAT.HISTOGRAMTEMPLATES ビューを照会することによって、ヒストグラム・テンプレートを表示できます。また、SYSCAT.HISTOGRAMTEMPLATEBINS ビューを照会することによって、対応するヒストグラム・テンプレートのビンの上限値を表示できます。最初のビンの下限値は常に 0 です。他のビンでは、直前のビンの上限値が下限値になります。

例

次の例は、表関数を作成して、サブクラス全体にわたって合計することによってサービス・スーパークラス全体としての
CoordActLifetime、CoordActExecTime、CoordActQueueTime、または CoordActEstCost

ヒストグラムを計算します。サブクラス全体にわたる合計は、実行中のアクティビティーが同一サービス・スーパークラス下の異なるサービス・サブクラスに再マップされる場合に役立ちます。再マップは、サービス・クラス層と特殊なしきい値を使用してアクティビティーのリソースを動的に制御する優先度変更のシナリオで実行されることがあります。この例は CoordActInterArrivalTime ヒストグラムには適用されません。サブクラスの CoordActInterArrivalTime ヒストグラムはそのサブクラスへの照会の到着から次の照会までの時間を計測するが、スーパークラスの CoordActInterArrivalTime ヒストグラムは自分のいずれかのサブクラスへの照会の到着から次の照会までの時間を計測するということが、加重平均の計算で考慮されないためです。

```
CONNECT TO SAMPLE

DROP FUNCTION histsuper

CREATE FUNCTION histsuper(superclass varchar(128),
                          histogram_type varchar(24))
RETURNS TABLE (statistics_timestamp timestamp,
               bin_top integer,
               number_in_bin integer,
               graph varchar(60))
LANGUAGE SQL
READS SQL DATA
NO EXTERNAL ACTION
DETERMINISTIC
RETURN WITH HISTOGRAMS AS
       (SELECT HISTOGRAM_TYPE,
              substr(PARENTSERVICECLASSNAME,1,26) as SUPERCLASS,
              STATISTICS_TIMESTAMP,
              TOP as BIN_TOP,
              sum(NUMBER_IN_BIN) as NUMBER_IN_BIN
       FROM HISTOGRAMBIN_DB2STATISTICS H,
            SYSCAT.SERVICECLASSES S
       WHERE H.SERVICE_CLASS_ID = S.SERVICECLASSID
         AND PARENTSERVICECLASSNAME = histsuper.superclass
         AND HISTOGRAM_TYPE = histsuper.histogram_type
         AND HISTOGRAM_TYPE IN ('CoordActLifetime', 'CoordActExecTime',
     'CoordActQueueTime', 'CoordActEstCost')
       GROUP BY HISTOGRAM_TYPE, PARENTSERVICECLASSNAME, STATISTICS_TIMESTAMP, TOP)
       SELECT STATISTICS_TIMESTAMP,
              BIN_TOP,
              NUMBER_IN_BIN,
              substr(repeat('#', cast(NUMBER_IN_BIN * 60 /
                (SELECT CASE WHEN MAX(NUMBER_IN_BIN) = 0 THEN 1
     ELSE MAX(NUMBER_IN_BIN) END FROM HISTOGRAMS) AS INTEGER)),1,60)
     AS GRAPH FROM HISTOGRAMS

CONNECT RESET
```

出力は以下のようになります。

```
STATISTICS_TIMESTAMP       BIN_TOP     NUMBER_IN_BIN GRAPH
-------------------------- ----------- ------------- ------------------------------------------------------------
2008-11-06-14.47.08.833188 -1          0
2008-11-06-14.47.08.833188 1           1
2008-11-06-14.47.08.833188 2           1
2008-11-06-14.47.08.833188 3           2
2008-11-06-14.47.08.833188 5           4
2008-11-06-14.47.08.833188 8           7
2008-11-06-14.47.08.833188 12          15
2008-11-06-14.47.08.833188 19          29            #
2008-11-06-14.47.08.833188 29          41            #
2008-11-06-14.47.08.833188 44          67            ##
2008-11-06-14.47.08.833188 68          112           ###
2008-11-06-14.47.08.833188 103         228           #####
2008-11-06-14.47.08.833188 158         335           ########
2008-11-06-14.47.08.833188 241         723           ################
2008-11-06-14.47.08.833188 369         1289          ##############################
```

```
2008-11-06-14.47.08.833188 562         1890          ########################################
2008-11-06-14.47.08.833188 858         2484          ##########################################################
2008-11-06-14.47.08.833188 1309        1943          ############################################
2008-11-06-14.47.08.833188 1997        478           ##########
2008-11-06-14.47.08.833188 3046        221           #####
2008-11-06-14.47.08.833188 4647        29            #
2008-11-06-14.47.08.833188 7089        7
2008-11-06-14.47.08.833188 10813       0
2008-11-06-14.47.08.833188 16493       2
2008-11-06-14.47.08.833188 25157       0
2008-11-06-14.47.08.833188 38373       1
2008-11-06-14.47.08.833188 58532       0
2008-11-06-14.47.08.833188 89280       0
2008-11-06-14.47.08.833188 136181      0
2008-11-06-14.47.08.833188 207720      0
2008-11-06-14.47.08.833188 316840      0
2008-11-06-14.47.08.833188 483283      0
2008-11-06-14.47.08.833188 737162      0
2008-11-06-14.47.08.833188 1124409     0
2008-11-06-14.47.08.833188 1715085     0
2008-11-06-14.47.08.833188 2616055     0
2008-11-06-14.47.08.833188 3990325     0
2008-11-06-14.47.08.833188 6086529     0
2008-11-06-14.47.08.833188 9283913     0
2008-11-06-14.47.08.833188 14160950    0
2008-11-06-14.47.08.833188 21600000    0

41 record(s) selected.
```

ヒストグラム・テンプレートの作成

CREATE HISTOGRAM TEMPLATE ステートメントを使用してヒストグラム・テンプレートを作成します。ヒストグラム・テンプレートは、サービス・サブクラスおよびワーク・アクションにより、ヒストグラムで維持される統計のビン値を定義するために使用されます。

始める前に

ヒストグラム・テンプレートを作成するためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、以下のトピックを参照してください。

- 21 ページの『ワークロード管理 DDL ステートメント』
- 559 ページの『付録 A. 一般的な命名規則』

このタスクについて

一部の DB2 サービス・サブクラス、ワーク・クラス・アクティビティー、および要求統計は、ヒストグラムを使用して収集されます。すべてのヒストグラムには特定数のビンがあり、それぞれのビンはアクティビティーまたは要求がカウントされる範囲を表します。ビンに使用される単位のタイプは、作成するヒストグラムのタイプによって異なります。ヒストグラム・テンプレートは、ヒストグラム中の最後から 2 番目のビンの最高値を表し、これはヒストグラム中のすべてのビンの値に影響します。ヒストグラムについて詳しくは、 292 ページの『ワークロード管理のヒストグラム』を参照してください。

手順

ヒストグラム・テンプレートを作成するには、次のようにします。

1. 作成するヒストグラム・テンプレートの名前と、最後から 2 番目のビンの最高値を設定する HIGH BIN VALUE キーワードの値を指定して、CREATE HISTOGRAM TEMPLATE ステートメントを発行します。
2. 変更をコミットします。 変更をコミットすると、ヒストグラムは SYSCAT.HISTOGRAMTEMPLATES ビューに追加され、ビンは SYSCAT.HISTOGRAMTEMPLATEBINS ビューに追加されます。

ヒストグラム・テンプレートの変更

既存のヒストグラム・テンプレートを変更するには、ALTER HISTOGRAM TEMPLATE ステートメントを使用します。ヒストグラム・テンプレートは、サービス・サブクラスおよびワーク・アクションにより、ヒストグラムで維持される統計のビン値を定義するために使用されます。

始める前に

ヒストグラム・テンプレートを変更するためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

このタスクについて

一部の DB2 サービス・サブクラス、ワーク・クラス・アクティビティー、および要求統計は、ヒストグラムを使用して収集されます。すべてのヒストグラムには特定数のビンがあり、それぞれのビンはアクティビティーまたは要求がカウントされる範囲を表します。ビンに使用される単位のタイプは、作成するヒストグラムのタイプによって異なります。ヒストグラム・テンプレートは、ヒストグラム中の最後から 2 番目のビンの最高値を表し、これはヒストグラム中のすべてのビンの値に影響します。ヒストグラムについて詳しくは、 292 ページの『ワークロード管理のヒストグラム』を参照してください。

手順

ヒストグラム・テンプレートを変更するには、次のようにします。

1. ALTER HISTOGRAM TEMPLATE ステートメントを発行し、変更するヒストグラム・テンプレートの名前と、最後から 2 番目のビンの最高値を変更するための HIGH BIN VALUE パラメーターの値を指定します。
2. 変更をコミットします。 変更をコミットすると、ヒストグラムのビンの最高値が SYSCAT.HISTOGRAMTEMPLATEBINS ビューで更新されます。この変更は、ワークロード管理統計が次回リセットされるまで有効になりません。詳しくは、 305 ページの『DB2 ワークロード管理オブジェクトの統計のリセット』を参照してください。
3. オプション: WLM_COLLECT_STATS ストアード・プロシージャーを実行して統計を収集およびリセットし、新しいヒストグラム・テンプレートが直ちに使用されるようにします。

ヒストグラム・テンプレートのドロップ

ヒストグラム・テンプレートは、必要ではなくなった場合にドロップできます。

始める前に

ヒストグラム・テンプレートをドロップするためには、WLMADM または DBADM 権限が必要です。

前提条件について詳しくは、 21 ページの『ワークロード管理 DDL ステートメント』を参照してください。

SYSDEFAULTHISTOGRAM ヒストグラム・テンプレートは、ドロップすることはできません。

サービス・サブクラス、ワーク・アクション、またはワークロードに参照されるヒストグラム・テンプレートはドロップできません。ヒストグラム・テンプレートを参照するサービス・サブクラスおよびワーク・アクションは、SYSCAT.HISTOGRAMTEMPLATESUSE ビューを照会して表示できます。

手順

ヒストグラム・テンプレートをドロップするには、次のようにします。

1. DROP HISTOGRAM TEMPLATE ステートメントを使用します。
2. 変更をコミットします。 変更をコミットすると、ヒストグラムは SYSCAT.HISTOGRAMTEMPLATES ビューから除去され、そのビンは SYSCAT.HISTOGRAMTEMPLATEBINS ビューから除去されます。

例: DB2 ワークロード管理構成におけるヒストグラムからの平均および標準偏差の計算

ヒストグラムの使用方法の 1 つは、アクティビティーの存続期間の標準偏差を入手することです。このトピックの例では、この統計の計算でビンを使用する方法を示します。

アクティビティーごとの平均存続期間の計算は、有用な情報の 1 つです。ただし、平均だけではユーザーの体感について正確に表すことはできません。アクティビティーの存続期間の変動性が大きいと、サポートを受けているユーザーが、照会の実行が高速なとき (望ましい状態) と、低速なとき (許容されないことのある状態) を体感されるかもしれません。アクティビティーの存続期間の目標を定義する場合、アクティビティーの平均存続期間だけでなく、アクティビティーの存続時間の標準偏差も重要になります。ユーザーが実際に体感しているのが実測上の平均であるようにするためには、変動性について理解し、それを制御する必要があります。

DB2 ワークロード管理構成において、統計はメンバーごとに収集されます。以下の例は、単一メンバーのアクティビティーの平均存続期間を取得する方法を示しています。

単一メンバー環境において、以下のビンが含まれるヒストグラムがあるとします。実際のヒストグラムではさらに多くのビンが存在しますが、ここでは例を簡潔にするためにビンの数は 8 に制限されています。

```
ビン 1 - 0 から 2 秒
ビン 2 - 2 から 4 秒
ビン 3 - 4 から 8 秒
ビン 4 - 8 から 16 秒
```

```
ビン 5 - 16 から 32 秒
ビン 6 - 32 から 64 秒
ビン 7 - 64 から 128 秒
ビン 8 - 128 秒から無限大
```

x から y の範囲のビンに該当する照会の平均応答時間を (x + y)/2 とすることで、平均の近似値を計算できます。次に、この数値とこのビンに入った照会数とを乗算し、すべてのビンに渡って合計してから、その和を全体のカウントで除算することができます。前述の例の場合に、各ビンの平均応答時間を以下のようであるとします。

```
ビン 1 の平均存続期間 = (0+2)/2 = 1
ビン 2 の平均存続期間 = (2+4)/2 = 3
ビン 3 の平均存続期間 = (4+8)/2 = 6
ビン 4 の平均存続期間 = (8+16)/2 = 12
ビン 5 の平均存続期間 = (16+32)/2 = 24
ビン 6 の平均存続期間 = (32+64)/2 = 48
ビン 7 の平均存続期間 = (64+128)/2 = 96
```

以下のヒストグラムが測定期間中に収集されたと仮定します。

ビン 1 カウント	ビン 2 カウント	ビン 3 カウント	ビン 4 カウント	ビン 5 カウント	ビン 6 カウント	ビン 7 カウント	ビン 8 カウント
20	30	80	10	5	3	2	0

平均存続期間を計算するには、ビン 8 は空でなければなりません。ビン 8 は、範囲の上限を変更する必要がある場合に通知するという目的でのみ存在します。このため、範囲の上限を指定する必要があります。

メンバー 1 の平均存続期間を以下のように概算することができます。

```
平均存続時間 = (20 x 1 + 30 x 3 + 80 x 6 + 10 x 12 + 5 x 24 + 3 x 48 + 2 x 96) / 150
             = (20 + 90 + 480 + 120 + 120 + 144 + 192) / 150
             = 1166 / 150
             = 7.77 秒
```

存続期間の標準偏差は、以下のように概算することができます。

標準偏差 = $[(20 \times (1 - 7.77)2 + 30 \times (3 - 7.77)2 + \ldots) / 150]^{1/2}$

マルチメンバー・データベース環境の場合、全データベース・メンバーの各ビンのカウントを加算し、全データベース・メンバーにわたる結合ヒストグラムを計算することで、平均および標準偏差を計算することができます。

例えば、データベースに 2 つのメンバーが含まれており、ヒストグラムのビンのサイズは前の部分で説明されているとおりで、ヒストグラムには以下のデータが含まれるとします。

データベース メンバー	ビン 1 カウント	ビン 2 カウント	ビン 3 カウント	ビン 4 カウント	ビン 5 カウント	ビン 6 カウント	ビン 7 カウント	ビン 8 カウント
1	20	30	80	10	5	3	2	0
2	1	5	20	20	4	0	0	0

ビンのサイズはすべてのデータベース・メンバーで同じであるため、ヒストグラム全体は以下のように簡単に計算することができます。

ビン 1 カウント	ビン 2 カウント	ビン 3 カウント	ビン 4 カウント	ビン 5 カウント	ビン 6 カウント	ビン 7 カウント	ビン 8 カウント
21	35	100	30	9	3	2	0

結合ヒストグラムから、全体の存続期間の平均および標準偏差を、単一メンバー環境で計算したときと同じ方法で以下のように計算できます。

```
平均存続時間 = (21 x 1 + 35 x 3 + 100 x 6 + 30 x 12 + 9 x 24 + 3 x 48 + 2 x 96) / 200
              = (21 + 105 + 600 + 360 + 216 + 144 + 192) / 200
              = 1638 / 200
              = 8.19 秒

Standard deviation = [(21 x (1 - 8.19)^2 + 35 x (3 - 7.77)^2 + ... ) / 200]^1/2
```

履歴分析ツール

DB2 データ・サーバーのインストール済み環境には、サンプルとして一対の Perl スクリプトが含まれています。このスクリプトは、履歴分析の実行によってどの表、索引、および列がアクセスされたかまたはアクセスされなかったかについての情報を生成します。

これらのスクリプトは、ワークロード管理のアクティビティー・イベント・モニターによってキャプチャーされた情報を使用することによって、履歴分析機能を提供します。ワークロード管理の履歴分析ツールは Perl で書かれています。これらのスクリプトはそのまま使用できるほか、必要性を満たす追加の履歴分析レポートを作成するように変更することもできます。

ワークロード管理の履歴分析ツールは、次の 2 つのスクリプトから成ります。これらのスクリプトは、インストール・ディレクトリーの `samples/perl` パスにあります。

- wlmhist.pl - 履歴データを生成します。
- wlmhistrep.pl - 履歴データからレポートを作成します。

この 2 つのスクリプトが使用する共通 Perl ルーチンの入った DB2WlmHist.pm ファイルも含まれています。

これらのスクリプトをセットアップして実行する方法について詳しくは、同じファイル・ディレクトリーにある README_WLMHIST ファイルを参照してください。

統計イベント・モニターを使用したワークロード管理統計の収集

DB2 ワークロード管理オブジェクトの統計は、統計イベント・モニターに送信して、履歴分析できます。

このタスクについて

統計を使用して、長期にわたるシステムの動作を理解し (例えばアクティビティーの平均存続時間、アクティビティーがキューで費やされた時間、小さなアクティビティーと比較した大きなアクティビティーの分散など)、しきい値を設定し (例えば並行アクティビティーの上限を見つけるなど)、問題を検出する (例えば現在ユーザーが経験している平均存続時間が通常より長いのかどうかを検出するなど) ことができます。各 DB2 ワークロード管理オブジェクトについてどの統計が収集されるのかについての説明は、280 ページの『DB2 ワークロード管理オブジェクトの統計』を参照してください。

ワークロード管理統計は、決められた時間間隔でイベント・モニターへ自動送信することも、任意の時点で手動で送信することもできます。

手順

ワークロード管理統計を決められた時間間隔で自動収集するには、次のようにします。

1. CREATE EVENT MONITOR ステートメントを使用して、STATISTICS イベント・モニターを作成します。 例えば、次のステートメントを発行できます。

   ```
   CREATE EVENT MONITOR STATS1 FOR STATISTICS WRITE TO TABLE
   ```

2. COMMIT ステートメントを使用して、変更をコミットします。
3. SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを活動化します。 SET EVENT MONITOR STATE ステートメントを使用せずに、STATISTICS イベント・モニターに AUTOSTART のデフォルトを使用して、データベースが次回活動化されたときにこのイベント・モニターが活動化されるようにする方法もあります。複数の STATISTICS イベント・モニターを定義する場合は、AUTOSTART オプションを使用しないでください。
4. COMMIT ステートメントを使用して、変更をコミットします。
5. オプション: 追加的な統計の収集を使用可能にします。 デフォルトでは、各 DB2 ワークロード管理オブジェクトについて、最小セットの統計のみが収集されます。各オブジェクトに対して、デフォルトでどの統計が収集されるかについては、280 ページの『DB2 ワークロード管理オブジェクトの統計』を参照してください。 ALTER SERVICE CLASS ステートメントおよび ALTER WORK ACTION SET ステートメントの COLLECT AGGREGATE ACTIVITY DATA キーワードを使用して、サービス・サブクラス、ワークロード、およびワーク・クラスの集約アクティビティー・データの収集を指定します。 ALTER SERVICE CLASS ステートメントの COLLECT AGGREGATE REQUEST DATA キーワードを使用して、サービス・サブクラスの集約要求データの収集を指定します。変更があればコミットします。
6. データベース構成パラメーター **wlm_collect_int** を更新して、収集間隔を指定します。 **wlm_collect_int** パラメーターは、時間間隔を分単位で指定します。間隔ごとに、すべての DB2 ワークロード管理オブジェクトのワークロード管理統計のコピーがアクティブな統計イベント・モニターに書き込まれ、統計がリセットされます。マルチメンバー・データベース環境では、**wlm_collect_int** パラメーターが、カタログ・メンバーで更新される必要があります。このパラメーターは、動的に更新できます。以下に例を示します。

   ```
   CONNECT TO database alias
   UPDATE DATABASE CONFIGURATION USING WLM_COLLECT_INT 5 IMMEDIATE
   ```

タスクの結果

上記の各ステップを実行すると、ワークロード管理統計は **wlm_collect_int** 分ごとに統計イベント・モニターに書き込まれます。この統計イベント・モニターに書き込まれた各レコードは、STATISTICS_TIMESTAMP 値と LAST_WLM_RESET 値を持っています。LAST_WLM_RESET から STATISTICS_TIMESTAMP までの時間間隔により、収集の間隔 (つまり、そのレコードの統計が収集された時間間隔) が定義されます。

収集は、日曜日の 00:00:00 を基準として測定される、指定の間隔で行われます。カタログ・メンバーがアクティブになると、この定刻を基準として次にスケジュールされた間隔が開始するときに、次の収集が行われます。スケジュールされた間隔

は、カタログ・メンバーがアクティブになった時間を基準とはしません。収集の時刻にメンバーがアクティブになっていない場合、そのメンバーの統計は収集されません。例えば、間隔の値が 60 に設定され、カタログ・メンバーが日曜日の午前 9:24 にアクティブ化された場合、収集が毎時正時に行われるようにスケジュールされます。つまり、次の収集は午前 10:00 に行われます。メンバーが午前 10:00 の時点でアクティブになっていない場合、そのメンバーの統計は収集されません。

wlm_collect_int パラメーターがゼロ以外の値に設定され、アクティブな統計イベント・モニターがない場合、ワークロード管理統計は引き続き **wlm_collect_int** 分ごとにリセットされますが、統計は収集されません。するとデータは失われます。このため、統計イベント・モニターを活動化せずに **wlm_collect_int** 値にゼロ以外の値を指定することはお勧めできません。

wlm_collect_int パラメーターが 0 に設定されている場合 (デフォルト)、統計が自動的に統計イベント・モニターに送信されることはありません。WLM_COLLECT_STATS ストアード・プロシージャーを使用すると、後で履歴分析をするために、統計を統計イベント・モニターに手動で送信できます。このプロシージャーが呼び出されると、自動統計収集間隔で行われる場合と同じアクションが実行されます。つまり、統計が統計イベント・モニターに送信されて、統計がリセットされます。アクティブな統計イベント・モニターがない場合、値はリセットされますが、データは収集されません。統計をリセットしようとするだけの場合は、アクティブな統計イベント・モニターがないときに WLM_COLLECT_STATS プロシージャーを呼び出すことができます。

統計の手動収集は、統計の自動収集の支障にはなりません。例えば、**wlm_collect_int** が 60 に設定されているとします。統計は、統計イベント・モニターに 1 時間ごとに送信されます。ここで、最後に統計が収集された時刻が午前 5:00 であると仮定します。 WLM_COLLECT_STATS プロシージャーを午前 5:55 に呼び出すと、統計の値がイベント・モニターに送信され、統計がリセットされます。次の自動統計収集は、前回の自動収集の 1 時間後である午前 6:00 に行われます。収集の間隔は手動による収集およびその間隔の間に発生した統計のリセットの影響を受けません。

注:

- DB2 ワークロード管理統計の表関数は、統計の現行値を報告します。自動ワークロード管理統計収集を使用可能にしている場合、これらの値は **wlm_collect_int** データベース構成パラメーターで定義された間隔で定期的にリセットされます。表関数で報告される統計を調べるときは、必ず LAST_RESET 列を検討するようにします。この列は、前回統計がリセットされた時刻を示しています。前回のリセット時刻から現在時刻までの時間間隔が不十分な場合、意味のある結論を引き出すのに十分なデータがない場合があります。
- ワークロード管理統計の自動収集を使用している場合は、イベント・モニターのファイルや表を定期的に整理する必要があります。イベント・モニターが収集されたデータを自動的に整理することはなく、自動収集によってファイルと表は時間とともにいっぱいになってしまいます。
- データベースが非活動化されると、統計はリセットされます。データベースを非活動化すると、統計は統計イベント・モニターには送信されません。前回の収集以降累積された統計を非活動化のために失いたくない場合

は、データベースを非活動化する前に WLM_COLLECT_STATS プロシージャーを手動で呼び出す必要があります。

- WLM_COLLECT_STATS プロシージャーは、**RESET MONITOR** コマンドとは別の方法で統計をリセットします。 **RESET MONITOR** コマンドは、現在の値を保管することによってスナップショット・モニター・エレメントの値をリセットします。 **RESET MONITOR** コマンドが発行された後で、スナップショット処理はこれらの値と現行値との間の差分を報告します。これに対して、WLM_COLLECT_STATS プロシージャーによるリセットでは値は保管されず、該当する各 DB2 ワークロード管理オブジェクトの統計カウンター自体がすべてリセットされます。

  また、**RESET MONITOR** コマンドでは、各プロセス (アタッチメント) には、モニター・データについての独自のプライベート・ビューがあります。あるユーザーがリセットを実行しても、他のユーザーは影響を受けません。これに対して、ワークロード管理統計のリセットは、すべてのユーザーに適用されます。

DB2 ワークロード管理オブジェクトの統計のリセット

このトピックは、DB2 ワークロード管理オブジェクトの統計をリセットする方法を説明します。

統計のリセットが適用されるのは、DB2 ワークロード管理統計だけであることに注意してください。モニター・インターフェースによって報告されるメトリックは収集されますが、リセットされません。

4 つのイベントによって、それぞれの DB2 ワークロード管理オブジェクトのために保管されている統計がリセットされます。(それぞれのオブジェクトのために維持されている統計についての説明は、 280 ページの『DB2 ワークロード管理オブジェクトの統計』を参照してください。)

- WLM_COLLECT_STATS ストアード・プロシージャーが呼び出される。詳しくは、 302 ページの『統計イベント・モニターを使用したワークロード管理統計の収集』を参照してください。
- **wlm_collect_int** データベース構成パラメーターにより制御される、自動の DB2 ワークロード管理統計収集およびリセットのプロセスによって、収集およびリセットが起こる。詳しくは、 302 ページの『統計イベント・モニターを使用したワークロード管理統計の収集』を参照してください。
- データベースが再活動化される。メンバー上でデータベースが活動化されるたびに、そのメンバー上のすべての DB2 ワークロード管理オブジェクトの統計はリセットされます。
- 統計が維持されているオブジェクトが変更され、その変更内容がコミットされる。例えば、あるサービス・サブクラスが変更される場合、ALTER ステートメントがコミットされるとそのサービス・サブクラスの統計はリセットされます。

統計の表関数を使用したり、LAST_RESET 列にあるタイム・スタンプを見たりして、特定の DB2 ワークロード管理オブジェクトの統計が最後にリセットされた時間を判別することができます。例えば、SYSDEFAULTUSERCLASS サービス・スー

パークラス下のサービス・サブクラス SYSDEFAULTSUBCLASS に対して最後に統計がリセットされた時間を知るために、以下のような照会を発行できるかもしれません。

```
SELECT LAST_RESET
   FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS( 'SYSDEFAULTUSERCLASS',
   'SYSDEFAULTSUBCLASS', -2)) AS T
```

すべての統計表関数は、統計が最後にリセットされた時から累算した統計を返します。統計のリセットは、データベースがアクティブにされた時または非アクティブにされた時、DB2 ワークロード管理オブジェクトを変更した時 (そのオブジェクトの統計だけがリセットされます)、そして WLM_COLLECT_STATS ストアード・プロシージャーを呼び出した時に発生します。**wlm_collect_int** データベース構成パラメーターをゼロ以外の値に設定した場合にも、統計はこのパラメーターによって定義される期間に従って自動的にリセットされます。

wlm_collect_int によって指定される期間は、この構成パラメーターによって指定されるインターバルの間に発生する統計のリセットによる影響を受けません。例えば、**wlm_collect_int** によって指定される 20 分間のインターバルが開始してから 5 分後に WLM_COLLECT_STATS 表関数を実行する場合、そのインターバルはやはり 15 分後に有効期限が切れます。統計の収集およびリセットが発生することによって、次の統計の収集とリセットの発生が 5 分遅くなることはありません。

別のヒストグラム・テンプレートを使用するようサービス・クラスまたはワークロードを変更する場合、あるいはヒストグラム・テンプレートを変更する場合、その変更は統計をリセットするまで有効になりません。

WLM_COLLECT_STATS 表関数を呼び出して統計の収集とリセットを行うとき、これと同時に別の収集とリセットが進行中であると (例えば、**wlm_collect_int** による定期的な収集およびリセットと表関数の呼び出しとがオーバーラップする場合、あるいは別のユーザーが同時に WLM_COLLECT_STATS を呼び出す場合)、WLM_COLLECT_STATS からの収集およびリセット要求は無視され、警告 SQL1632W が返されます。

DB2 ワークロード管理のためのモニター・メトリック

モニター・メトリックは、DB2 データ・サーバーの正常性および照会パフォーマンスに関するデータを提供します。そのデータをサード・パーティー・ツールに対する入力として使用するか、自分で作成した追加のスクリプトと組み合わせて使用することにより、返されたメトリックを分析できます。

メトリックは、多くの DB2 データベース・オブジェクトについて維持されています。これらのメトリックはメモリー内に常駐しており、DB2 モニター・メトリック表関数を使用してリアルタイムで表示することが可能です。あるいは、メトリックを収集してイベント・モニターに送信し、後で履歴分析を行うときに表示することもできます。

アクティビティーのためのモニター・メトリック

以下を使用すると、アクティビティーのモニター・メトリックを取得できます。

- アクティビティー・イベント・モニター (ACTIVITYMETRICS 表、または ACTIVITY 表の DETAILS_XML 列)
- MON_GET_ACTIVITY_DETAILS 表関数

アクティビティーのモニター・メトリックは、**mon_act_metrics** データベース構成パラメーターと、ワークロードに対する COLLECT ACTIVITY METRICS 節によって制御されます。このデータベース構成パラメーターが NONE 以外の値に設定されている場合、または NONE 以外の COLLECT ACTIVITY METRICS 設定値を持つワークロードに関連付けられた接続によってアクティビティーがサブミットされる場合、アクティビティーのメトリックが収集されます。

すべてのアクティビティーのメトリックを収集しない場合には、ワークロード・レベルの制御を使用するとモニターの細分性を高めることができます。データベース・レベルでアクティビティー・メトリックの収集が有効な場合 (デフォルトでは有効)、ワークロード・レベルでの設定には関係なく、すべてのアクティビティーのメトリックが収集されます。

詳しくはモニター資料を参照してください。

システム・レベル・モニター・メトリック

サービス・クラスおよびワークロードによって集約されたシステム・レベルのモニター・メトリックを、以下を使用して取得できます。

- 統計イベント・モニター (EVENT_SCSTATS および EVENT_WLSTATS 論理データ・グループのモニター・エレメント **details_xml** および **metrics**、または EVENT_SCMETRICS および EVENT_WLMETRICS 論理データ・グループの個々のモニター・エレメント)
- MON_GET_SERVICE_SUBCLASS、MON_GET_SERVICE_SUBCLASS_DETAILS、MON_GET_WORKLOAD および MON_GET_WORKLOAD_DETAILS 表関数

データ・サーバーに対する要求 (アクティビティーの一部となる要求を含む) のモニター・メトリックは、**mon_req_metrics** データベース構成パラメーター、およびサービス・スーパークラスの COLLECT REQUEST METRICS 節によって制御されます。このデータベース構成パラメーターが NONE 以外の値に設定されている場合、または NONE 以外の COLLECT COLLECT REQUEST METRICS 設定値を持つスーパークラスの下のサブクラスにマップされる接続に要求がサブミットされる場合に、要求のメトリックが収集されます。

すべての要求のメトリックを収集しない場合には、サービス・スーパークラス・レベルの制御を使用するとモニターの細分性を高めることができます。データベース・レベルで要求メトリックの収集が有効な場合 (デフォルトでは有効)、サービス・スーパークラス・レベルでの設定には関係なく、すべての要求のメトリックが収集されます。

詳しくはモニター資料を参照してください。

ワークロード管理表関数とスナップショット・モニターの統合

問題判別またはパフォーマンス調整を実行する際、DB2 ワークロード管理表関数をスナップショット・モニター表関数とともに使用できます。

DB2 ワークロード管理表関数とスナップショット・モニター表関数は、以下のフィールドを共有します。こうしたフィールドでデータを結合して、診断およびパフォーマンス調整を実施するのに必要なデータを得ることができます。スナップショット表関数と異なり、WLM 表関数は自分の情報をスナップショット・モニターからは取得しないので、WLM 表関数で使用できる情報をスナップショット・モニターから入手できるわけではないことに注意してください。

表 61. DB2 ワークロード管理表関数とスナップショット・モニター表関数で共有するフィールド

ワークロード管理表関数フィールド	スナップショット・モニター表関数フィールド
agent_tid	**agent_pid**
application_handle	**agent_id** **agent_id_holding_lock**
session_auth_id	**session_auth_id**
member	**node_number**
utility_id	**utility_id**
workload_id	**workload_id**

異なる表関数の間で結合を使用する理由を示す例として、BATCH サービス・スーパークラスで実行されているすべてのユーティリティーに関する基本情報を取得したいとします。以下の照会を発行できます。

```
SELECT SUBSTR(UTILITY_TYPE,1,4) TYPE,
       UTILITY_PRIORITY PRIORITY,
       SUBSTR(UTILITY_DESCRIPTION,1,12) DESCRIPTION,
       SUBSTR(UTILITY_DBNAME,1,8) DBNAME,
       UTILITY_STATE STATE,
       SUBSTR(UTILITY_INVOKER_TYPE,1,7) INVOKER,
       SUBSTR(CHAR(WLM.MEMBER),1,4) MEMB,
       SUBSTR(CLASSES.PARENTSERVICECLASSNAME,1,19) SUPERCLASS_NAME,
       SUBSTR(CLASSES.SERVICECLASSNAME,1,18) SUBCLASS_NAME
FROM SYSIBMADM.SNAPUTIL SNAP,
     TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(CAST(NULL AS BIGINT), -2)) WLM,
     SYSCAT.SERVICECLASSES CLASSES
WHERE SNAP.UTILITY_ID = WLM.UTILITY_ID
  AND WLM.SERVICE_CLASS_ID = CLASSES.SERVICECLASSID
  AND CLASSES.SERVICECLASSNAME = 'SYSDEFAULTSUBCLASS'
  AND CLASSES.PARENTSERVICECLASSNAME = 'BATCH'
ORDER BY WLM.MEMBER
```

出力は、以下のようになります。

```
TYPE PRIORITY DESCRIPTION  DBNAME STATE   INVOKER MEMB SUPERCLASS_NAME SUBCLASS_NAME
---- -------- ------------ ------ ------- ------- ---- --------------- ------------------
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    1    BATCH           SYSDEFAULTSUBCLASS
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    1    BATCH           SYSDEFAULTSUBCLASS
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    1    BATCH           SYSDEFAULTSUBCLASS
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    2    BATCH           SYSDEFAULTSUBCLASS
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    2    BATCH           SYSDEFAULTSUBCLASS
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    2    BATCH           SYSDEFAULTSUBCLASS
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    3    BATCH           SYSDEFAULTSUBCLASS
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    3    BATCH           SYSDEFAULTSUBCLASS
LOAD        - OFFLINE LOAD SAMPLE EXECUTE USER    3    BATCH           SYSDEFAULTSUBCLASS
```

しきい値違反のモニター

DB2 ワークロード・マネージャーのしきい値の違反があった場合には、アクティブな THRESHOLD VIOLATIONS イベント・モニターがある場合、それにしきい値違反レコードが書き込まれます。

このタスクについて

しきい値違反レコードには、以下の情報が含まれます。

- 違反のあったしきい値の説明 (ID、最大値、など)。
- しきい値に違反したアクティビティーの ID。これには、そのアクティビティーをサブミットしたアプリケーションの ID、アクティビティーの固有 ID、および作業単位 ID が含まれます。
- しきい値の違反が生じた時刻。
- 取られた処置。この処置は、しきい値に違反したアクティビティーが続行を許可されたか、または停止されたかを示します。アクティビティーが停止された場合は、そのアクティビティーをサブミットしたアプリケーションは SQL4712N エラーを受け取ることになります。

REMAP ACTIVITYアクション が定義されているしきい値で違反が生じた場合、しきい値違反レコードはオプションです。しきい値違反レコードが記録されるかどうかは、CREATE THRESHOLD ステートメントの NO EVENT MONITOR RECORD または LOG EVENT MONITOR RECORD 節によって決まります。

オプションで、アクティビティーによってしきい値違反が起こった場合に、詳細なアクティビティー情報 (ステートメント・テキストを含む) がアクティブなアクティビティー・イベント・モニターに書き込まれるようにすることができます。アクティビティー情報は、しきい値の違反が生じたときではなく、アクティビティーの完了時に書き込まれます。しきい値の違反時にアクティビティー情報を収集するように指定するには、COLLECT ACTIVITY DATA キーワードを CREATE または ALTER しきい値またはワーク・アクション・セットのステートメントで使用します。

手順

しきい値の違反をモニターするには、以下のようにします。

1. CREATE EVENT MONITOR ステートメントを使用して、タイプが THRESHOLD VIOLATIONS であるイベント・モニターを作成します。 以下に例を示します。

   ```
   CREATE EVENT MONITOR VIOLATIONS FOR THRESHOLD VIOLATIONS WRITE TO TABLE
   ```

2. COMMIT ステートメントを使用して、変更をコミットします。
3. SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを活動化します。 SET EVENT MONITOR STATE ステートメントを使用せずに、THRESHOLD VIOLATIONS イベント・モニターに AUTOSTART のデフォルトを使用して、データベースが次回活動化されたときにこのイベント・モニターが活動化されるようにする方法もあります。複数の THRESHOLD VIOLATIONS イベント・モニターを定義する場合は、AUTOSTART オプションを使用しないでください。

4. COMMIT ステートメントを使用して、変更をコミットします。

注: しきい値を作成する場合には、しきい値違反が生じたときにモニターできるよう、しきい値違反イベント・モニターを作成してアクティブにすることをお勧めします。しきい値違反イベント・モニターが与える影響は、しきい値違反が生じない限り、全くありません。

例

この例は、 REMAP ACTIVITY アクションが含まれていたしきい値違反の結果としてどのアクティビティーでどの再マップが発生したかを判別する方法を示しています。再マップされたアクティビティーを見つけるには、次のようなステートメントを使用します。

```
SELECT VARCHAR(APPL_ID, 30) AS APPLID,
   UOW_ID,
   ACTIVITY_ID,
   VARCHAR(T.PARENTSERVICECLASSNAME,20) AS SERVICE_SUPERCLASS,
   VARCHAR(T.SERVICECLASSNAME,20) AS FROM_SERVICE_SUBCLASS,
   VARCHAR(S.SERVICECLASSNAME,20) AS TO_SERVICE_SUBCLASS
FROM THRESHOLDVIOLATIONS_TH1,
   SYSCAT.SERVICECLASSES AS T,
   SYSCAT.SERVICECLASSES AS S
WHERE SOURCE_SERVICE_CLASS_ID = T.SERVICECLASSID AND
      DESTINATION_SERVICE_CLASS_ID = S.SERVICECLASSID AND
      THRESHOLD_ACTION = 'REMAP'
ORDER BY APPLID, ACTIVITY_ID, UOW_ID, TIME_OF_VIOLATION ASC;
```

この例では、アクティビティー ID 1 および作業単位 (UOW) ID 1 で識別される ID *N0.swalkty.080613140844 のアプリケーションによってサブミットされたアクティビティーで 2 度の再マップが発生しました。

```
APPLID                         UOW_ID      ACTIVITY_ID          SERVICE_SUPERCLASS   FROM_SERVICE_SUBCLASS TO_SERVICE_SUBCLASS
------------------------------ ----------- -------------------- -------------------- --------------------- --------------------
*N0.swalkty.080613140844                 1                    1 WORK                 HIGH                  MED
*N0.swalkty.080613140844                 1                    1 WORK                 MED                   LOW

  2 record(s) selected.
```

出力は、しきい値違反の時刻によって配列されており、アクティビティーが実行を開始した後、2 度の再マップが発生したことを示しています。出力には示されていませんが、アクティビティーがマップされた初期サービス・サブクラスは高優先順位のサービス・サブクラスであると思われます。実行時間の短い照会を短時間で完了できる 3 層構成では通常そのようになります。アクティビティーは高優先順位のサービス・サブクラスとしての妥当な時間に完了しなかったため、しきい値違反となり、中優先順位のサービス・サブクラスに再マップされました。その後 2 度目のしきい値違反となり、再び再マップが発生し、低優先順位のサービス・サブクラスに再マップされました。

しきい値違反に関する E メール通知の生成方法

ここに記載するメソッドを使用すると、 DB2 ワークロード・マネージャー (WLM) のしきい値が違反された時に、E メール通知を生成できます。

始める前に

この E メール通知方法を実装するには、DB2 バージョン 9.7 以上がインストールされていなければなりません。ここで使用する SMTP サポートは、DB2 V9.7 以降で提供されています。

このタスクについて

このタスクを完了させると、しきい値通知プロシージャーが最後に実行されてから 10 分の間に、 WLM しきい値違反が発生した場合に、E メール通知が送信されます。この例では、DB2 管理用タスク・スケジューラーを使用して、しきい値通知プロシージャーを 10 分間隔で実行するようにスケジュールしています。

手順

1. 次のコマンドを実行して、**smtp_server** データベース構成パラメーターを更新します。

   ```
   UPDATE DB CONFIG USING SMTP_SERVER smtp_server_name
   ```

2. 次のステートメントを実行して、しきい値違反の表書き込みイベント・モニターを作成し、 TEST.THRESHOLDVIOLATIONS_T 表に違反を書き込むようにします。

   ```
   CREATE EVENT MONITOR T FOR THRESHOLD VIOLATIONS WRITE TO TABLE
   THRESHOLDVIOLATIONS( TABLE TEST.THRESHOLDVIOLATIONS_T )
   ```

3. 次のステートメントを実行して、しきい値違反の表書き込みイベント・モニター T をアクティブ化します。

   ```
   SET EVENT MONITOR T STATE 1
   ```

4. 次のステートメントを実行して、最後にアラートが生成されたしきい値をトラックするためのコントロール表を作成します。

   ```
   CREATE TABLE TEST.THRESHOLD_NOTIFY_CONTROL( LAST_NOTIFICATION TIMESTAMP )
   ```

5. しきい値違反メッセージを生成する、しきい値通知ストアード・プロシージャーを作成します。次の例のプロシージャーは、しきい値違反表に対して反復処理を実行して、前回のプロシージャー実行以降に発生したすべてのしきい値違反をリストしたレポートを作成します。このレポートは、DB2 SMTP プロシージャーを使用して E メール通知されます。

   ```
   CREATE PROCEDURE TEST.NOTIFY_ON_THRESHOLD_VIOLATION()
     LANGUAGE SQL
     BEGIN
        DECLARE NEWEST_VIOLATION    TIMESTAMP;
        DECLARE LAST_VIOLATION_SEEN TIMESTAMP;
        DECLARE NOT_FOUND           INTEGER DEFAULT 0;
        DECLARE SENDER              VARCHAR(128);
        DECLARE RECIPIENTS          VARCHAR(128);
        DECLARE MESSAGE             VARCHAR(8192);
        DECLARE SUBJECT             VARCHAR(128);
        DECLARE THRESHOLDID         BIGINT;
        DECLARE APPL_ID             VARCHAR(64);
        DECLARE THRESHOLD_PREDICATE VARCHAR(64);
        DECLARE TIME_OF_VIOLATION   TIMESTAMP;

        DECLARE C1 CURSOR FOR SELECT MAX(TIME_OF_VIOLATION)
                              FROM TEST.THRESHOLDVIOLATIONS_T;
        DECLARE C2 CURSOR FOR SELECT LAST_NOTIFICATION
                              FROM TEST.THRESHOLD_NOTIFY_CONTROL;
        DECLARE C3 CURSOR FOR SELECT THRESHOLD_PREDICATE, THRESHOLDID,
                                     TIME_OF_VIOLATION, APPL_ID
                              FROM TEST.THRESHOLDVIOLATIONS_T
   ```

```
                        WHERE LAST_VIOLATION_SEEN IS NULL OR TIME_OF_VIOLATION
                                                          > LAST_VIOLATION_SEEN;

      DECLARE CONTINUE HANDLER FOR NOT FOUND
         SET NOT_FOUND = 1;

      OPEN C1;
      FETCH C1 INTO NEWEST_VIOLATION;
      CLOSE C1;

      IF ( NOT_FOUND = 0 ) THEN

         OPEN C2;
         FETCH C2 INTO LAST_VIOLATION_SEEN;
         CLOSE C2;

         IF ( NOT_FOUND = 1 ) THEN
            SET LAST_VIOLATION_SEEN = NULL;
         END IF;

         IF ( NOT_FOUND = 1 OR NEWEST_VIOLATION > LAST_VIOLATION_SEEN ) THEN

            DELETE FROM TEST.THRESHOLD_NOTIFY_CONTROL;
            INSERT INTO TEST.THRESHOLD_NOTIFY_CONTROL VALUES
                                      ( NEWEST_VIOLATION );

            SET SENDER = '<sender email address>';
            SET RECIPIENTS = '<receiver email address>';
            SET SUBJECT = 'New WLM Threshold Violations' ;

            SET NOT_FOUND = 0;
            SET MESSAGE = '';

            OPEN C3;

            FETCH C3 INTO THRESHOLD_PREDICATE, THRESHOLDID, TIME_OF_VIOLATION,
                                                                    APPL_ID;

            WHILE ( NOT_FOUND = 0 ) DO

               SET MESSAGE = MESSAGE || 'Violation Timestamp = ' ||
                                TIME_OF_VIOLATION || CHAR(X'0A');
               SET MESSAGE = MESSAGE || 'Threshold Predicate = ' ||
                              THRESHOLD_PREDICATE || CHAR(X'0A');
               SET MESSAGE = MESSAGE || 'Threshold Id        = ' ||
                                 THRESHOLDID || CHAR(X'0A');
               SET MESSAGE = MESSAGE || 'Appl Id             = ' ||
                              APPL_ID || CHAR(X'0A') || CHAR(X'0A');

               FETCH C3 INTO THRESHOLD_PREDICATE, THRESHOLDID,
                                 TIME_OF_VIOLATION, APPL_ID;
            END WHILE;

            CLOSE C3;

            CALL UTL_MAIL.SEND( SENDER, RECIPIENTS, NULL, NULL, SUBJECT,
                                                          MESSAGE );

            COMMIT;

         END IF;

      END IF;

   END@
```

6. 次のコマンドを実行して、DB2 管理用タスク・スケジューラーを有効にします。

```
db2set DB2_ATS_ENABLE=YES
```

7. しきい値通知プロシージャーを、10 分間隔で実行するようにスケジュールします。プロシージャーをスケジュールするには、プロシージャーに対する実行特権が必要です。実行例を次に示します。

```
CALL SYSPROC.ADMIN_TASK_ADD(
   'CHECK THRESHOLD VIOLATIONS EVERY 10 MINUTES',
   NULL,
   NULL,
   NULL,
   '0-59/10 * * * *',
   'TEST',
   'NOTIFY_ON_THRESHOLD_VIOLATION',
   NULL,
   NULL,
   NULL )@
```

タスクの結果

WLM しきい値違反が発生すると、必ず E メールが送信されます (待ち時間は最大 10 分間です)。E メールには、 10 分間隔で実行するようにスケジュールされた、しきい値違反プロシージャーの前回の実行以降に発生したすべての WLM しきい値違反が記載されています。

例

次の出力は、E メール通知メッセージの内容の例です。しきい値違反プロシージャーの前回の実行以降、累積された新規しきい値違反が表示されています。

```
Subject: New WLM Threshold Violations

Violation Timestamp = 2010-01-11-10.57.21.000000
Threshold Predicate = CPUTime
Threshold Id        = 1
Appl Id             = *LOCAL.horton.100111154912

Violation Timestamp = 2010-01-11-10.57.28.000000
Threshold Predicate = CPUTime
Threshold Id        = 1
Appl Id             = *LOCAL.horton.100111154912

Violation Timestamp = 2010-01-11-10.57.35.000000
Threshold Predicate = CPUTime
Threshold Id        = 1
Appl Id             = *LOCAL.horton.100111154912
```

個々のアクティビティーのデータ収集

ACTIVITIES イベント・モニターを使用して、システム上で実行される個々のアクティビティーのデータを収集することができます。収集されるデータには、ステートメント・テキストおよびコンパイル環境といった項目が含まれており、問題の調査と診断に使用したり、他のツール (例えば、設計アドバイザー) への入力データとして使用したりすることができます。

このタスクについて

サービス・サブクラス、ワークロード、ワーク・クラス (ワーク・アクションを通して)、およびしきい値違反に対して、個々のアクティビティーに関する情報を収集できます。これらの DB2 ワークロード管理オブジェクトに対するアクティビティーの収集は、CREATE または ALTER ステートメントの COLLECT ACTIVITY DATA キーワードを使用して有効にします。以下の場合に、アクティビティーの完了時に、そのアクティビティーに関する情報は、アクティブな ACTIVITIES イベント・モニターに送信されます。

- アクティビティーは、COLLECT ACTIVITY DATA が指定されているワークロードにマップされたアプリケーションによってサブミットされた。
- または、アクティビティーは、COLLECT ACTIVITY DATA が指定されているサービス・サブクラス内で実行されている。
- または、アクティビティーには COLLECT ACTIVITY DATA ワーク・アクションが適用されている。
- または、アクティビティーは、COLLECT ACTIVITY DATA アクションで定義されたしきい値に違反する。

また、WLM_SET_CONN_ENV プロシージャーを使用すると、ユーザーの照会を実行する前に独自のアプリケーション接続用のアクティビティー収集をオンにして、それからユーザーの照会を実行し、その後 WLM_SET_CONN_ENV を使用して独自のアプリケーション接続用のアクティビティー収集をオフにできます。アクティビティー・イベント・モニターを作成して活動化したとすると、アプリケーションでは以下に類似したものになります。

```
call WLM_SET_CONN_ENV(cast (NULL as bigint),
  '<collectactdata>WITHOUT DETAILS</collectactdata>')
```

... ユーザー照会の実行 ...

```
call WLM_SET_CONN_ENV(cast(NULL as bigint), '<collectactdata>NONE</collectactdata>')
```

COLLECT ACTIVITY DATA キーワードは、ACTIVITIES イベント・モニターへ送信される情報量もコントロールします。このキーワードで WITH DETAILS が指定されている場合、ステートメント情報 (ステートメント・テキストなど) が収集されます。キーワードで WITH DETAILS AND VALUES が指定されている場合、データ値も収集されます。

アクティビティーには、複数の COLLECT ACTIVITY DATA キーワードが適用される場合があります。例えば、アクティビティーは、COLLECT ACTIVITY DATA が指定されているサービス・サブクラス内で実行され、実行中に COLLECT ACTIVITY DATA アクションを持つしきい値に違反するかもしれません。この場合は、アクティビティーは一度のみ収集されます。アクティビティーには、最も大量の情報の収集を指定する COLLECT キーワードが適用されます。例えば、COLLECT ACTIVITY DATA WITHOUT DETAILS と COLLECT ACTIVITY DATA WITH DETAILS の両方が 1 つのアクティビティーに適用された場合、アクティビティーは詳細情報と共に収集されます。

COLLECT ACTIVITY DATA 節と共に ON ALL DATABASE MEMBERS キーワードを使用すると、マルチメンバー・データベース環境でアクティビティーが実行される各メンバーにおいてアクティビティー・レコードが収集されます。そのメンバ

ーでアクティビティーを操作する最後のエージェントの実行が完了したときに、アクティビティー・イベント・モニター・レコードが書き込まれます。セクション内のイベントの順序付けによっては、あるメンバーでエージェントがアクティビティーの処理を何度も開始/停止して、そのために同じ照会で複数のアクティビティー・レコードがそのメンバーで収集される可能性があります。そのメンバーでアクティビティーが実行した作業の合計は、メンバーでアクティビティーに関して収集された各レコードのメトリックの集計です。

手順

特定の DB2 ワークロード管理オブジェクトに対するアクティビティーの収集を有効にするには、以下のようにします。

1. CREATE EVENT MONITOR ステートメントを使用して、ACTIVITIES イベント・モニターを作成します。
2. COMMIT ステートメントを使用して、変更をコミットします。
3. SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを活動化します。 SET EVENT MONITOR STATE ステートメントを使用せずに、ACTIVITIES イベント・モニターに AUTOSTART のデフォルトを使用して、データベースが次回活動化されたときにこのイベント・モニターが活動化されるようにする方法もあります。複数の ACTIVITIES イベント・モニターを定義する場合は、AUTOSTART オプションを使用しないでください。
4. COMMIT ステートメントを使用して、変更をコミットします。
5. ALTER SERVICE CLASS、ALTER WORK ACTION SET、ALTER THRESHOLD、または ALTER WORKLOAD ステートメントを使用してどのオブジェクトに対してアクティビティーを収集するかを示し、COLLECT ACTIVITY DATA キーワードを指定します。
6. COMMIT ステートメントを使用して、変更をコミットします。

タスクの結果

注: 個々のアクティビティー収集は、ワークロード管理統計収集より高コストです。可能な限り少数のアクティビティーを収集するようアクティビティー収集をセットアップしてください。例えば、特定のアプリケーションによってサブミットされたアクティビティーを調査する必要がある場合、そのアプリケーション専用のワークロードまたはサービス・クラスを作成してそのアプリケーションを隔離し、そのワークロードまたはサービス・クラスに対する収集のみを有効にすることができます。

アクティビティーをキャプチャーする必要が常に事前にわかるとは限りません。例えば、ある照会の実行に長時間かかっている場合、後に分析するためにその情報を収集する必要があるかもしれません。この場合、アクティビティーが既にシステムに入っているため、この DB2 ワークロード管理オブジェクトに対して COLLECT ACTIVITY DATA キーワードを指定するには遅すぎます。この場合には、WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを使用できます。 WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーは、実行しているアクティビティーに関する情報をアクティブな ACTIVITIES イベント・モニターに送信します。アプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を使用して、収集するアクティビティーを指定します。プ

ロシージャーが呼び出されると、このアクティビティーに関する情報は直ちに ACTIVITIES イベント・モニターに送信されます。アクティビティーの完了まで待つ必要はありません。

設計アドバイザーへのアクティビティー情報のインポート

アクティビティー・イベント・モニターによって収集されたアクティビティーを設計アドバイザーにインポートして、これらアクティビティーがアクセスするデータベース・オブジェクトについての決定をする際に役立てることができます。

このタスクについて

設計アドバイザーにインポートされるアクティビティーは、必ず、COLLECT ACTIVITY DATA WITH DETAILS オプションまたは COLLECT ACTIVITY DATA WITH DETAILS AND VALUES オプションを使用して収集されたものです。COLLECT ACTIVITY DATA WITHOUT DETAILS オプションは不十分で、設計アドバイザーが必要とするステートメント・テキストをキャプチャーしません。

アクティビティー情報をアクティビティー・イベント・モニター表から設計アドバイザーにインポートするには、**db2advis** コマンドに **-wlm** パラメーターを付け、その後に以下のようなその他のパラメーターを付けて実行します。

1. アクティビティー・イベント・モニター名
2. オプション: ワークロード名またはサービス・クラス名
3. オプション: 開始時刻と終了時刻

例えば、SAMPLE データベースで DB2ACTIVITIES イベント・モニターによって収集されたすべてのアクティビティーに関する情報をインポートするには、次のコマンドを使用します。

```
db2advis -d SAMPLE -wlm DB2ACTIVITIES
```

注: 設計アドバイザーのコマンド行インターフェースでは、アクティビティー・イベント・モニター表からの情報しかインポートできません。

アクティビティーの取り消し

アクティビティーが消費するリソースが多すぎる場合、または実行時間が長すぎる場合は、そのアクティビティーをキャンセルできます。アクティビティーのキャンセルの方が、そのアクティビティーをサブミットしたアプリケーションを強制終了するのに比べ、影響範囲が小さいです。キャンセルされたアクティビティーにより、ユーザーに SQL4725N が戻されますが、接続を終了したり、その他のユーザー・アクティビティーに影響を与えたりはしません。アプリケーションを強制終了すると、接続とユーザー・アクティビティーの両方が終了します。

このタスクについて

アクティビティーを明示的にキャンセルできるのは、コーディネーター・アクティビティーが現在そのアクティビティーの要求を処理している場合のみです。IDLE 状態 (つまり、処理されている要求がない状態) でアクティビティーをキャンセルした場合、アクティビティーは CANCEL_PENDING 状態に置かれ、次に受信される要

求でキャンセルされます。例えば、CURSOR アクティビティーをフェッチの間にキャンセルしようとすると、SQL4725N エラーはキャンセルの後の次のフェッチまでユーザーに戻されません。

ロード・ユーティリティーおよびストアード・プロシージャーを含むすべてのユーザー・アクティビティーはキャンセル可能です。

手順

1. キャンセルするアクティビティーを識別します。
   WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用すると、アプリケーションで実行されているアクティビティーを識別できます。また、アクティビティーが実行している作業を識別するために WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES の情報だけでは不十分な場合、特定のアクティビティーについての詳細をさらに表示するために MON_GET_ACTIVITY_DETAILS_COMPLETE 表関数を使用することもできます。
2. WLM_CANCEL_ACTIVITY ストアード・プロシージャーを使用してアクティビティーをキャンセルします。 ストアード・プロシージャーには、以下の引数をとります。*application_handle*、*uow_id*、および *activity_id*。このストアード・プロシージャーの使用方法は、394 ページの『シナリオ: 完了に時間がかかり過ぎているアクティビティーの識別』の例を参照してください。

不良アクティビティーに関する情報のキャプチャーと調査についてのガイドライン

このトピックでは、不良アクティビティーに関する情報のキャプチャーと調査についてのガイドラインが提供されています。

最初に、不良アクティビティーであると見なす一連の基準を設定します。以下に例を示します。

- 見積コストの小さいアクティビティー用のサービス・クラスで実行されているものの、1 時間を超えて実行されているアクティビティー
- 異常なほど大量の行を戻すアクティビティー
- TEMPORARY 表スペースを異常に大量に消費するアクティビティー

次に、こうした基準を記述したしきい値を作成して、COLLECT ACTIVITY DATA WITH DETAILS アクションを組み込みます。そのしきい値に違反すると、アクティビティー完了時に、しきい値に違反したそのアクティビティーに関する情報がアクティブな ACTIVITIES イベント・モニターに送信されます。

例えば、3 時間を超えて実行されているデータベース・アクティビティーに関する情報を収集するには、以下のようなしきい値を作成します。

```
CREATE THRESHOLD LONGRUNNINGACTIVITIES
   FOR DATABASE ACTIVITIES ENFORCEMENT DATABASE
   WHEN ACTIVITYTOTALTIME > 3 HOURS COLLECT ACTIVITY DATA WITH DETAILS
   CONTINUE
```

この例では、少なくとも 3 時間以上実行されている照会のみがモニターされます。この例のように、DB2 ワークロード・マネージャーを使用したモニターは、特定の

照会のサブセットにのみ適用するのであれば負荷が少なくなります。グローバル・データベース・レベルではなくユーザー定義スーパークラスのレベルのしきい値を作成することによって、この例をさらに詳細化できます。有効範囲を絞り込んだ次のようなモニターが目的に合えば、モニターのコストをさらに削減できるとともに、必要なレベルの情報のみが提供されます。

```
CREATE SERVICE CLASS LONGQUERIES
   AGENT PRIORITY 20
   PREFETCH PRIORITY LOW

CREATE THRESHOLD LONGRUNNINGACTIVITIES2
   FOR SERVICE CLASS LONGQUERIES ACTIVITIES ENFORCEMENT DATABASE
   WHEN ACTIVITYTOTALTIME > 3 HOURS COLLECT ACTIVITY DATA WITH DETAILS
   CONTINUE
```

. このしきい値用に作成されるサービス・クラスのエージェント優先順位とプリフェッチ優先順位には、低い値が割り当てられます。長期実行照会に使用されることを目的としたものだからです (この SQL ステートメントは、UNIX オペレーティング・システムおよび Linux 上で使用できます。Windows オペレーティング・システム上では、エージェント優先順位として -6 に置き換えてください)。

データ・サーバーが何らかの作業を実行した後、しきい値違反およびアクティビティー・イベント・モニターに書き込まれた情報を分析できます。また DML アクティビティーにはアクティビティー・イベント・モニターに書き込まれたステートメント・テキストおよびコンパイル環境情報があるので、こうした情報に対して **DB2 Explain** を実行してアクティビティーのパフォーマンスをさらに調査することができます。

ワークロード管理のパフォーマンスのモデル化

システムのワークロードはモデル化が可能です。これは、アクティビティーの到着比率分布 (しばしば、その反対である*到着間隔* 時間分布の形で測定される) が定める比率でシステムに到着するアクティビティーのセット、およびサービス時間分布に従ってアクティビティーがシステムで実行に費やす時間の量、という形でモデル化できます。

到着間隔時間とは、1 つのアクティビティーの到着から次のアクティビティーの到着までの間の時間です。サービス時間は、アクティビティーがシステム上で実行に費やす時間です。例えば、0 秒の時点で照会をサブミットし、これがキューで 2 秒間費やし、5 秒の時点で完了する場合、サービス時間は 5 - 2 = 3 秒となります。サービス時間は、他の作業がシステム上で実行されていないことを前提としています (つまり、これは実測上の実行時間ではなく、独立してアクティビティーを実行するのに要するであろうと考えられる時間です)。 DML アクティビティーの場合、サービス時間分布は、アクティビティーのプロセッサー時間と入出力時間の両方を考慮に入れた見積コスト (timeron 単位) を使用して概算することができます。

システムのワークロード・モデルを作成するには、システム上のアクティビティーの到着間隔時間分布およびサービス時間分布を測定します。到着間隔時間分布およびおおよそのサービス時間分布 (見積コストを使用) は、サービス・サブクラスまたはワーク・クラス (ワーク・アクションを使用) の拡張集約アクティビティー統計、および統計イベント・モニターを使用して入手することができます。これらの統計は、デフォルトでは収集されません。詳しくは、以下を参照してください。

- 280 ページの『DB2 ワークロード管理オブジェクトの統計』
- A gentle introduction to histograms
- Understanding the six histograms of DB2 workload management
- Visualizing and deriving statistics from DB2 histograms using SQL

例: 事後分析のためのアクティビティーに関する情報のキャプチャー

ワークロード管理フィーチャーを使用して、後ほど分析するために、アクティビティーに関する情報をキャプチャーできます。

MYSCHEMA.MYSLOWSTP というストアード・プロシージャーがあり、通常より実行速度が遅いとします。この状況に関して苦情が報告されたので、速度が低下した原因を調査することにします。ストアード・プロシージャーの実行中にその調査を行うことが実際的ではない場合には、そのストアード・プロシージャー・アクティビティーおよびそのアクティビティーにネストしているすべてのアクティビティーに関する情報をキャプチャーできます。

DB2ACTIVITIES というアクティブ・アクティビティー・イベント・モニターがあるとします。CALL ステートメントのワーク・クラスを作成して、MYSCHEMA.MYSLOWSTP ストアード・プロシージャーのスキーマに適用できます。その後、CALL アクティビティーおよびすべてのネストされたアクティビティーを、アクティビティー・コレクションが使用可能なサービス・クラスにマップするようなワーク・アクションを作成できます。CALL アクティビティーおよびそのすべてのネストされているアクティビティーはイベント・モニターに送信されます。以下は、DB2 ワークロード管理オブジェクトを作成するために必要な DDL の例です。

```
CREATE SERVICE CLASS SC1;
CREATE WORKLOAD WL1 APPLNAME ('DB2BP') SERVICE CLASS SC1;
CREATE SERVICE CLASS PROBLEMQUERIESSC UNDER SC1 COLLECT ACTIVITY DATA ON COORDINATOR WITH DETAILS;

CREATE WORK CLASS SET PROBLEMQUERIES
(WORK CLASS CALLSTATEMENTS WORK TYPE CALL ROUTINES IN SCHEMA MYSCHEMA);

CREATE WORK ACTION SET DATABASEACTIONS FOR SERVICE CLASS SC1 USING WORK CLASS SET PROBLEMQUERIES
(WORK ACTION CAPTURECALL ON WORK CLASS CALLSTATEMENTS MAP ACTIVITY WITH NESTED TO PROBLEMQUERIESSC);
```

MYSCHEMA.MYSLOWSTP ストアード・プロシージャーの実行後に以下の照会を発行すると、アクティビティーのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を取得できます。

```
SELECT AGENT_ID,
       UOW_ID,
       ACTIVITY_ID
FROM ACTIVITY_DB2ACTIVITIES
WHERE SC_WORK_ACTION_SET_ID = (SELECT ACTIONSETID
                              FROM SYSCAT.WORKACTIONSETS
                              WHERE ACTIONSETNAME = 'DATABASEACTIONS')
AND SC_WORK_CLASS_ID = (SELECT WORKCLASSID
                    FROM SYSCAT.WORKCLASSES
                    WHERE WORKCLASSNAME = 'CALLSTATEMENTS'
                    AND WORKCLASSSETID =
                    (SELECT WORKCLASSSETID FROM SYSCAT.WORKACTIONSETS WHERE ACTIONSETNAME
                    = 'DATABASEACTIONS'));
```

キャプチャーされたアクティビティーのアプリケーション・ハンドルが 1 で、作業単位 ID は 2、アクティビティー ID は 3 であると想定すると、以下の結果が生成されます。

```
AGENT_ID             UOW_ID      ACTIVITY_ID
==================== =========== ===========
                   1           2           3
```

この情報を使用して、ACTIVITY_DB2ACTIVITIES 表および ACTIVITYSTMT_DB2ACTIVITIES 表に対して以下の照会を発行し、アクティビティーが時間を費やす対象を判別できます。

```
WITH RAH (LEVEL, APPL_ID, PARENT_UOW_ID, PARENT_ACTIVITY_ID,
          UOW_ID, ACTIVITY_ID, STMT_TEXT, TIME_CREATED, TIME_COMPLETED) AS
  (SELECT 1, ROOT.APPL_ID, ROOT.PARENT_UOW_ID,
          ROOT.PARENT_ACTIVITY_ID, ROOT.UOW_ID, ROOT.ACTIVITY_ID,
          ROOTSTMT.STMT_TEXT, ROOT.TIME_CREATED, ROOT.TIME_COMPLETED
   FROM ACTIVITY_DB2ACTIVITIES ROOT, ACTIVITYSTMT_DB2ACTIVITIES ROOTSTMT
   WHERE ROOT.APPL_ID = ROOTSTMT.APPL_ID AND ROOT.AGENT_ID = 1
     AND ROOT.UOW_ID = ROOTSTMT.UOW_ID AND ROOT.UOW_ID = 2
     AND ROOT.ACTIVITY_ID = ROOTSTMT.ACTIVITY_ID AND ROOT.ACTIVITY_ID = 3
  UNION ALL
   SELECT PARENT.LEVEL +1, CHILD.APPL_ID, CHILD.PARENT_UOW_ID,
          CHILD.PARENT_ACTIVITY_ID, CHILD.UOW_ID,
          CHILD.ACTIVITY_ID, CHILDSTMT.STMT_TEXT, CHILD.TIME_CREATED,
          CHILD.TIME_COMPLETED
   FROM RAH PARENT, ACTIVITY_DB2ACTIVITIES CHILD,
        ACTIVITYSTMT_DB2ACTIVITIES CHILDSTMT
   WHERE PARENT.APPL_ID = CHILD.APPL_ID AND
         CHILD.APPL_ID = CHILDSTMT.APPL_ID AND
         PARENT.UOW_ID = CHILD.PARENT_UOW_ID AND
         CHILD.UOW_ID = CHILDSTMT.UOW_ID AND
         PARENT.ACTIVITY_ID = CHILD.PARENT_ACTIVITY_ID AND
         CHILD.ACTIVITY_ID = CHILDSTMT.ACTIVITY_ID AND
         PARENT.LEVEL < 64
  )
SELECT UOW_ID, ACTIVITY_ID, SUBSTR(STMT_TEXT,1,40),
               TIMESTAMPDIFF(2, CHAR(TIME_COMPLETED - TIME_CREATED)) AS
               LIFE_TIME
FROM RAH
  ORDER BY UOW_ID, ACTIVITY_ID;
```

結果は以下のようになります。

```
UOW_ID ACTIVITY_ID STMT_TEXT                                LIFE_TIME
====== =========== ======================================== =============
2      3           CALL SLOWPROC                                   1000
2      4           SELECT COUNT(*) FROM ORG                           1
2      5           SELECT * FROM MYHUGETABLE                        999
```

ストアード・プロシージャーがその時間のほとんどを費やしたのは MYHUGETABLE 表の照会であることを、この結果は示しています。次のステップは、MYHUGETABLE 表に対して加えられたどの変更が、その表に対する照会の実行速度を低下させる原因となっているかを調査することです。

多くのストアード・プロシージャーが同時に実行されると、分析の実行時にオーバーヘッドが一層大きくなります。この問題を解決するには、ワークロードおよびサービス・クラスを作成して、特定の許可 ID またはアプリケーション (あるいはその両方) ごとに発行されるストアード・プロシージャーを実行します。その後、前述の方法を使用して、ストアード・プロシージャーの動作を分析します。

第 5 章 オペレーティング・システムのワークロード・マネージャーとの統合

可能な場合は、DB2 ワークロード管理をオペレーティング・システムのワークロード・マネージャーと併せて使用してください。追加の機能を使用できます。

DB2 ワークロード管理とオペレーティング・システムのワークロード・マネージャーの統合のポイントは、DB2 サービス・クラスです。 DB2 サービス・クラスを定義するときに、CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントの OUTBOUND CORRELATOR オプションを使用して、DB2 サービス・クラスとオペレーティング・システムのワークロード・マネージャーのクラスの間のマッピングを作成します。

OUTBOUND CORRELATOR が設定されると、次のアクティビティーが開始されるときは、DB2 サービス・クラス内のすべてのスレッドが OUTBOUND CORRELATOR を使用してオペレーティング・システムのワークロード・マネージャーに関連付けられます。

AIX ワークロード・マネージャーと DB2 ワークロード管理の統合

AIX オペレーティング・システムでは、DB2 サービス・クラスと AIX WLM クラスの統合オプションを使用できます。この統合により、各サービス・クラスに割り振られるプロセッサー・リソース量を制御できます。

パフォーマンス目標を満たすうえで AIX WLM 制御をインプリメントしなくてもよい場合がありますが、AIX WLM を使用する必要がないとしても、AIX WLM が提供する AIX クラスごとのオペレーティング・システム統計は、モニター作業や調整作業にしばしば役立ちます。

AIX WLM は、クラスにプロセッサー・リソースの相対量または絶対量をシェアとして割り当てます。クラスに対する制御は動的に変更でき、即時に有効になるというメリットがあります。 AIX CPU 相対シェアが必要制御レベルを満たさない場合は、CPU リソースの強制最大パーセンテージを割り当てることもできます。こうするとオフピーク時に好都合な CPU 相対割り振りの柔軟性の一部を放棄することになりますが、CPU 時間リソース割り振りに対する強制最大限度を使用して最適かつ確実に制御することもできるようになります。

DB2 サービス・クラスと AIX クラスとの間の推奨マッピング

AIX WLM プロセッサー制御を活用するには、DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスの 1:1 マッピングを使用します。 DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスの 1:1 マッピングを使用して、DB2 サービス・クラスごとに個別に AIX プロセッサー・リソースを調整することで、ビジネス優先度の目標を達成することができます。

以下の図は、DB2 ワークロード管理と AIX ワークロード・マネージャーの統合を示しています。サービス・スーパークラスとサービス・サブクラスのレベルでの、

DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間での 1:1 マッピングに注目してください。

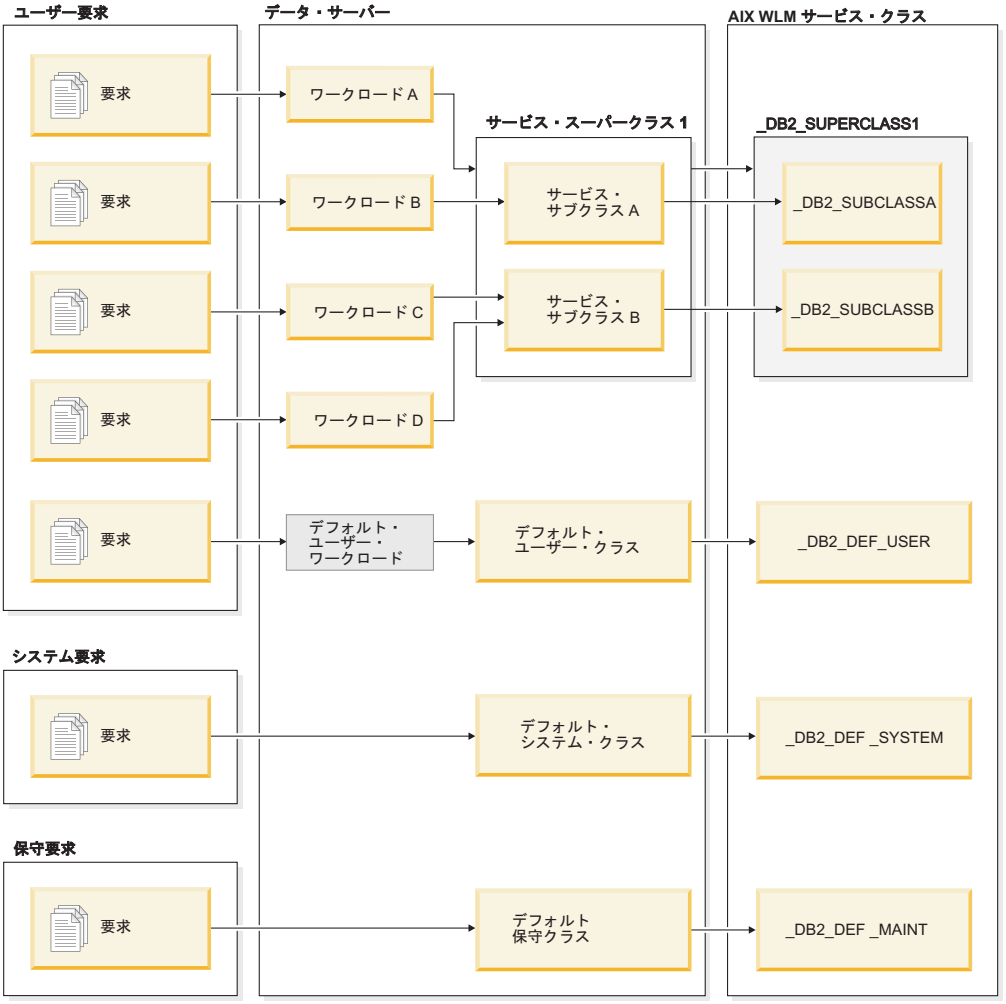


図 52. DB2 ワークロード管理と AIX ワークロード・マネージャーの統合

前の図で示す例のように、DB2 環境が単一の DB2 インスタンス内の単一のデータベースで構成される場合、DB2 サービス・クラスと AIX ワークロード・マネージャーのクラスとの間での直接のマップが可能です。それぞれの DB2 サービス・スーパークラスは、対応する AIX ワークロード・マネージャーのサービス・スーパークラスを持つことができ、それぞれの DB2 サービス・サブクラスは、対応する AIX サービス・サブクラスにマップすることができます。

DB2 環境が複数のデータベースと DB2 インスタンスで構成されている状況では、リソース制御の候補として複数のレベルが考えられます。 AIX ワークロード・マネージャーはスーパークラスとサブクラスの 2 つのレベルの階層をサポートしているので、DB2 環境の 2 つのレベルだけは AIX ワークロード・マネージャーのクラスにいつでもマップできます。以下の図は、複数のデータベースのそれぞれが複数のスーパークラスを持つ場合の、1:1 マッピングを実現するための 1 つの方法を示しています。ここで、各データベースにはその固有の AIX ワークロード・マネージャーのスーパークラスがあり、各 DB2 サービス・スーパークラスは AIX ワークロー

ド・マネージャーのサブクラスにマップされています。

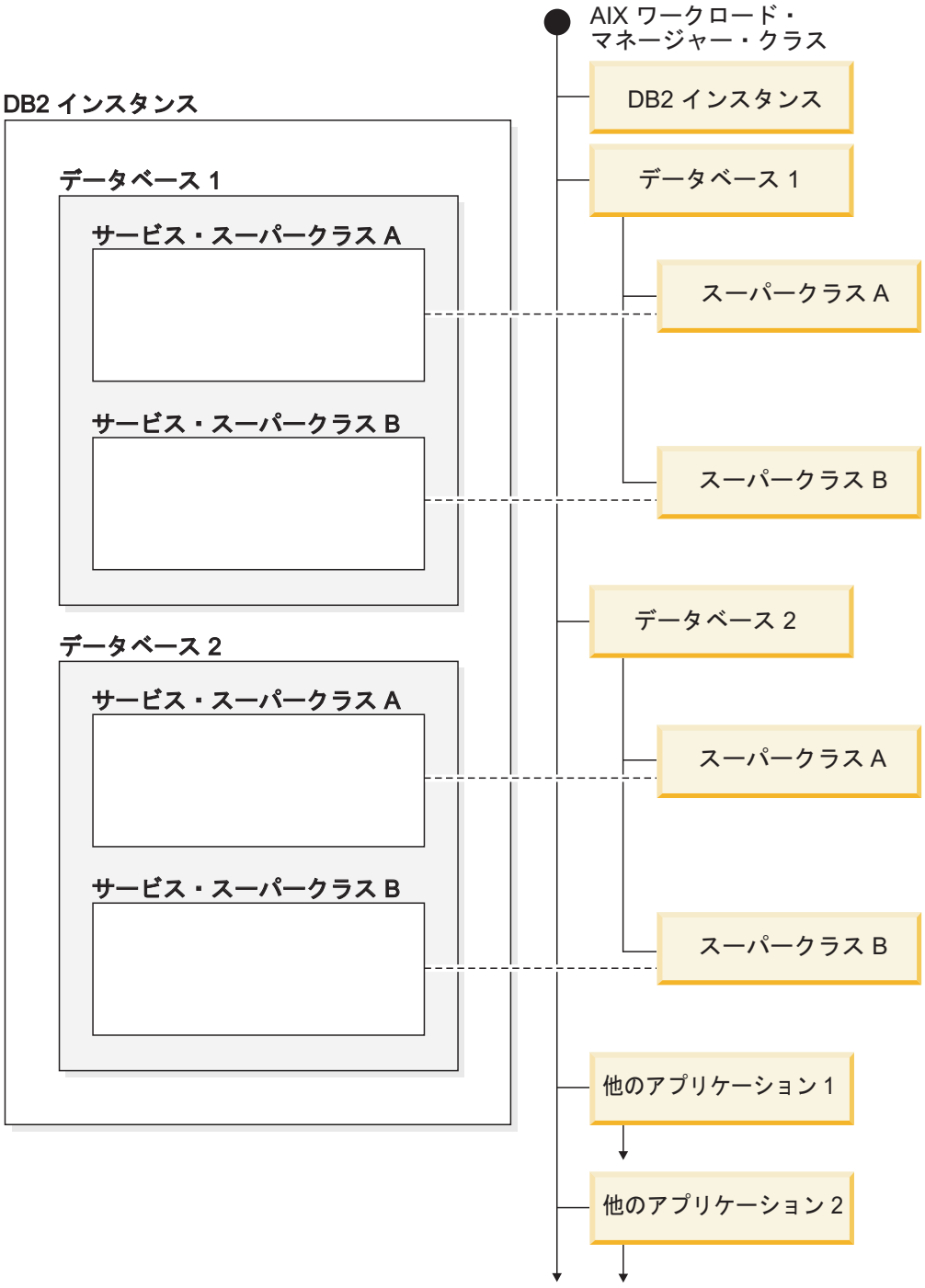


図 53. AIX クラスにマップされる DB2 サービス・クラス (DB2 サービス・スーパークラスのみを含む)

代替構成として、各 DB2 サービス・スーパークラスをその固有の AIX ワークロード・マネージャーのスーパークラスにマップすることができます。この例では結果として 4 つのスーパークラスになります。この状況では、データベース・レベルのリソース制御は、AIX ワークロード・マネージャーのサービス・クラス定義で明示的に表されます。

以下の図は、それぞれサービス・スーパークラスとサービス・サブクラスを持つ複数のデータベースがある場合に、1:1 マッピングを実現するための 1 つの方法を示しています。ここで、各データベースは AIX スーパークラスに対応し、各 DB2 サービス・サブクラスは AIX ワークロード・マネージャーのサブクラスにマップされます。 DB2 サービス・スーパークラスは、AIX ワークロード・マネージャーのサービス・クラス定義には明示的に示されません。

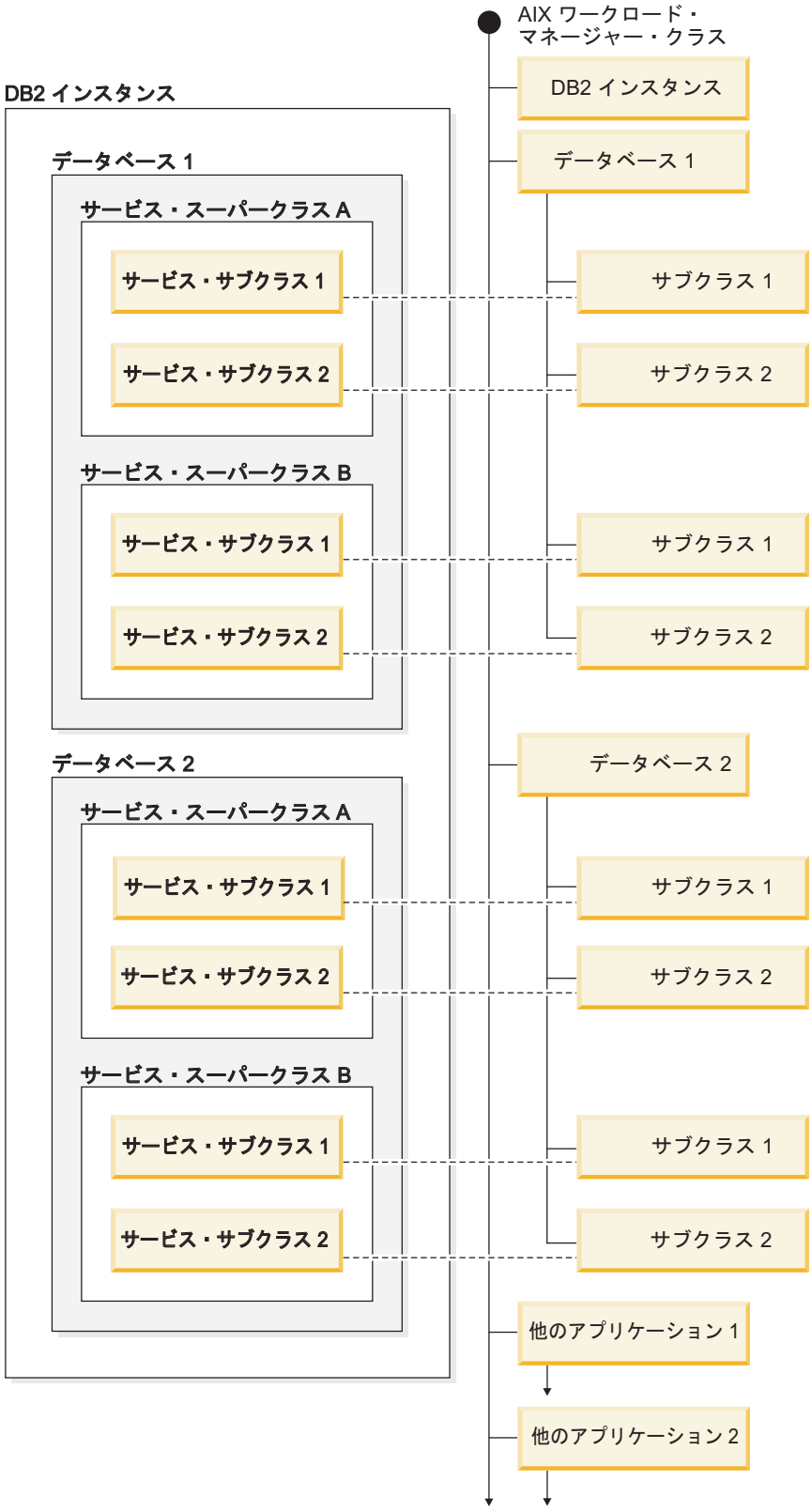


図 54. AIX ワークロード・マネージャーのクラスにマップされる DB2 サービス・クラス (DB2 サービス・サブクラスを含む)

DB2 サービス・クラスと AIX のクラスとの間のマッピングの定義

DB2 サービス・クラスと AIX ワークロード・マネージャーのクラスとの間のマッピングは、CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントの OUTBOUND CORRELATOR キーワードを使用して、DB2 サービス・クラスに指定されます。

DB2 データ・サーバーを使用して AIX ワークロード・マネージャーのクラスをセットアップする手順は以下のとおりです。

1. DB2 サービス・スーパークラスとサービス・サブクラスを作成し、OUTBOUND CORRELATOR タグを指定します。
2. それに対応する AIX のクラスを作成します。
3. DB2 ワークロード管理と AIX ワークロード・マネージャーとのマッピングを含めるための、関連した AIX ワークロード・マネージャー規則ファイルを、タグ列の下で OUTBOUND CORRELATOR タグを使用して作成します。
4. AIX ワークロード・マネージャーを開始します。
5. 必要な場合、この AIX ワークロード・マネージャー構成をアクティブに設定します。

スレッドと DB2 サービス・クラスが結合されると、DB2 データ・サーバーは該当する AIX ワークロード・マネージャー API を呼び出して、そのスレッドを対応する AIX サービス・クラスと関連付けます。 DB2 データ・サーバーは、OUTBOUND CORRELATOR パラメーターで設定されているアプリケーション・タグを渡すことによって、そのスレッドのターゲット AIX サービス・クラスを AIX ワークロード・マネージャーに送信します。

AIX ワークロード・マネージャーが正しくインストールおよび構成されており、アクティブになっていることを確認する必要があります。 DB2 データ・サーバーが AIX ワークロード・マネージャーと通信できない場合、メッセージが db2diag ログ・ファイルおよび DB2 管理者ログに記録されます。データベース・アクティビティーは続行します。

DB2 データ・サーバーは、AIX ワークロード・マネージャーに渡す OUTBOUND CORRELATOR 値が AIX ワークロード・マネージャーによって認識されているかどうかを検出できません。 DB2 サービス・クラスに指定されている値が、DB2 スレッドを AIX サービス・クラスにマップするアプリケーション・タグと一致していることを確認する必要があります。 OUTBOUND CORRELATOR 値が AIX ワークロード・マネージャーによって認識されていない場合、データベース・アクティビティーは実行を続行します。

他の注目すべき点には以下があります。

- DB2 サービス・クラスは、AIX ワークロード・マネージャーの継承フィーチャーを処理することができません。継承は AIX サービス・クラスのデフォルト設定です。継承は継承属性を NO に設定することで明示的に使用不可にする必要があります。AIX ワークロード・マネージャーの継承によって、すべての子スレッドおよびプロセスは親スレッドまたはプロセスと同じクラスに強制的にマップされます。継承が使用可能な場合、DB2 ワークロード管理は、タグを使用してスレッドの AIX ワークロード・マネージャー・クラスを変更することはできません。

この制限により、DB2 ワークロード管理と AIX ワークロード・マネージャーの一切の統合は使用できなくなります。 DB2 データ・サーバーは、AIX ワークロード・マネージャーの継承が使用可能かどうかを検出できないため、継承が使用可能な場合でもエラー・メッセージは発行されません。

- DB2 サービス・クラスには、AIX ワークロード・マネージャーの手動割り当てフィーチャーとの互換性はありません。手動割り当てフィーチャーを使用した場合、ユーザーは特定の AIX ワークロード・マネージャーのクラスに処理を手動で割り当てることができます。 DB2 プロセスを手動で割り当てることにより、プロセス内のすべてのスレッドはターゲット AIX ワークロード・マネージャーのクラスに割り当てられ、DB2 サービス・クラスのマッピング・ロジックは無効になり、結果は予測できません。

AIX ワークロード・マネージャーについて詳しくは、 http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp の AIX インフォメーション・センターを参照してください。

AIX クラスに対するプロセッサー制御の設定

AIX ワークロード・マネージャーは、各サービス・クラスに割り振られるプロセッサー・リソース量を制御するために使用できます。オプションには、各サービス・クラスに対するプロセッサー・リソースの最小、最大、または相対比率の共有の設定が含まれます。

AIX ワークロード・マネージャーと DB2 ワークロード管理を統合する場合、プロセッサー・リソース割り振りのみがサポートされます。 AIX クラスのメモリーおよび入出力設定を行わないでください。 DB2 データベース・レベルのメモリーは、異なる DB2 サービス・クラスのすべてのエージェントで共有されるため、異なるサービス・クラス間でメモリー割り振りを分割することはできません。 AIX レベルの入出力制御は、DB2 エンジン・スレッド化モデルをサポートしません。入出力を制御するには、DB2 サービス・クラスのプリフェッチャー優先順位属性を使用して、異なる DB2 サービス・クラス間での入出力の優先順位を区別することができます。

AIX を使用してサービス・クラスに割り振られるプロセッサー・リソースの量を制御する場合は、その DB2 サービス・クラスのエージェント優先順位設定も変更しないでください。プロセッサー・リソースへのアクセスを管理するには、これらのメカニズムの 1 つだけを使用します。サービス・クラスに AGENT PRIORITY および OUTBOUND CORRELATOR 値の両方を設定することはできません。詳しくは、87 ページの『サービス・クラスのエージェント優先順位』を参照してください。

AIX ワークロード・マネージャーの設定は、インスタンスに関与するすべての物理コンピューターで一貫していなければなりません。例えば、あるコンピューターで AIX サービス・クラスのリソース設定が高く設定されている場合、他のすべてのコンピューターのその AIX サービス・クラスに同じ設定を使用しなければなりません。リソース使用量の設定がコンピューター間で一貫していない場合、同じ AIX サービス・クラスで実行されている要求は、別のデータベース・メンバー上で異なるパフォーマンス・レベルを示します。この状態によって、AIX サービス・クラスにおける接続の全体のスループットが落ちる可能性があります。

Linux ワークロード管理と DB2 ワークロード管理の統合

Linux オペレーティング・システムでは、DB2 サービス・クラスと Linux クラス (制御グループ) の統合オプションを使用できます。この統合により、各サービス・クラスに割り振られるプロセッサー・リソース量を制御できます。使用可能にすると、DB2 サービス・クラスで実行されるすべてのスレッドは Linux のクラスにマップされ、定義されたプロセッサー・リソース制御の制約を受けます。

Linux ワークロード管理サポートを使用するには、64 ビット・システムの Linux カーネルのバージョン 2.6.26 以降と、libcgroup ライブラリー・パッケージが必要です。

Linux ワークロード管理はスーパークラスとサブクラスを持つクラスの階層をサポートしており、サブクラスのプロセッサー・シェアは親クラスのシェアの比率に応じて分割されます。これらのシェアにより、システムの全スレッドが常時実行しつつ、各スレッドは Linux クラスに割り当てられたシェア数に応じたプロセッサー時間を受け取るという、プロセッサー・リソースの制御方式が提供されます。

Linux オペレーティング・システム上のプロセッサー・リソースは、Linux ワークロード管理のデフォルト・クラスを基準としたシェアで割り当てられます。このデフォルト・クラスのプロセッサー・シェアの値は、デフォルトでは 1024 です。Linux のクラスが他に定義されていない場合は、すべてのスレッドがこのデフォルト・クラスで実行されます。1024 と同じシェア値を持つクラスを定義すると、そのクラスは、デフォルトのプロセッサー・シェアを持つ Linux デフォルト・クラスと同量のプロセッサー・リソースを受け取ります。同様に、シェアが 2048 のクラスは、デフォルト・クラスの 2 倍のターゲット・プロセッサー使用割り当て量を受け取ります。より複雑なシステムの場合、Linux デフォルト・クラスのプロセッサー・シェアを引き上げることを考慮する必要があります。これによりシステム全体のシェアの細分度が増し、プロセッサー・リソースをより細かく割り当てることができます。

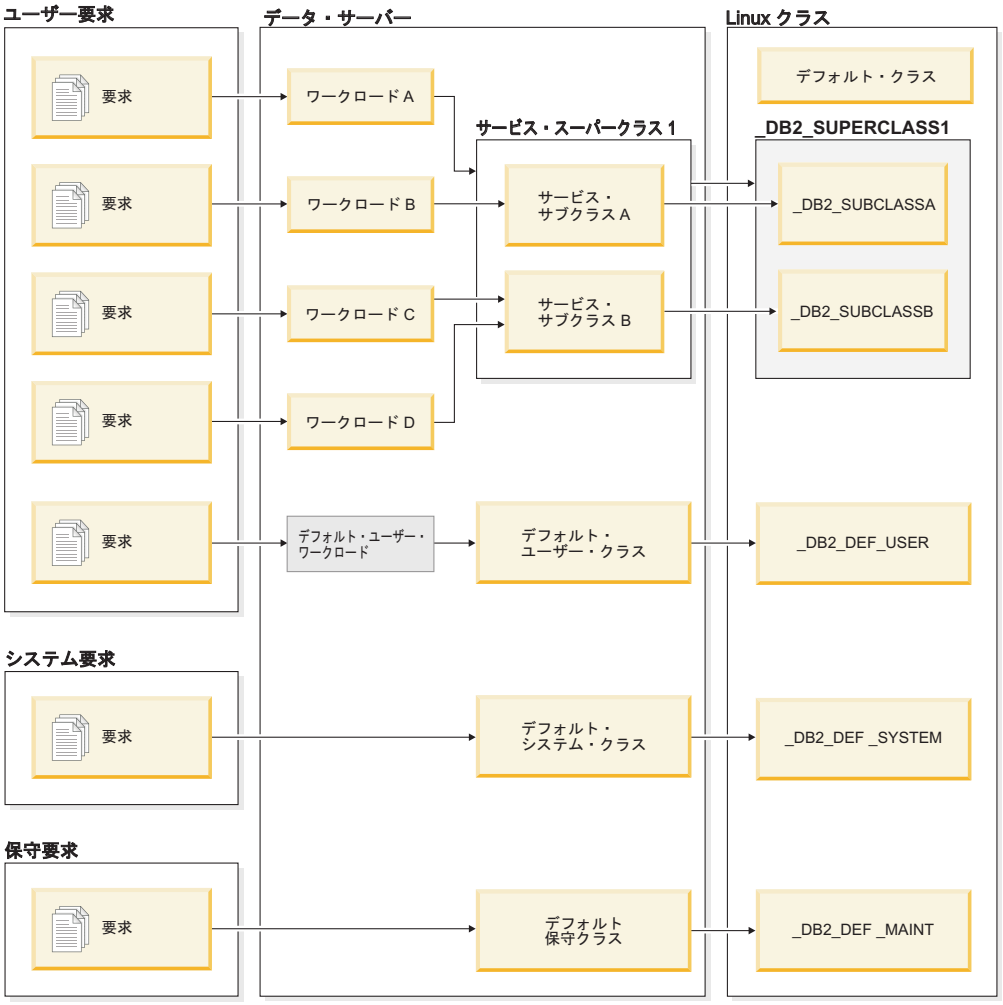
DB2 サービス・クラスと Linux クラスとの間の推奨マッピング

DB2 サービス・クラスと Linux のクラスは 1:1 のマッピングとなるようにしてください。こうすることで、各 DB2 サービス・クラス内のアクティビティーに割り当てられた Linux プロセッサー・シェアを、ビジネス優先度に従って個別に調整できます。すべての DB2 サービス・クラスを Linux WLM クラスと関連付けることは重要です。関連付けは、各サービス・スーパークラスとサブクラスにアウトバウンド相関関係子を設定することによって、あるいは親サービス・クラスからサブクラスへの継承を通じて行います。この対象としては、デフォルトの SYSDEFAULTSYSTEMCLASS、SYSDEFAULTMAINTENANCECLASS、および SYSDEFAULTUSERCLASS サービス・クラスが含まれます。

次の図は、同一ユーザー定義サービス・スーパークラス下の 2 つの DB2 サービス・サブクラスを、共通スーパークラス下の Linux サブクラスに 1:1 でマップする方法を示しています。この例では、各 DB2 サービス・サブクラスに対して 2 つのワークロードによって識別され、割り当てられた作業は、対応する Linux サブクラス (_DB2_SUBCLASSA、_DB2_SUBCLASSB) によるプロセッサー・リソース制御から制約を受けます。DB2 ワークロード管理のデフォルトのサービス・クラス (_DB2_DEF_USER、_DB2_DEF_SYSTEM、_DB2_DEF_MAINT) に対応する 3 つの

Linux クラスも示しています。 DB2 ワークロード管理を Linux ワークロード管理と統合する場合は常に、これらの追加の Linux クラスを作成してデフォルトの DB2 サービス・クラスに対応させる必要があります。ボトルネックを回避するには、DB2 のデフォルトのシステム・クラスに対応する Linux クラスが、DB2 アクティビティーがマップされる他のどの Linux クラスよりも大きなプロセッサー・シェアを受け取り、デフォルトの保守クラスに対応する Linux クラスは、他のどのクラスよりも小さなプロセッサー・シェアを受け取る必要があります。

図 55. DB2 ワークロード管理と Linux ワークロード管理の統合



DB2 サービス・クラスと Linux ワークロード管理クラスの間のマッピング定義

DB2 ワークロード管理を Linux ワークロード管理 (オペレーティング・システムのサービスとして実行される) と統合するためのステップは、以下のとおりです。

1. 制御グループ構成ファイル `/etc/cgconfig.conf` を編集して、Linux クラス、クラス権限、およびプロセッサー・シェアを定義します。どのような Linux クラスを作成するかは、データ・サーバーが実行する作業のビジネス優先度で決まる条件によって異なります。例えば、特定の作業のソースに基づいてプロセッサ

ー・リソースを適用する場合は、作業を識別するワークロードによって作業が割り当てられることになる DB2 サービス・クラスに対応する Linux クラスを作成します。作成される DB2 サービス・クラスに対応する Linux クラスごとに、マッピングに使用する項目を定義します。 /etc/wlm/cgconf.conf 構成ファイルでは、次のセクションを設定する必要があります。

- group: Linux クラス名。例えば、グループ _class1 を指定すると、スーパークラス _class1 が作成されます。グループ _class1/_subclass1 を指定すると、スーパークラス _class1 の下にサブクラス _subclass1 が作成されます。
  - perm: Linux クラスにどのスレッドを割り当てるかを誰が制御できるか、および /etc/cgconfig.conf 構成ファイル内でクラスのプロセッサー・シェアを誰が変更できるかを決定する権限セクション。
    - task: 所有するスレッドを Linux ワークロード管理クラスで実行することができる、ユーザー ID (**uid**) およびグループ ID (**gid**)。 Linux ワークロード管理を DB2 ワークロード管理と連携できるようにするには、**uid** を DB2 インスタンス所有者ユーザー ID に設定する必要があります。
    - admin: Linux ワークロード管理クラスのプロセッサー・シェアを変更できるユーザー ID (**uid**) およびグループ ID (**gid**)。
  - cpu: プロセッサー・シェアの定義セクション。
    - cpu.shares: この Linux クラスに割り当てるシェア (デフォルト・クラスを基準とした相対指定)。

  /etc/cgconfig.conf 構成ファイルには、上記のセクションを以下の形式で含めなければなりません。

```
# Superclass name
group _name
{
   perm
   {
      task
      {
         uid = db2inst1;
         gid = db2iadm1;
      }
      admin
      {
         uid = db2inst1;
         gid = db2iadm1;
      }
   }

   cpu
   {
      cpu.shares = 1024;
   }
}
```

2. **service cgconfig start** コマンドで Linux ワークロード管理サービス・デーモンを開始し、次いで **db2start** コマンドで DB2 データ・サーバーを開始します。
3. DB2 サービス・クラスを Linux クラスの 1 つにマップするには、サービス・クラスを作成または変更するときに、OUTBOUND CORRELATOR 節に Linux クラス名を含めます。これによって、DB2 サービス・クラスでのスレッドが Linux の外部クラスに関連付けられます。

4. Linux の特定のクラスに割り当てられるスレッドを調べる場合は、/cgroup/*class_name*/tasks ファイルに対して cat コマンドを使用します。*class_name* は、目的の Linux クラスの名前を表します。Linux のユーザー定義クラスにマップされないスレッドはすべて、Linux デフォルト・クラスに割り当てられます。こうしたスレッドは、MOUNTPOINT/sysdefault にあります。MOUNTPOINT は cgconfig.conf 構成ファイルで定義されています。
5. Linux クラスを追加または除去するには、**service cgconfig stop** コマンドで Linux ワークロード管理サービスを停止し、変更を行ってから、このサービスを再始動する必要があります。サービスを停止すると、すべてのタスクがデフォルト・クラスに移動するため、システム全体に影響が及ぶことに注意してください。サービス・デーモンを開始するために /etc/init.d/cgred スクリプトを使用した場合は、**/etc/init.d/cgred stop** を発行してこれを停止してください。

DB2 ワークロード管理との統合が機能するためには、Linux ワークロード管理サービスを正しくインストールおよび構成し、アクティブにしておく必要があります。DB2 データ・サーバーが Linux ワークロード管理サービスと通信できない場合は、メッセージが db2diag ログ・ファイルおよび DB2 管理者ログに記録されます。データベース・アクティビティーは実行を続けます。

DB2 データ・サーバーは、外部ワークロード・マネージャーに渡すアウトバウンド相関関係子が Linux ワークロード管理によって認識されるかどうかを検出できません。DB2 サービス・クラスに指定されている OUTBOUND CORRELATOR 値が Linux クラス名と一致し、DB2 スレッドがその Linux クラスにマップされることを確認する必要があります。アウトバウンド相関関係子が認識されなくても、データベース・アクティビティーは実行を続けます。

例

以下の例は、DB2 ワークロード管理と統合することによって Linux ワークロード管理のプロセッサー制御を活用する方法を示しています。この例では、ユーザー定義の DB2 サービス・クラスを 2 つ作成します。1 つはバッチ・アプリケーション用 (BATCHAPPS)、もう 1 つはオンライン・アプリケーション用 (ONLINEAPPS) です。この例では、簡潔にするために、デフォルトのサービス・クラスは示されていません。DB2 サービス・クラスと Linux クラス間で推奨されている 1 対 1 のマッピングを作成するには、このデフォルトのサービス・クラスが含まれていなければなりません。オンライン・アプリケーションでは応答時間が重要になるため、Linux デフォルト・クラスで実行される作業の 3 倍の量のプロセッサー・シェア (3 x 1024 = 3072 シェア) を ONLINEAPPS サービス・クラスが受け取るようにします。バッチ・アプリケーションのビジネス優先度は比較的低いので、BATCHAPPS クラスには Linux デフォルト・クラスで実行される作業の半分のプロセッサー・リソース (1024 / 2 = 512 シェア) を割り当てます。システム上の他の作業はすべて、Linux デフォルト・クラスで実行されます。この例では DB2 ワークロード管理の 3 つのデフォルト・サービス・クラスに対応する Linux クラスは作成していないので注意してください。

この設定のためには、/etc/cgconfig.conf タスク・ファイルを編集して、まず対応する Linux の 2 つのクラス _BATCHAPPS および _ONLINEAPPS を作成し、それぞれのプロセッサー相対シェアを設定します。編集後のタスク・ファイルには、Linux クラスごとの次の 2 つの項目が含まれます。

```
# Superclass ONLINEAPPS
group _ONLINEAPPS
{
   perm
   {
      task
      {
         uid = db2inst1;
         gid = db2iadm1;
      }
      admin
      {
         uid = db2inst1;
         gid = db2iadm1;
      }
   }

   cpu
   {
      # 3 x 1024 = 3072 shares
      cpu.shares = 3072;
   }
}

# Superclass BATCHAPPS
group _BATCHAPPS
{
   perm
   {
      task
      {
         uid = db2inst1;
         gid = db2iadm1;
      }
      admin
      {
         uid = db2inst1;
         gid = db2iadm1;
      }
   }

   cpu
   {
      # 1024 / 2 = 512 shares
      cpu.shares = 512;
   }
}
```

各 Linux クラスにプロセッサー・シェアとして割り当てられる絶対プロセッサー時間 (パーセント) は、以下のとおりです。

表 62. Linux クラスに割り当てられるプロセッサー・シェアと絶対プロセッサー時間

Linux クラス	**シェア**	**絶対プロセッサー時間 (パーセント)**
デフォルト・クラス	1024 (デフォルト)	1024 / 4608 = 22%
_ONLINEAPPS	1024 x 3 = 3072	3072 / 4608 = 67%
_BATCHAPPS	1024 x ½ = 512	512 / 4608 = 11%
	合計 = 1024 + 3072 + 512 = 4608 シェア	

Linux WLM クラスを作成したら、次のコマンドで Linux ワークロード管理サービスを開始できます。

```
service cgconfig start
```

次に、以下のステートメントを使用して、関連付けられる DB2 サービス・クラスを作成します。

```
DB2 CREATE SERVICE CLASS BATCHAPPS OUTBOUND CORRELATOR '_BATCHAPPS'
DB2 CREATE SERVICE CLASS ONLINEAPPS OUTBOUND CORRELATOR '_ONLINEAPPS'
```

Linux クラスで実行されているスレッドを調べるには、cat コマンドを発行します。ビジネスに欠くことのできない _ONLINEAPPS 用の Linux クラスの場合、コマンドと出力は次のようになります。この Linux クラスでは、6 つのスレッドが実行されていることが分かります。

```
cat /cgroup/_ONLINEAPPS/tasks

1056
1087
1107
985
1036
1205
```

第 6 章 DB2 ワークロード管理のためのチュートリアル

このチュートリアルの演習は、DB2 ワークロード管理を実践的に紹介するために設計されています。個々の演習は、DB2 ワークロード管理で使用可能なワークロード管理フィーチャーの 1 つ以上を強調しています。

これらの演習は DB2 ワークロード管理フィーチャーの使用法に関する指針を示しているので、これらのフィーチャーを独自の目的に応じて適合させることができますが、独自のデータ・サーバーに応じて選択する初期構成が違う可能性があり、特定のワークロード管理の目標に基づいて選択する必要があることに注意してください。

始める前に

このチュートリアルは、SAMPLE データベースに対して実行するように設計されており、特に断りのない限り、DBADM または WLMADM 権限 (COLLECT ACTIVITY DATA 節のみを指定する場合は SQLADM 権限) を必要とします。また、以下のようにインスタンスを開始し、SAMPLE データベースを活動化してから続行する必要があります。

```
db2start
db2 activate db sample
```

これらの演習中に示されているコマンドや照会のステートメントの中には、非常に長いものもあります。これらのステートメントのほとんどはテキスト・ファイル `wlm-tutorial-steps.txt` 中にあるので、演習をひととおり行う際にこのファイルからコピーできます。さまざまな演習に必要なワークロードを表すスクリプトも組み込まれています。

`wlm-tutorial-steps.txt` とワークロード・スクリプトは、両方ともここにあります。

演習 1: デフォルトの DB2 ワークロード管理オブジェクトを使用した基本モニターから始める

この演習では、デフォルトのワークロード・オブジェクトおよびサービス・クラス・オブジェクトから入手できる基本型のモニター情報について説明します。

時間の見積もり: 20 分から 25 分

デフォルトでは、データベースごとにユーザー・ワークロード (SYSDEFAULTUSERWORKLOAD) およびデフォルトのユーザー・サービス・クラス (SYSDEFAULTUSERCLASS) が常に作成されます。これらのデフォルトのオブジェクトを使用すると、ユーザー定義のワークロードやサービス・クラスを作成せずに、新しい DB2 ワークロード管理モニター・フィーチャーを活用できます。ユーザー定義のワークロードやサービス・クラスを作成しないと、すべてのユーザー・アクティビティーはこれらのデフォルトのオブジェクトに関連付けられます。

この演習では、以下の 2 種類のモニター・フィーチャーについて説明します。

1. サービス・クラス中で実行されるすべてのアクティビティーに関する集約統計を収集する機能。集約アクティビティー統計により、サービス・クラス中の作業全体を低コストで表示できます。サービス・クラス中で実行されたアクティビティーの数や、これらのアクティビティーの平均存続時間などの情報が表示されます。
2. 個別のアクティビティーに関する情報をキャプチャーする機能。アクティビティー情報は、特定のアクティビティーのパフォーマンスや動作を調査する際に役立てることができます。アクティビティー情報には、ステートメント・テキストやコンパイル環境などが含まれます。集約アクティビティー統計よりアクティビティー情報の方が収集のコストが高いので、普通は特定のアクティビティーのサブセットをターゲットにします。

ステップ 1: イベント・モニターの作成と使用可能化

データベースに接続し、アクティビティーと統計に関するイベント・モニターを作成して使用可能にします。

```
CONNECT TO SAMPLE

CREATE EVENT MONITOR DB2ACTIVITIES FOR ACTIVITIES WRITE TO TABLE
CREATE EVENT MONITOR DB2STATISTICS FOR STATISTICS WRITE TO TABLE

SET EVENT MONITOR DB2ACTIVITIES STATE 1
SET EVENT MONITOR DB2STATISTICS STATE 1
```

ステップ 2: 個別のアクティビティーの収集

CREATE または ALTER WORKLOAD STATEMENT 上で COLLECT ACTIVITY DATA 節を使用して、個別のアクティビティーの収集を使用可能にします。ワークロードに関する COLLECT ACTIVITY DATA 節を指定すると、そのワークロードのオカレンスによってサブミットされたアクティビティーに関する情報が、アクティビティー完了時にアクティブな ACTIVITIES イベント・モニターに送信されます。COLLECT ACTIVITY DATA 節を使用すると、以下のいずれかのオプションを適用して、収集する必要がある情報量を指定できます。

- WITHOUT DETAILS: ステートメントとコンパイル環境を除くアクティビティー情報を収集します。
- WITH DETAILS: ステートメントとコンパイル環境を含むアクティビティー情報を収集します。
- WITH DETAILS AND VALUES: ステートメントとコンパイル環境を含むアクティビティー情報と、入力データ値を収集します。

この演習の場合、ステートメント・テキスト情報をキャプチャーできるように、WITH DETAILS 節を指定します。

```
ALTER WORKLOAD SYSDEFAULTUSERWORKLOAD
   COLLECT ACTIVITY DATA ON COORDINATOR WITH DETAILS
```

この例では、デフォルトのユーザー・ワークロードに関するアクティビティー・データが収集されます。その他のユーザー定義ワークロードは現在アクティブではないので、すべてのユーザー・アクティビティーに関する情報が収集される結果になります。実稼働環境では非常に高コストになります。特定のユーザー定義ワークロ

ードかサービス・クラスを使用して対象のアクティビティーを分離し、そのワークロードかサービス・クラスのみに COLLECT ACTIVITY DATA 節を適用する方が良い方法といえます。

追加情報: サービス・クラス、ワーク・クラス (ワーク・アクションを使用)、またはしきい値に対して COLLECT ACTIVITY DATA 節を指定することもできます。サービス・クラスに関する節を指定すると、そのサービス・クラス中で実行されるアクティビティーに関する情報が収集されます。ワーク・クラス (ワーク・アクションを使用) に関する節を指定すると、そのワーク・アクションが適用されるアクティビティーが収集されます。しきい値に関する節を指定すると、しきい値違反の場合にアクティビティー情報が収集されます。

ステップ 3: 集約アクティビティー統計の収集

COLLECT AGGREGATE ACTIVITY DATA 節を使用して、デフォルトのユーザー・サービス・クラスの下のデフォルト・サブクラスに関する集約アクティビティー統計の収集を使用可能にします。この節を指定すると、対応するサービス・クラスに関する集約統計 (例えば、アクティビティーの平均存続時間などの統計) がメモリー内に保守されます。統計は、サービス・サブクラス統計表関数を使用して表示したり、後で分析するために収集してアクティブな統計イベント・モニターに送信したりできます。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
   COLLECT AGGREGATE ACTIVITY DATA BASE
```

追加情報: デフォルトで収集される、すべての DB2 ワークロード管理オブジェクトに関する統計のセットがあります。COLLECT AGGREGATE ACTIVITY DATA 節を使用すると、アクティビティー存続時間ヒストグラムなどの、多数の追加のオプション統計を収集できます。

この例では、ユーザー定義サービス・クラスが作成されていないので、SYSDEFAULTUSERCLASS サービス・スーパー・クラスの下の SYSDEFAULTSUBCLASS サービス・サブクラス中ですべてのユーザー・アクティビティーが実行されます。したがって、すべてのユーザー・アクティビティーに関する情報が収集されます。

ステップ 4: 一部のアクティビティーの実行

一部のアクティビティーを実行します。その結果、統計が更新され、アクティビティーが収集されます。

```
db2 -o -tvf work1.db2
db2 -o -tvf work2.db2
```

アプリケーションを表すスクリプト (work1.db2 や work2.db2 など) によりデータベースから切断されるので、これらのスクリプトの実行後に再接続する必要があります。

ステップ 5: 統計の表示

WLM_GET_SERVICE_SUBCLASS_STATS 表関数を使用してサービス・クラス統計を表示できます。以下に例を示します。

```
CONNECT TO SAMPLE

SELECT VARCHAR(SERVICE_SUPERCLASS_NAME, 30) AS SUPERCLASS,
       VARCHAR(SERVICE_SUBCLASS_NAME, 30) AS SUBCLASS,
       LAST_RESET,
       COORD_ACT_COMPLETED_TOTAL,
       COORD_ACT_REJECTED_TOTAL,
       COORD_ACT_ABORTED_TOTAL,
       COORD_ACT_LIFETIME_AVG
FROM TABLE(SYSPROC.WLM_GET_SERVICE_SUBCLASS_STATS('SYSDEFAULTUSERCLASS',
       'SYSDEFAULTSUBCLASS', -1)) AS T
```

この照会からの出力は、以下のようになります。

```
SUPERCLASS                     SUBCLASS                       LAST_RESET
COORD_ACT_COMPLETED_TOTAL COORD_ACT_REJECTED_TOTAL COORD_ACT_ABORTED_TOTAL
COORD_ACT_LIFETIME_AVG
------------------------------ ------------------------------ -------------------------- --
---------------------- ---------------------- ---------------------- ------------------
------
SYSDEFAULTUSERCLASS            SYSDEFAULTSUBCLASS             2007-07-18-16.03.51.752190
74                         0                      0   +1.40288000000000E+002

  1 record(s) selected.
```

COORD_ACT_COMPLETED_TOTAL 列は、このサービス・クラス中で正常に完了したアクティビティーの数を示します。前回のリセット時刻は、このサービス・クラスに関する統計が前回リセットされた時刻を示します。

追加情報: COLLECT AGGREGATE ACTIVITY DATA 節を使用してサービス・クラスに関する集約アクティビティー統計を使用可能にしないと、WLM_GET_SERVICE_SUBCLASS_STATS 表関数によって報告される統計の一部は NULL になります。

ステップ 6: 統計のイベント・モニターへの送信

WLM_COLLECT_STATS ストアード・プロシージャーを使用して、すべての DB2 ワークロード管理オブジェクトの統計をアクティブな統計イベント・モニターに送信します。統計が収集されて統計イベント・モニターに送信される際に、値がリセットされます。

```
CALL SYSPROC.WLM_COLLECT_STATS()
```

追加情報: アクティブな統計イベント・モニターがない場合でも、WLM_COLLECT_STATS プロシージャーを使用して統計をリセットできますが、現行値は失われます。WLM_COLLECT_INT データベース構成パラメーターを使用して、ワークロード管理統計の収集を自動化できます。このパラメーターをゼロ以外の値に設定すると、(**wlm_collect_int** 分ごとに手動で WLM_COLLECT_STATS プロシージャーを呼び出すかのように) **wlm_collect_int** 分ごとにワークロード管理統計が自動的に収集されます。

ステップ 7: 統計の再表示

再度 WLM_GET_SERVICE_SUBCLASS_STATS 表関数を呼び出します。LAST_RESET タイム・スタンプが更新され、統計がリセットされていることに注意してください。

```
SELECT VARCHAR(SERVICE_SUPERCLASS_NAME, 30) AS SUPERCLASS,
       VARCHAR(SERVICE_SUBCLASS_NAME, 30) AS SUBCLASS,
       LAST_RESET,
```

```
       COORD_ACT_COMPLETED_TOTAL,
       COORD_ACT_REJECTED_TOTAL,
       COORD_ACT_ABORTED_TOTAL,
       COORD_ACT_LIFETIME_AVG
FROM TABLE(SYSPROC.WLM_GET_SERVICE_SUBCLASS_STATS('SYSDEFAULTUSERCLASS',
       'SYSDEFAULTSUBCLASS', -1)) AS T
```

出力は、以下のようになります。

```
SUPERCLASS                     SUBCLASS                       LAST_RESET
COORD_ACT_COMPLETED_TOTAL COORD_ACT_REJECTED_TOTAL
COORD_ACT_ABORTED_TOTAL COORD_ACT_LIFETIME_AVG
------------------------------ ------------------------------ ----------
--------------- ------------------------ ----------------------- ----
------------------ ------------------------
SYSDEFAULTUSERCLASS            SYSDEFAULTSUBCLASS             2007-07-18-
16.04.03.505818                         0                         0
0   +0.00000000000000E+000

  1 record(s) selected.
```

ステップ 8: 統計イベント・モニターによって収集されたサービス・クラス統計の表示

WLM_COLLECT_STATS プロシージャーは、サービス・クラス統計を統計イベント・モニターに送信します。以下のようなステートメントを使用して、イベント・モニターによって収集された統計を表示できます。

```
SELECT VARCHAR(SERVICE_SUPERCLASS_NAME, 30) AS SUPERCLASS,
       VARCHAR(SERVICE_SUBCLASS_NAME, 30) AS SUBCLASS,
       LAST_WLM_RESET,
       STATISTICS_TIMESTAMP,
       COORD_ACT_COMPLETED_TOTAL,
       COORD_ACT_REJECTED_TOTAL,
       COORD_ACT_ABORTED_TOTAL,
       COORD_ACT_LIFETIME_AVG
FROM SCSTATS_DB2STATISTICS
```

出力は、以下のようになります。

```
SUPERCLASS                     SUBCLASS
LAST_WLM_RESET             STATISTICS_TIMESTAMP
COORD_ACT_COMPLETED_TOTAL COORD_ACT_REJECTED_TOTAL
COORD_ACT_ABORTED_TOTAL COORD_ACT_LIFETIME_AVG
------------------------------ ------------------------------ -----------
--------------- ------------------------ ------------------------ ----
------------------- ---------------------- ----------------------
SYSDEFAULTSYSTEMCLASS          SYSDEFAULTSUBCLASS             2007-07-18-
16.03.46.333724 2007-07-18-16.04.03.505818                         0
0                       0                       -1
SYSDEFAULTMAINTENANCECLASS     SYSDEFAULTSUBCLASS             2007-07-18-
16.03.46.334301 2007-07-18-16.04.03.505818                         0
0                       0                       -1
SYSDEFAULTUSERCLASS            SYSDEFAULTSUBCLASS             2007-07-18-
16.03.51.752190 2007-07-18-16.04.03.505818                        75
0                       0                      136

  3 record(s) selected.
```

統計がイベント・モニターに送信されるたびに、DB2 ワークロード管理オブジェクトごとに統計レコードが作成されます。LAST_WLM_RESET と STATISTICS_TIMESTAMP の 2 つのタイム・スタンプに注意してください。LAST_WLM_RESET から STATISTICS_TIMESTAMP までの時間間隔は、そのレコードの統計が収集された時間間隔を示します。STATISTICS_TIMESTAMP は、統計

が収集された時点を示します。デフォルトのシステムおよび保守サービス・クラスに関する、コーディネーター上のアクティビティーの平均存続時間は -1 であることに注意してください。COLLECT AGGREGATE ACTIVITY DATA 節を使用して集約アクティビティー統計を使用可能にすると、サービス・クラスに関するアクティビティー平均存続時間統計のみ保守されます。

ステップ 9: アクティビティー情報の表示

ステップ 2 でデフォルト・ワークロードに対して COLLECT ACTIVITY DATA 節を指定したので、デフォルトのユーザー・ワークロードに関連した個別のアクティビティーすべてに関する情報もアクティビティー・イベント・モニターによって収集されています。以下のような照会を使用して、このアクティビティー情報を表示できます。

```
SELECT VARCHAR(A.APPL_NAME, 15) as APPL_NAME,
       VARCHAR(A.TPMON_CLIENT_APP, 20) AS CLIENT_APP_NAME,
       VARCHAR(A.APPL_ID, 30) as APPL_ID,
       A.ACTIVITY_ID,
       A.UOW_ID,
       VARCHAR(S.STMT_TEXT, 300) AS STMT_TEXT
FROM ACTIVITY_DB2ACTIVITIES AS A,
     ACTIVITYSTMT_DB2ACTIVITIES AS S
WHERE A.APPL_ID = S.APPL_ID AND
      A.ACTIVITY_ID = S.ACTIVITY_ID AND
      A.UOW_ID = S.UOW_ID
```

出力は、以下のようになります。

```
APPL_NAME       CLIENT_APP_NAME      APPL_ID
ACTIVITY_ID          UOW_ID      STMT_TEXT
--------------- ------------------- ----------------------------- -----
-------------- ----------- ---------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
-----------------------------------
db2bp           CLP wlmmonbasic.db2  *LOCAL.db2inst1.070718200344
1           8 ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER
SYSDEFAULTUSERCLASS COLLECT AGGREGATE ACTIVITY DATA BASE
db2bp           CLP work1.db2        *LOCAL.db2inst1.070718200352
1           1 values(current client_applname)
db2bp           CLP work1.db2        *LOCAL.db2inst1.070718200352
2           1 select * from org
db2bp           CLP work1.db2        *LOCAL.db2inst1.070718200352
3           1 select * from employee
db2bp           CLP work1.db2        *LOCAL.db2inst1.070718200352
4           1 select * from sales

...
```

切り捨ての警告 (SQL0445) が表示される可能性があることに注意してください。

CLP は、スクリプトを実行する際に、CURRENT CLIENT_APPLNAME 特殊レジスターを「CLP **script name**」に設定します。したがって、前の部分で示された照会から、各アクティビティーをサブミットしたスクリプトが分かります。

ステップ 10: 次の演習のためのリセット

アクティビティー・データまたは集約アクティビティー統計を収集できないように SYSDEFAULTUSERWORKLOAD ワークロードと SYSDEFAULTSUBCLASS サービ

ス・サブクラスを更新し、イベント・モニターを使用不可にして、アクティビティー表と統計表をクリアし、WLM_COLLECT_STATS() を呼び出して統計をリセットします。

```
ALTER WORKLOAD SYSDEFAULTUSERWORKLOAD COLLECT ACTIVITY DATA NONE

ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
      COLLECT AGGREGATE ACTIVITY DATA NONE

SET EVENT MONITOR DB2ACTIVITIES STATE 0
SET EVENT MONITOR DB2STATISTICS STATE 0

DELETE FROM ACTIVITY_DB2ACTIVITIES
DELETE FROM ACTIVITYSTMT_DB2ACTIVITIES
DELETE FROM SCSTATS_DB2STATISTICS
DELETE FROM WLSTATS_DB2STATISTICS

CALL WLM_COLLECT_STATS()
```

演習 2: サービス・クラスとワークロードを使用したアクティビティーの分離

この演習では、サービス・クラスを作成する方法と、ワークロードを使用してサービス・クラスにアクティビティーを送信する方法について説明します。また、いくつかの WLM モニター・フィーチャーを使用して、アクティビティーのマップ先のワークロードを判別し、サービス・クラス中およびワークロード下で実行されるアクティビティーに関する情報を取得する方法を示します。

時間の見積もり: 20 分から 25 分

サービス・クラスとは、データベース・アクティビティーに関するリソース制御の 1 次点のことです。サービス・クラスは、モニターにも役立ちます。例えば、特定のサービス・クラス中のアクティビティーに関する統計を収集して、そのサービス・クラスのパフォーマンス目標が満たされるかどうかを判別できます。デフォルトでは、データベースごとに 3 つのデフォルト・サービス・クラス (SYSDEFAULTSYSTEMCLASS、 SYSDEFAULTMAINTENANCECLASS、および SYSDEFAULTUSERCLASS) が作成されます。ユーザー定義のサービス・クラスを作成しないと、ユーザー・アクティビティーはデフォルトのユーザー・サービス・クラス (SYSDEFAULTUSERCLASS) の下で実行されます。

ワークロードとは、システム・ユーザー ID やセッション・ユーザー ID などの基準に基づいて 1 つ以上の作業単位をグループ化するエンティティーのことです。ワークロードは、作業をサービス・クラスに割り当てて、その作業を後で管理できるようにする手段を提供します。データベースごとにデフォルトのユーザー・ワークロード (SYSDEFAULTUSERWORKLOAD) とデフォルトの管理ワークロード (SYSDEFAULTADMWORKLOAD) が作成されます。ユーザー定義のワークロードを作成しないと、すべてのユーザー・アクティビティーはデフォルトのユーザー・ワークロードに関連付けられます。

この演習では、以下の 4 種類のフィーチャーについて説明されています。

- サービス・クラスを作成する方法
- ワークロードを作成する方法
- 基本ワークロード統計を調べる方法

- 個別のワークロードの下で実行されるアクティビティーに関するアクティビティー情報を収集する方法

ステップ 1: ユーザー定義サービス・クラスとワークロードがない場合にアクティビティーが実行される場所の調査

最初に、ユーザー定義サービス・クラスやワークロードがない場合に、アクティビティーが実行される場所を調べます。すべての DB2 アクティビティーは 1 つのワークロードに割り当てられ、1 つのサービス・クラス中で実行されます。ユーザー定義のサービス・クラスを作成しないと、アクティビティーはデフォルトのユーザー・サービス・クラス (SYSDEFAULTUSERCLASS) の下のデフォルトのサブクラス (SYSDEFAULTSUBCLASS) 中で実行され、ユーザー定義のワークロードを作成しないと、アクティビティーはデフォルトのユーザー・ワークロード (SYSDEFAULTUSERWORKLOAD) の下で実行されます。

work1.db2 および work2.db2 スクリプトを実行してから、WLM_GET_SERVICE_SUBCLASS_STATS を使用して SYSDEFAULTUSERCLASS の SYSDEFAULTSUBCLASS に関する統計を調べます。

```
db2 -o -tvf work1.db2
db2 -o -tvf work2.db2

CONNECT TO SAMPLE

SELECT VARCHAR( SERVICE_SUPERCLASS_NAME, 30) SUPERCLASS,
       VARCHAR( SERVICE_SUBCLASS_NAME, 30) SUBCLASS,
       COORD_ACT_COMPLETED_TOTAL
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('','',-1)) AS T
```

以下のような出力が表示されます。

```
SUPERCLASS                     SUBCLASS                       COORD_ACT_COMPLETE
D_TOTAL
------------------------------ ------------------------------ ------------------
-------
SYSDEFAULTSYSTEMCLASS          SYSDEFAULTSUBCLASS
      0
SYSDEFAULTMAINTENANCECLASS     SYSDEFAULTSUBCLASS
      0
SYSDEFAULTUSERCLASS            SYSDEFAULTSUBCLASS
     75

  3 record(s) selected.
```

すべてのアクティビティーが SYSDEFAULTUSERCLASS サービス・スーパー・クラス中で実行されることに注意してください。

追加情報: その他に SYSDEFAULTSYSTEMCLASS および SYSDEFAULTMAINTENANCECLASS という 2 つのサービス・クラスもあります。これらのサービス・クラスは内部保守およびシステム・レベルのタスク用に使用されます。ユーザー・アクティビティーはこれらのサービス・クラス中で実行されません。DB2 データ・サーバーが内部アクティビティーを発行した場合、これらのサービス・クラス中にもゼロ以外のアクティビティー・カウントがある可能性があります。

WLM_GET_WORKLOAD_STATS 表関数を使用して、ワークロード統計を表示し、アプリケーションに関連付けられているワークロードを判別します。

```
SELECT SUBSTR(WORKLOAD_NAME, 1, 22) AS WL_DEF_NAME,
       WLO_COMPLETED_TOTAL,
       CONCURRENT_WLO_ACT_TOP FROM
       TABLE(WLM_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2))
     AS WLSTATS
```

出力は、以下のようになります。

```
WL_DEF_NAME           WLO_COMPLETED_TOTAL  CONCURRENT_WLO_ACT_TOP
--------------------- -------------------- ----------------------
SYSDEFAULTUSERWORKLOAD                    3                      5
SYSDEFAULTADMWORKLOAD                     0                      0

  2 record(s) selected.
```

両方のスクリプト (work1.db2 および work2.db2) で完了したワークロード・オカレンスが 1 つあり、前述のコマンドの実行に使用された接続に関するワークロード・オカレンスが 1 つあることに注意してください。

ステップ 2: サービス・クラスとワークロードの作成

サービス・クラスを作成してから、ワークロードを作成し、work1.db2 スクリプトから実行されるすべてのアクティビティーを新しく作成されたサービス・クラスにマップできるようにします。CLP がスクリプトを実行する際に、CURRENT CLIENT_APPLNAME 特殊レジスター値が「CLP **script name**」に設定されます。

```
CREATE SERVICE CLASS work1_sc

CREATE WORKLOAD work1_wl CURRENT CLIENT_APPLNAME('CLP work1.db2')
       SERVICE CLASS work1_sc
```

追加情報: ワークロードやサービス・クラスの作成時に指定できる属性が多数あります。例えば、ワークロードの作成時に、アプリケーション名やセッション・ユーザーなどに基づいて接続を識別できます。詳しくは、CREATE WORKLOAD と CREATE SERVICE CLASS の資料を参照してください。

ステップ 3: ワークロードに対する使用権の付与

ワークロードに対する使用権を付与します (ACCESSCTRL または SECADM 権限が必要)。

```
GRANT USAGE ON WORKLOAD work1_wl TO PUBLIC
```

追加情報: セッション・ユーザーがそのワークロードに対する USAGE 特権を持っている場合のみ、接続をワークロードに関連付けることができます。この操作は、優先順位の高いサービス・クラスで作業を試行中に、ユーザーがアプリケーションの接続属性を変更できないようにするために必要です。プログラムで (例えば、sqleseti API を使用して) 変更できる接続属性もあります。この演習では、単純に USAGE 特権を PUBLIC に付与します。実際のシステムでは、さらに細かく区別する場合もあるでしょう。このサンプルは DBADM として実行されるので、このステップはすべてスキップしてもかまいません。

ステップ 4: 統計のリセット

WLM_COLLECT_STATS 関数を使用して統計をリセットし、収集された統計をクリアします。

```
CALL SYSPROC.WLM_COLLECT_STATS()
```

ステップ 5: 一部のアクティビティーの実行

work1.db2 スクリプトと work2.db2 スクリプトを両方とも実行します。

```
db2 -o -tvf work1.db2
db2 -o -tvf work2.db2
```

ステップ 6: ワークロードとサービス・クラスの統計の表示

WLM_GET_WORKLOAD_STATS 表関数を使用して、ワークロード統計を表示し、アプリケーションに関連付けられているワークロードを判別します。

```
CONNECT TO SAMPLE

SELECT SUBSTR(WORKLOAD_NAME, 1, 22) AS WL_DEF_NAME,
       WLO_COMPLETED_TOTAL,
       CONCURRENT_WLO_ACT_TOP
       FROM TABLE(WLM_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2))
     AS WLSTATS
```

出力は、以下のようになります。

```
WL_DEF_NAME            WLO_COMPLETED_TOTAL  CONCURRENT_WLO_ACT_TOP
---------------------- -------------------- ----------------------
WORK1_WL                                 1                      5
SYSDEFAULTUSERWORKLOAD                   1                      5
SYSDEFAULTADMWORKLOAD                    0                      0
```

work1.db2 スクリプトである WORK1_WL の下で 1 つのワークロード・オカレンスが完了したことに注意してください。work2.db2 スクリプトである SYSDEFAULTUSERWORKLOAD の下で 1 つのワークロード・オカレンスが完了しています。

SYSDEFAULTUSER WORKLOAD で完了した 2 つ目のワークロード・オカレンスが表示されることがありますが、これは WLM_COLLECT_STATS プロシージャーの呼び出しに使用された接続です。WLM_COLLECT_STATS は非同期プロシージャーで、統計が実際に収集される前に完了する可能性があるので、組み込まれる可能性があります。

さらに、WLM_GET_SERVICE_SUBCLASS_STATS 表関数を使用して、新しいワークロードを作成した結果として、実行されているアクティビティーがどのサービス・クラスに属しているか表示することもできます。

```
SELECT VARCHAR( SERVICE_SUPERCLASS_NAME, 30) SUPERCLASS,
       VARCHAR( SERVICE_SUBCLASS_NAME, 23) SUBCLASS,
       COORD_ACT_COMPLETED_TOTAL COORDACTCOMP
       FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('','',-1)) AS T
```

結果は以下のようになります。

```
SUPERCLASS                     SUBCLASS                COORDACTCOMP
------------------------------ ----------------------- --------------------
SYSDEFAULTSYSTEMCLASS          SYSDEFAULTSUBCLASS                     0
SYSDEFAULTMAINTENANCECLASS     SYSDEFAULTSUBCLASS                     0
SYSDEFAULTUSERCLASS            SYSDEFAULTSUBCLASS                    37
WORK1_SC                       SYSDEFAULTSUBCLASS                    37
```

WORK1_WL ワークロード・マッピングのために WORK1_SC の下で完了したアクティビティーに注意してください。

ステップ 7: 別のサービス・クラスとワークロードの作成

2 つ目のサービス・クラスを作成してから、ワークロードを作成し、work2.db2 アプリケーションから実行するすべてのアクティビティーを新しく作成されたサービス・クラスにマップできるようにします。さらに、一部のアクティビティー・データが収集されるように、ワークロードをセットアップします。この例の場合、追加の詳細や値を指定せず、単にアクティビティー・データを収集します。

```
CREATE SERVICE CLASS work2_sc

CREATE WORKLOAD work2_wl
              CURRENT CLIENT_APPLNAME('CLP work2.db2')
              SERVICE CLASS work2_sc
              COLLECT ACTIVITY DATA
```

追加情報: ワークロードに関する COLLECT ACTIVITY DATA 節を指定すると、そのワークロードのオカレンスによってサブミットされたアクティビティーに関する情報が、アクティビティー完了時にアクティブな ACTIVITIES イベント・モニターに送信されます。COLLECT ACTIVITY DATA 節を使用すると、以下のいずれかのオプションを適用して、収集する必要がある情報量を指定できます。

- WITHOUT DETAILS: ステートメントとコンパイル環境を除くアクティビティー情報を収集します (デフォルト)。
- WITH DETAILS: ステートメントとコンパイル環境を含むアクティビティー情報を収集します。
- WITH DETAILS AND VALUES: ステートメントとコンパイル環境を含むアクティビティー情報と、入力データ値を収集します。

ステップ 8: アクティビティー・イベント・モニターの使用可能化

アクティビティーに関するイベント・モニターを使用可能にします。

アクティビティー・イベント・モニターは、演習 1 で作成しました。

```
SET EVENT MONITOR DB2ACTIVITIES STATE 1
```

ステップ 9: 統計のリセットと一部のアクティビティーの実行

WLM_COLLECT_STATS ストアード・プロシージャーを使用して、統計を再度リセットし、再度 work1.db2 および work2.db2 スクリプトを稼働します。

```
CALL SYSPROC.WLM_COLLECT_STATS()

db2 -o -tvf work1.db2
db2 -o -tvf work2.db2
```

ステップ 10: ワークロードとサービス・クラスの統計の表示

再度 WLM_GET_WORKLOAD_STATS 表関数を使用して、アプリケーションに関連付けられているワークロードを判別します。

```
CONNECT TO SAMPLE

SELECT SUBSTR(WORKLOAD_NAME, 1, 22) AS WL_DEF_NAME,
       WLO_COMPLETED_TOTAL,
       CONCURRENT_WLO_ACT_TOP
FROM TABLE(WLM_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2))
     AS WLSTATS
```

出力は、以下のようになります。

```
WL_DEF_NAME            WLO_COMPLETED_TOTAL  CONCURRENT_WLO_ACT_TOP
---------------------- -------------------- ----------------------
WORK1_WL                                 1                      5
WORK2_WL                                 1                      5
SYSDEFAULTUSERWORKLOAD                   0                      0
SYSDEFAULTADMWORKLOAD                    0                      0
```

今回は、両方のワークロード定義とも、スクリプトごとに 1 つずつワークロード・オカレンスを実行していることに注意してください。

統計が収集される前に、WLM_COLLECT_STATS プロシージャーに対する呼び出しがサブミットされたワークロード・オカレンスが閉じるかどうかに応じて、SYSDEFAULTUSERWORKLOAD で完了したワークロード・オカレンスが表示されることも表示されないこともあります。

再度 WLM_GET_SERVICE_SUBCLASS_STATS を使用して、新しいワークロードを作成した結果、実行されているアクティビティーがどのサービス・クラスに属しているか表示します。

```
SELECT VARCHAR( SERVICE_SUPERCLASS_NAME, 30) SUPERCLASS,
       VARCHAR( SERVICE_SUBCLASS_NAME, 23) SUBCLASS,
       COORD_ACT_COMPLETED_TOTAL COORDACTCOMP
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('','',-1)) AS T
```

結果は、以下のようになります。

```
SUPERCLASS                     SUBCLASS                COORDACTCOMP
------------------------------ ----------------------- --------------------
SYSDEFAULTSYSTEMCLASS          SYSDEFAULTSUBCLASS                         0
SYSDEFAULTMAINTENANCECLASS     SYSDEFAULTSUBCLASS                         0
SYSDEFAULTUSERCLASS            SYSDEFAULTSUBCLASS                         1
WORK1_SC                       SYSDEFAULTSUBCLASS                        37
WORK2_SC                       SYSDEFAULTSUBCLASS                        37
```

今回は、WORK2_WL マッピングのために、サービス・スーパー・クラス work2_sc の下で一部のアクティビティーが実行されることに注意してください。SYSDEFAULTUSERCLASS の下のアクティビティーの 1 つに、以前に WLM_GET_WORKLOAD_STATS 上で実行した照会があります。

ステップ 11: 収集されたアクティビティー・データの表示

実行したアクティビティーに関する情報について、アクティビティー表を照会します。work2_wl ワークロード定義のみ COLLECT ACTIVITY DATA 属性が指定されているので、work2.db2 スクリプトからのアクティビティーのみ収集されることに注意してください。

```
SELECT SUBSTR(WORKLOADNAME, 1, 20) WL_DEF_NAME,
       SUBSTR(APPL_NAME, 1, 20) APPL_NAME,
       SUBSTR(ACTIVITY_TYPE, 1, 10) ACT_TYPE
FROM SYSCAT.WORKLOADS, ACTIVITY_DB2ACTIVITIES
WHERE WORKLOADID = WORKLOAD_ID
```

結果は、以下のようになります。

```
WL_DEF_NAME          APPL_NAME            ACT_TYPE
-------------------- -------------------- ----------
WORK2_WL             db2bp                READ_DML
WORK2_WL             db2bp                READ_DML
WORK2_WL             db2bp                READ_DML
```

```
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               WRITE_DML
WORK2_WL            db2bp               WRITE_DML
WORK2_WL            db2bp               WRITE_DML
WORK2_WL            db2bp               WRITE_DML
WORK2_WL            db2bp               WRITE_DML
WORK2_WL            db2bp               WRITE_DML
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               OTHER
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               OTHER
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               LOAD
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               CALL
WORK2_WL            db2bp               READ_DML
WORK2_WL            db2bp               CALL
WORK2_WL            db2bp               CALL
WORK2_WL            db2bp               CALL
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               DDL
WORK2_WL            db2bp               DDL
:
:
```

ステップ 12: サービス・クラスへのリソースの割り当て

これらの 2 つのスクリプトによって発行されるアクティビティーを別個のサービス・クラスに分離したので、サービス・クラスにリソースを割り当てたり、これらのサービス・クラス中で実行されるアクティビティーをモニターしたりできます。2、3 の例を挙げます。スクリプト `work2.db2` によって実行される作業が、スクリプト `work1.db2` によって実行される作業より重要な場合は、以下のようなステートメントを使用して、WORK2_SC サービス・クラス中で実行するエージェントの優先順位を高くすることもできます。

UNIX 稼働環境では、以下のようにします (負の値で指定される優先順位の方が高くなります)。

```
ALTER SERVICE CLASS WORK2_SC AGENT PRIORITY -6
```

Windows 稼働環境では、以下のようにします (正の値で指定される優先順位の方が高くなります)。

```
ALTER SERVICE CLASS WORK2_SC AGENT PRIORITY 6
```

WORK2_SC サービス・クラス中で実行する個別のアクティビティーすべてに関する詳細をキャプチャーする場合は、以下のステートメントを使用して、このサービス・クラスに関するアクティビティーの収集を使用可能にすることもできます。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER WORK2_SC
   COLLECT ACTIVITY DATA ON COORDINATOR WITH DETAILS
```

ステップ 13: 次の演習のためのリセット

アクティビティー・データを収集できないようにワークロード work2_wl を更新し、イベント・モニターを使用不可にして、イベント・モニター表をクリーンアップし、WLM_COLLECT_STATS() を呼び出して統計をリセットします。

```
ALTER WORKLOAD work2_wl
      COLLECT ACTIVITY DATA NONE

SET EVENT MONITOR DB2ACTIVITIES STATE 0

DELETE from ACTIVITY_DB2ACTIVITIES

CALL WLM_COLLECT_STATS()
```

演習 3: しきい値を使用した不良アクティビティーの制御と、しきい値違反モニターの使用

この演習では、特定のリソースの使用量による制限を確立して、しきい値を使用してリソースの誤用を検出する方法、またはシステムの過負荷を初期のうちに検出する方法について説明します。

時間の見積もり: 15 分から 20 分

しきい値に違反すると、特定のアクションが起動するようにできます。サポートされるアクションは以下のとおりです。

- STOP EXECUTION: しきい値に違反する原因となったアクティビティーの処理を停止します。
- CONTINUE: 処理を続行します。
- しきい値に違反したアクティビティーに関する情報を収集します。このアクションは、CONTINUE アクションか STOP EXECUTION アクションと組み合わせて指定できます。

しきい値に違反したアクティビティーが停止するか、それとも実行を続行できるかにかかわらず、しきい値に違反するたびに違反のレコードがアクティブな THRESHOLD VIOLATIONS イベント・モニター (事前に定義されていることが前提) に書き込まれます。レコードには、違反のあったしきい値、違反の時刻、しきい値アクションなどの情報が含まれます。

この演習では、しきい値を使用して、不良アクティビティーを検出したり、システム上で実行できないようにしたりして、システム・リソースを使い切る方法について説明します。不良アクティビティーとは、予想外に大量のリソースを使用するアクティビティーのことです。例えば、異常に長時間実行する照会や、予想外に大きな結果セットを戻す照会があります。

ステップ 1: しきい値違反イベント・モニターの作成

しきい値違反情報のキャプチャーに使用する表書き込みイベント・モニターを作成して使用可能にし、演習 1 で作成したアクティビティー・イベント・モニターを使用可能にします。

```
CREATE EVENT MONITOR threvio FOR THRESHOLD VIOLATIONS WRITE TO TABLE
          THRESHOLDVIOLATIONS(IN userspace1),
          CONTROL(IN userspace1)

SET EVENT MONITOR threvio STATE 1

SET EVENT MONITOR db2activities STATE 1
```

ステップ 2: ワークロードの作成

ワークロードを作成し、workth.db2 スクリプトから実行するすべてのアクティビティーを work1_sc サービス・クラスにマップできるようにします。

work1_sc サービス・クラスは演習 2 で作成したので既存です。

```
CREATE WORKLOAD workth_wl
          CURRENT CLIENT_APPLNAME('CLP workth.db2')
          SERVICE CLASS work1_sc
```

ステップ 3: しきい値の作成

2 つのしきい値を作成します。その 1 つ (th_estcost) は ESTIMATEDSQLCOST のしきい値で、もう 1 つ (th_sqlrows) は SQLROWSRETURNED のしきい値です。アクティビティーを制御するサービス・クラス (この演習では work1_sc サービス・クラス) にこれらの値を適用します。

th_estcost しきい値は、work1_sc サービス・クラス中で実行しているアクティビティーのオプティマイザー見積もりコスト (timeron 単位) の上限 (10000 timeron) を指定します。見積もりコストが 10000 timeron を超える照会を work1_sc サービス・クラス中で実行しようとすると、このしきい値に違反し、照会を実行できません。

th_sqlrows しきい値は、work1_sc サービス・クラス中で実行しているアクティビティーがデータ・サーバーから最大 30 行を戻せることを指定します。照会が 30 より多い行を戻そうとすると、このしきい値に違反するので、30 行のみクライアントに戻され、照会は停止します。さらに、しきい値の違反の原因となったアクティビティーに関するデータが収集されます。

どちらの場合も、アクティビティーがしきい値に違反すると、しきい値違反レコードがステップ 1 の定義どおりに THRESHOLD VIOLATIONS イベント・モニターに書き込まれ、アクティビティーの実行が (STOP EXECUTION アクションのために) 停止します。アクティビティーをサブミットしたアプリケーションは、SQL4712N エラーを受信します。

```
CREATE THRESHOLD th_estcost
    FOR SERVICE CLASS work1_sc ACTIVITIES
    ENFORCEMENT DATABASE
    WHEN ESTIMATEDSQLCOST > 10000
    STOP EXECUTION

CREATE THRESHOLD th_sqlrows
    FOR SERVICE CLASS work1_sc ACTIVITIES
    ENFORCEMENT DATABASE
    WHEN SQLROWSRETURNED > 30
    COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
    STOP EXECUTION
```

追加情報: しきい値は予測的か反応的かのどちらかになります。

- 予測的しきい値: 予測的しきい値境界は、アクティビティーが実行を開始する前に検査されます。予測的しきい値が違反するかどうか検査するために、データ・サーバーは照会コンパイラーから使用量の見積もりを入手します。この例の場合、th_estcost しきい値が予測的しきい値です。
- 反応的しきい値: 反応的しきい値境界は、アクティビティーが実行されている間に検査されます。制御されるリソースの概算の実行時使用量は、反応的しきい値境界を評価するのに使用されます。実行時使用量の推定量は連続して入手するのではなく、むしろトラッキングされる作業の存続時間中の、選択された定義済みのチェックポイントで入手します。この例の場合、th_sqlrows が反応的しきい値です。

ステップ 4: 一部のアクティビティーの実行

一部のアクティビティーを実行します。その一部は前述のステップで定義したしきい値の上限に違反します。

```
db2 -o -tvf workth.db2
```

前述のステップで定義したしきい値に違反するステートメントが、SQL4712N/SQLSTATE 5U026 のエラーで失敗することに注意してください。

ステップ 5: しきい値違反イベント・モニターの表示

すべてのしきい値違反に関する情報が、THRESHOLD VIOLATIONS イベント・モニターによって収集されます。以下の例のように、しきい値違反モニター表に対して正規の SQL ステートメントを発行して、しきい値違反情報を照会できます。

```
CONNECT TO SAMPLE

SELECT  APPL_ID,
       UOW_ID,
       ACTIVITY_ID,
       COORD_PARTITION_NUM AS COORDPART,
       THRESHOLD_PREDICATE,
       THRESHOLD_ACTION,
       TIME_OF_VIOLATION
FROM THRESHOLDVIOLATIONS_THREVIO
ORDER BY THRESHOLD_ACTION, THRESHOLD_PREDICATE, TIME_OF_VIOLATION
```

出力は、以下のようになります。

```
APPL_ID                                                         UOW_ID
ACTIVITY_ID          COORDPART   THRESHOLD_PREDICATE
                                       THRESHOLD_ACTION TIME_OF_VIOLATION
------------------------------------------------------------------------ ----
--------------- ----------- ------------------------
-------------------------------------------------------- ------------------------
--
*LOCAL.DB2.070821150008                                                      11
1           0 EstimatedSQLCost
                                        Stop            2007-08-21-
11.00.11.000000
*LOCAL.DB2.070821150008                                                      10
1           0 SQLRowsReturned
                                      Stop              2007-08-21-
11.00.10.000000

  2 record(s) selected.
```

ステップ 6: しきい値に違反したアクティビティーに関する情報の表示

COLLECT 節で定義されているしきい値に違反するアクティビティーに関するアクティビティー情報が収集されます。以下の照会を使用して、しきい値に違反するアクティビティーに関する詳細情報を表示します。

```
SELECT VARCHAR(A.APPL_NAME, 15) as APPL_NAME,
       VARCHAR(A.TPMON_CLIENT_APP, 20) AS CLIENT_APP_NAME,
       A.ACTIVITY_ID,
       A.ACTIVITY_TYPE,
       A.WORKLOAD_ID,
       T.THRESHOLD_PREDICATE,
       A.QUERY_CARD_ESTIMATE,
       T.THRESHOLD_MAXVALUE,
       T.TIME_OF_VIOLATION,
       VARCHAR(AS.STMT_TEXT, 100) AS STMT_TEXT
FROM  THRESHOLDVIOLATIONS_THREVIO AS T,
       ACTIVITY_DB2ACTIVITIES AS A,
       ACTIVITYSTMT_DB2ACTIVITIES AS AS
WHERE T.APPL_ID = A.APPL_ID AND
       T.UOW_ID = A.UOW_ID AND
            T.ACTIVITY_ID = A.ACTIVITY_ID AND
            A.APPL_ID = AS.APPL_ID AND
            A.ACTIVITY_ID = AS.ACTIVITY_ID AND
            A.UOW_ID = AS.UOW_ID
```

出力は、以下のようになります。

```
APPL_NAME       CLIENT_APP_NAME      ACTIVITY_ID          ACTIVITY_TYPE
                                          WORKLOAD_ID THRESHOLD_PREDICATE
                                    QUERY_CARD_ESTIMATE  THRESHOLD_MAXVALUE
  TIME_OF_VIOLATION          STMT_TEXT

--------------- ------------------- ------------------- ----------------------
---------------------------------------- ----------- ------------------------
------------------------------------- ------------------- ------------------
- ------------------------- --------------------------------------------------
-------------------------------------------------
db2bp           CLP workth.db2                          3 READ_DML
                                                      3 SQLRowsReturned
                                                           41                   3
0 2007-08-31-09.01.16.000000 SELECT * FROM SALES
```

th_estcost (EstimatedSqlCost) しきい値に違反したアクティビティーが表示されないことに注意してください。その理由は、このしきい値は COLLECT ACTIVITY DATA 節を指定していないので、このアクティビティーに関するアクティビティー・データが収集されなかったからです。

ステップ 7: 次の演習のためのリセット

使用可能にしたイベント・モニターを使用不可にします。また、作成した th_estcost しきい値と th_sqlrows しきい値を使用不可にしてドロップします。

```
SET EVENT MONITOR threvio STATE 0
SET EVENT MONITOR db2activities STATE 0

ALTER THRESHOLD th_estcost DISABLE
DROP THRESHOLD th_estcost

ALTER THRESHOLD th_sqlrows DISABLE
DROP THRESHOLD th_sqlrows
```

さらに、アクティビティー・イベント・モニター表としきい値違反表をクリーンアップします。

```
DELETE from ACTIVITY_DB2ACTIVITIES
DELETE from ACTIVITYSTMT_DB2ACTIVITIES
DELETE from THRESHOLDVIOLATIONS_THREVIO

CALL WLM_COLLECT_STATS()
```

演習 4: アクティビティー・タイプごとのアクティビティーの区別

この演習では、ワーク・アクション・セットを使用して、特定のタイプのアクティビティーすべてに関する情報を収集する方法、特定のタイプのアクティビティーすべてにしきい値を適用する方法、特定のタイプのアクティビティーを特定のサービス・サブクラスにマップして分離する方法について説明します。

時間の見積もり: 25 分から 30 分

ワーク・アクション・セットは、アクティビティーをサブミットした人ではなくアクティビティーが行うことに基づいてアクティビティーにアクションを適用するのに使用します (ワークロードと併用して行います)。

以下のいずれかにアクションを適用できます。

- 特定のタイプのデータベース・アクティビティーすべて (データベース・ワーク・アクション・セットを使用)。
- 特定のサービス・クラス中の特定のタイプのアクティビティーのみ (サービス・クラス・ワーク・アクション・セットを使用)。

この演習では両方の方式とも実行します。

追加情報: この演習で取り上げられていない特定のタイプのアクティビティーに関する統計の収集などの、他のアクションも適用できます。

ステップ 1: ワーク・クラス・セットの作成

最初に、対象となる特定のタイプのアクティビティーを表すワーク・クラスを含むワーク・クラス・セットを作成します。このワーク・クラス・セットは、選択したタイプのアクティビティーに関するアクションを実行するワーク・クラス・セットと組み合わせて使用します。以下の例では、可能なすべてのタイプのワーク・クラスを含むワーク・クラス・セットを作成しますが、1 つのアクティビティー・タイプのみを対象とする場合は、その 1 つのワーク・クラスのみを含むようにワーク・クラス・セットを作成することもできます。

```
CREATE WORK CLASS SET all_class_types
    (WORK CLASS read_wc WORK TYPE READ,
     WORK CLASS write_wc WORK TYPE WRITE,
     WORK CLASS ddl_wc WORK TYPE DDL,
     WORK CLASS call_wc WORK TYPE CALL,
     WORK CLASS load_wc WORK TYPE LOAD,
     WORK CLASS all_wc WORK TYPE ALL POSITION LAST)
```

ステップ 2: アクティビティー・イベント・モニターの使用可能化

演習 1 で作成したアクティビティーに関するイベント・モニターを使用可能にします。

```
SET EVENT MONITOR DB2ACTIVITIES STATE 1
```

ステップ 3: データベース・ワーク・アクション・セットの作成

特定のタイプのアクティビティーすべてに関する特定のアクション (しきい値の適用やアクティビティー情報の収集など) を実行する場合は、データベース・ワーク・アクション・セットを使用します。

分離するアクティビティーのタイプを表す特定のワーク・クラスに関するワーク・アクションを含むワーク・アクション・セットをデータベース・レベルで作成します。この例の場合、システム上で実行するすべての DDL、READ、および LOAD アクティビティーに関するアクティビティー・データを収集することができます。また、大量の読み取りアクティビティーの実行を停止することもできます。この演習の場合、大量の読み取りアクティビティーは、10000 より大きな見積もりコスト (timerons 単位) のある SELECT ステートメントです。

```
CREATE WORK ACTION SET db_was FOR DATABASE
   USING WORK CLASS SET  all_class_types
   (WORK ACTION collect_load_wa ON WORK CLASS load_wc
       COLLECT ACTIVITY DATA WITH DETAILS AND VALUES,
    WORK ACTION collect_ddl_wa ON WORK CLASS ddl_wc
       COLLECT ACTIVITY DATA WITH DETAILS AND VALUES,
    WORK ACTION collect_read_wa ON WORK CLASS read_wc
       COLLECT ACTIVITY DATA WITH DETAILS AND VALUES,
    WORK ACTION stop_large_read_wa on WORK CLASS read_wc
       WHEN ESTIMATEDSQLCOST > 10000 STOP EXECUTION )
```

ステップ 4: アクティビティーの実行およびワーク・アクション・セット統計の表示

work1.db2 および work3.db2 スクリプトを実行します。

```
db2 -o -tvf work1.db2
db2 -o -tvf work3.db2
```

WLM_GET_WORK_ACTION_SET_STATS 表関数を使用して、メモリー内のワーク・アクション・セット統計にアクセスし、特定のアクティビティー・タイプが実行された回数を取得できます。適用可能なワーク・アクションのあるワーク・クラスは load_wc、read_wc、および ddl_wc ワーク・クラスのみなので、以下の照会を実行すると、これらのワーク・クラスのみ表示されることに注意してください。その他のアクティビティーはすべて「*」の下でカウントされます。

```
CONNECT TO SAMPLE

SELECT SUBSTR(WORK_ACTION_SET_NAME, 1, 12) AS WORK_ACTION_SET_NAME,
         SUBSTR(WORK_CLASS_NAME, 1, 12) AS WORK_CLASS_NAME,
         LAST_RESET,
         SUBSTR(CHAR(ACT_TOTAL), 1, 10) AS TOTAL_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS('', -2)) AS WASSTATS
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME
```

出力は、以下のようになります。

```
WORK_ACTION_SET_NAME WORK_CLASS_NAME  LAST_RESET                TOTAL_ACTS
-------------------- ---------------- -------------------------- ----------
DB_WAS               *                2007-08-15-19.02.47.305556 12
DB_WAS               DDL_WC           2007-08-15-19.02.47.305556 12
```

```
DB_WAS               LOAD_WC          2007-08-15-19.02.47.305556 1
DB_WAS               READ_WC          2007-08-15-19.02.47.305556 13

  4 record(s) selected.
```

ステップ 5: 収集されたアクティビティー・データの表示

ステップ 3 で ddl_wc、read_wc、および load_wc ワーク・クラスに適用する COLLECT ACTIVITY DATA ワーク・アクションを指定したので、個別の DDL、READ、および LOAD アクティビティーすべてに関する情報もアクティビティー・イベント・モニターによって収集されています。以下の 2 つの例は、このアクティビティー情報を表示する方法を示しています。

以下のようなステートメントを使用してアクティビティー・イベント・モニター表を照会することで、アクティビティーに関する基本情報の一部を取得できます。

```
SELECT ACTIVITY_ID,
               SUBSTR(ACTIVITY_TYPE, 1, 8) AS ACTIVITY_TYPE,
               VARCHAR(APPL_ID, 30) AS APPL_ID,
               VARCHAR(APPL_NAME, 10) AS APPL_NAME
FROM ACTIVITY_DB2ACTIVITIES
```

出力は、以下のようになります。

```
ACTIVITY_ID          ACTIVITY_TYPE APPL_ID                        APPL_NAME
-------------------- ------------- ------------------------------ ----------
                   1 READ_DML      *LOCAL.karenam.070815192410    db2bp
                   1 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   2 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   3 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   4 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   1 DDL           *LOCAL.karenam.070815192418    db2bp
                   2 DDL           *LOCAL.karenam.070815192418    db2bp
                   3 DDL           *LOCAL.karenam.070815192418    db2bp
                   2 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   1 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   2 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   3 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   4 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   6 LOAD          *LOCAL.karenam.070815192418    db2bp
                   1 DDL           *LOCAL.karenam.070815192418    db2bp
                   1 DDL           *LOCAL.karenam.070815192418    db2bp
                   2 DDL           *LOCAL.karenam.070815192418    db2bp
                   3 DDL           *LOCAL.karenam.070815192418    db2bp
                   4 DDL           *LOCAL.karenam.070815192418    db2bp
                   5 READ_DML      *LOCAL.karenam.070815192418    db2bp
                  10 READ_DML      *LOCAL.karenam.070815192418    db2bp
                   1 DDL           *LOCAL.karenam.070815192418    db2bp
                   2 DDL           *LOCAL.karenam.070815192418    db2bp
                   3 DDL           *LOCAL.karenam.070815192418    db2bp
                   4 DDL           *LOCAL.karenam.070815192418    db2bp
                   1 READ_DML      *LOCAL.karenam.070815192426    db2bp

  26 record(s) selected.
```

アクティビティー・テキストや、アクティビティーの実行元のサービス・クラスなどの各アクティビティーに関する追加情報を入手するには、以下のような照会を実行できます。

```
SELECT VARCHAR(A.APPL_NAME, 15) as APPL_NAME,
               VARCHAR(A.TPMON_CLIENT_APP, 20) AS CLIENT_APP_NAME,
               VARCHAR(A.APPL_ID, 30) as APPL_ID,
               VARCHAR(A.SERVICE_SUPERCLASS_NAME, 20) as SUPER_CLASS,
               VARCHAR(A.SERVICE_SUBCLASS_NAME, 20) as SUB_CLASS,
```

```
                SQLCODE,
                VARCHAR(S.STMT_TEXT, 300) AS STMT_TEXT
FROM ACTIVITY_DB2ACTIVITIES AS A, ACTIVITYSTMT_DB2ACTIVITIES AS S
WHERE A.APPL_ID = S.APPL_ID AND
               A.ACTIVITY_ID = S.ACTIVITY_ID AND
               A.UOW_ID = S.UOW_ID
```

出力は、以下のようになります。

```
APPL_NAME       CLIENT_APP_NAME      APPL_ID
SUPER_CLASS
         SUB_CLASS            SQLCODE     STMT_TEXT
--------------- -------------------- ----------------------------- ---------
---
-------- -------------------- ----------- ---------------------------------
---
------------------------------------------------------------------------------
---
------------------------------------------------------------------------------
---
------------------------------------------------------------------------------
---
----------------------
db2bp           CLP wasdbsc.db2      *LOCAL.karenam.070815192410
SYSDEFAULTUS
ERCLASS  SYSDEFAULTSUBCLASS             0 SELECT DISTINCT CURRENT SQLID FROM
SYS
IBM.SYSTABLES

db2bp           CLP work1.db2        *LOCAL.karenam.070815192418
SYSDEFAULTUS
ERCLASS  SYSDEFAULTSUBCLASS             0 values(current client_applname)

      :
      :
db2bp           CLP work1.db2        *LOCAL.karenam.070815192418
SYSDEFAULTUS
ERCLASS  SYSDEFAULTSUBCLASS             0 drop procedure stp2


db2bp           CLP work3.db2        *LOCAL.karenam.070815192426
SYSDEFAULTUS
ERCLASS  SYSDEFAULTSUBCLASS         -4712 select count(*) from syscat.tables,
sy
scat.tables, syscat.tables, syscat.tables, syscat.tables, syscat.tables

      :
      :
```

アクティビティーの 1 つに SQLCODE -4712 があることに注意してください。これは、しきい値違反のためにアクティビティーの実行が停止したことを示しています。stop_large_read_wa ワーク・アクションの定義済みのしきい値により、見積もりコストが 10000 より大きな SELECT ステートメントは実行されません。

追加情報: アクティビティー・ステートメント・イベント・モニター表 (activitystmt_db2activities 表) 中にはロード・アクティビティー (カーソルからのロードは含まれていない) の項目がありません。これは、work1.db2 スクリプトによって実行される単一のロード・アクティビティーに関するレコードが前述の直前の照会で表示された出力中にない理由に関する説明になります。この理由は、ロード・アクティビティーが SQL ステートメントでないためです。カーソル・アクティビティーからのロードの場合、カーソル自体が別個のアクティビティーなので、アクティビティー・ステートメント・イベント・モニター表にカーソル・ステート

メントの項目があります。アクティビティー・イベント・モニター表 (activity_db2activities) の中に、すべてのロード・アクティビティーの項目があります。

ステップ 6: ワーク・アクションの使用不可化

サービス・クラス・ワーク・アクション・セットに移る前に、データベース・ワーク・アクション・セットをドロップします。

```
DROP WORK ACTION SET db_was
```

追加情報: 並行性のしきい値をドロップする前に、最初にこのしきい値を使用不可にしなければなりません。この実習では並行性のしきい値を表すワーク・アクションはありませんが、仮にあった場合は、このワーク・アクションを使用不可にしないとこのしきい値を使用不可にできません。THRESHOLD SQL ステートメントを使用してワーク・アクションしきい値を操作できません。WORK ACTION SET SQL ステートメントを使用する場合のみ操作できます。以後ワーク・アクション・セットをドロップするには、その前に並行性のしきい値を表すワーク・アクションのみ使用不可にする必要があります。この実習の場合、並行性のしきい値を表すワーク・アクションがないので、ワーク・アクション・セットをドロップする前にワーク・アクションを使用不可にする必要はありません。

しきい値などの特定のアクションを、サービス・スーパー・クラス中で実行している特定のタイプのアクティビティーすべてに適用する場合は、サービス・クラス・ワーク・アクション・セットの使用を考慮する必要があります。マッピング・ワーク・アクションを作成して、特定のタイプのアクティビティーを特定のサービス・サブクラスにマップしてから、このサービス・サブクラスにしきい値を適用できます。以下のステップでは、サービス・クラス・ワーク・アクション・セットの使用法について説明します。

ステップ 7: サービス・クラスとワークロードの作成

演習 2 のステップ 2 で作成した work1_sc サービス・スーパー・クラスの下にサービス・サブクラスを作成します。

サービス・スーパー・クラス work1_sc は、ワークロードによってアクティビティーがマップされるサービス・クラスです。サービス・サブクラス work1_sc_read は、ワーク・アクションによって読み取りアクティビティーがマップされるサービス・クラスです。

```
CREATE SERVICE CLASS work1_sc_read UNDER work1_sc
```

ワークロードを作成して、work3.db2 スクリプトによってサブミットされるすべてのアクティビティーを work1_sc サービス・スーパー・クラスにマップできるようにします。work1.db2 からのアクティビティーが、以前の実習のいずれかから work1_sc にすでにマップされていることに注意してください。

```
CREATE WORKLOAD work3_wl CURRENT CLIENT_APPLNAME('CLP work3.db2')
   SERVICE CLASS work1_sc
```

ステップ 8: サービス・クラス・ワーク・アクション・セットの作成

分離するアクティビティーのタイプを表す特定のワーク・クラスに適用するワーク・アクションを含むワーク・アクション・セットをサービス・クラス・レベルで

作成します。この例の場合、work1_sc サービス・クラスの下で実行するすべての DDL、読み取り、およびロード・アクティビティーに関するアクティビティー・データを収集することができます。また、読み取りアクティビティーを別個のサービス・サブクラスにマップして別々に扱えるようにすることもできます。この演習では、しきい値をサービス・サブクラスに適用して大規模な SELECT ステートメントの実行を停止します。

```
CREATE WORK ACTION SET sc_was FOR SERVICE CLASS work1_sc
  USING WORK CLASS SET all_class_types (
    WORK ACTION collect_load_wa ON WORK CLASS load_wc
      COLLECT ACTIVITY DATA ON ALL MEMBERS WITH DETAILS AND VALUES,
    WORK ACTION collect_ddl_wa ON WORK CLASS ddl_wc
      COLLECT ACTIVITY DATA ON ALL MEMBERS WITH DETAILS AND VALUES,
   WORK ACTION collect_read_wa ON WORK CLASS read_wc
      COLLECT ACTIVITY DATA ON ALL MEMBERS WITH DETAILS AND VALUES,
   WORK ACTION map_read_wa on WORK CLASS read_wc
          MAP ACTIVITY TO work1_sc_read)
```

ステップ 9: サービス・クラスしきい値の作成

大規模な SELECT ステートメントを実行しないようにする stop_large_read_wa ワーク・アクションと同様の効果を得るには、ESTIMATEDSQLCOST しきい値を作成して work1_sc_read サービス・サブクラスに適用します。

```
CREATE THRESHOLD stop_large_activities FOR SERVICE CLASS work1_sc_read
             UNDER work1_sc
             ACTIVITIES ENFORCEMENT DATABASE
             WHEN ESTIMATEDSQLCOST >10000 STOP EXECUTION
```

ステップ 10: アクティビティー表のクリア、統計のリセット、およびアクティビティーの実行

スクリプトを再実行する前に、アクティビティー表をすべてクリアして、新たに開始できるようにします。次に、wlm_collect_stats() ストアード・プロシージャーを呼び出して、統計をリセットします。

```
DELETE FROM activity_db2activities
DELETE FROM activitystmt_db2activities
DELETE FROM activityvals_db2activities

CALL wlm_collect_stats()
```

この時点で `work1.db2` および `work3.db2` スクリプトを 1 回実行します。

```
db2 -o -tvf work1.db2
db2 -o -tvf work3.db2
```

しきい値を超過させたアクティビティーに関する SQL04712 エラーに注意してください。

ステップ 11: ワーク・アクション・セット統計の表示

WLM_GET_WORK_ACTION_SET_STATS 表関数を使用して、メモリー内のワーク・アクション・セット統計にアクセスし、特定のアクティビティー・タイプを実行した回数を取得します。適用するワーク・アクションのあるワーク・クラスは load_wc、ddl_wc、および read_wc ワーク・クラスの 3 つのみなので、以下の照会を実行すると、これらのワーク・クラスのみ表示されることに注意してください。その他のアクティビティーは最終的にすべて「*」の下でカウントされます。

```
CONNECT TO SAMPLE

SELECT SUBSTR(WORK_ACTION_SET_NAME, 1, 12) AS WORK_ACTION_SET_NAME,
               SUBSTR(CHAR(MEMBER), 1, 4) AS MEMB,
               SUBSTR(WORK_CLASS_NAME, 1, 12) AS WORK_CLASS_NAME, LAST_RESET,
               SUBSTR(CHAR(ACT_TOTAL), 1, 10) AS TOTAL_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS('', -2)) AS WASSTATS
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME, MEMB
```

今回は、出力は以下のようになります。

```
WORK_ACTION_SET_NAME MEMB WORK_CLASS_NAME LAST_RESET                TOTAL_ACTS
-------------------- ---- -------------- -------------------------- ----------
SC_WAS               0    *              2007-08-15-19.02.54.597999 12
SC_WAS               0    DDL_WC         2007-08-15-19.02.54.597999 12
SC_WAS               0    LOAD_WC        2007-08-15-19.02.54.597999 1
SC_WAS               0    READ_WC        2007-08-15-19.02.54.597999 12

  4 record(s) selected.
```

ステップ 12: 収集されたアクティビティー・データの表示

次に、再度アクティビティー表を照会し、個別のアクティビティーに関する情報を取得します。アクティビティーの実行元のサービス・サブクラスに注意してください。

```
SELECT VARCHAR(A.APPL_NAME, 15) as APPL_NAME,
                VARCHAR(A.TPMON_CLIENT_APP, 20) AS CLIENT_APP_NAME,
                VARCHAR(A.APPL_ID, 30) as APPL_ID,
                VARCHAR(A.SERVICE_SUPERCLASS_NAME, 20) as SUPER_CLASS,
                VARCHAR(A.SERVICE_SUBCLASS_NAME, 20) as SUB_CLASS,
                SQLCODE,
                VARCHAR(S.STMT_TEXT, 300) AS STMT_TEXT
FROM ACTIVITY_DB2ACTIVITIES AS A, ACTIVITYSTMT_DB2ACTIVITIES AS S
WHERE A.APPL_ID = S.APPL_ID AND
               A.ACTIVITY_ID = S.ACTIVITY_ID AND
               A.UOW_ID = S.UOW_ID
```

出力は、以下のようになります。

```
APPL_NAME       CLIENT_APP_NAME      APPL_ID                        SUPER_CLASS
         SUB_CLASS            SQLCODE     STMT_TEXT
--------------- -------------------- ------------------------------ ------------
-------- -------------------- ----------- ------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
----------------------

db2bp           CLP work1.db2        *LOCAL.karenam.070815195555    WORK1_SC
         WORK1_SC_READ                    0 values(current client_applname)

db2bp           CLP work1.db2        *LOCAL.karenam.070815195555    WORK1_SC
         WORK1_SC_READ                    0 select * from org
:
:
db2bp           CLP work1.db2        *LOCAL.karenam.070815195555    WORK1_SC
         SYSDEFAULTSUBCLASS               0 drop procedure stp2

db2bp           CLP work3.db2        *LOCAL.karenam.070815195600    WORK1_SC
         WORK1_SC_READ                -4712 select count(*) from syscat.tables, sy
scat.tables, syscat.tables, syscat.tables, syscat.tables, syscat.tables
```

アクティビティーの 1 つに SQLCODE -4712 があることにも注意してください。今回の理由は、この SELECT ステートメントの見積もりコストが大きすぎるので、

ステップ 9 で作成した stop_large_activities サービス・クラスのしきい値に違反したためです。読み取りアクティビティーはすべて work1_sc_read サービス・サブクラスの下で実行されていることにも注意してください。

ステップ 13: 次の演習のためのリセット

イベント・モニターを使用不可にして、サービス・クラスしきい値をドロップし、サービス・クラス・ワーク・アクション・セットをドロップします。

```
SET EVENT MONITOR DB2ACTIVITIES STATE 0

DROP THRESHOLD STOP_LARGE_ACTIVITIES
ALTER WORK ACTION SET SC_WAS
 ALTER WORK ACTION COLLECT_LOAD_WA DISABLE
 ALTER WORK ACTION COLLECT_DDL_WA DISABLE
 ALTER WORK ACTION COLLECT_READ_WA DISABLE
 ALTER WORK ACTION MAP_READ_WA DISABLE;
DROP WORK ACTION SET SC_WAS
```

スクリプトを再実行する前に、アクティビティー表をすべてクリアして、新たに開始できるようにします。

```
DELETE FROM activity_db2activities
DELETE FROM activitystmt_db2activities
DELETE from activityvals_db2activities
```

作成したワークロードをすべて使用不可にして、すべてのアクティビティーがデフォルトのユーザー・ワークロードの下で実行され、デフォルトのサービス・スーパー・クラスにマップされるようにします。

```
ALTER WORKLOAD work1_wl DISABLE
ALTER WORKLOAD work2_wl DISABLE
ALTER WORKLOAD work3_wl DISABLE
ALTER WORKLOAD workth_wl DISABLE
```

wlm_collect_stats() ストアード・プロシージャーを呼び出して、統計をリセットします。

```
CALL WLM_COLLECT_STATS()
```

演習 5: サービス・クラスに関するヒストグラムの使用

この演習では、サービス・クラスに対する COLLECT AGGREGATE ACTIVITY DATA BASE オプションを使用して、コーディネーター・アクティビティーの存続時間、コーディネーター・アクティビティーの実行時間、およびコーディネーター・アクティビティーのキュー時間のヒストグラムを作成する方法について説明します。

時間の見積もり: 25 分から 30 分

これらの 3 つのヒストグラムは、標準偏差の計算に使用したり異常値を示したりすることができるので、単にシステム上で実行するアクティビティーの平均存続時間、実行時間、またはキュー時間以上のことを知るのに役立ちます。ヒストグラムについて詳しくは、292 ページの『ワークロード管理のヒストグラム』を参照してください。

ヒストグラムには、統計イベント・モニターによってアクセスします。この演習では、演習 1 のステップ 1 で作成した統計イベント・モニターを再利用します。

追加情報: 統計イベント・モニターは、表書き込みイベント・モニターで、論理データ・グループが含まれます。初めはすべてのイベント・モニターにある制御論理データ・グループで、その後に統計イベント・モニターのタイプに固有の論理データ・グループになります。固有の論理データ・グループとは、以下のとおりです。

- ヒストグラム情報に関する histogrambin
- しきい値キュー統計に関する qstats
- scstats サービス・クラス統計
- ワーク・クラス統計に関する wcstats
- ワークロード統計に関する wlstats

ステップ 1: ヒストグラム統計の表示用のビューの作成

複数のビューを作成して、HISTOGRAMBIN_DB2STATISTICS 表の照会を容易にします。最初のビューは、使用可能なすべてのヒストグラム・タイプをリストします。この演習では、存続時間、実行時間、およびキュー時間の 3 つの基本タイプのみ報告します。

```
CREATE VIEW HISTOGRAMTYPES AS
  SELECT DISTINCT SUBSTR(HISTOGRAM_TYPE,1,24) AS HISTOGRAM_TYPE
  FROM HISTOGRAMBIN_DB2STATISTICS
```

2 番目のビューは、ヒストグラムの収集対象のサービス・クラスの検出を容易にします。HISTOGRAMBIN_DB2STATISTICS 表は、サービス・クラス ID を使用して、ヒストグラムの収集対象のサービス・クラスを識別します。この表と SERVICECLASSES カタログ表を結合すると、サービス・クラス情報を、サービス・クラス ID ではなくサービス・スーパー・クラス名およびサービス・サブクラス名と共に表示できます。

```
CREATE VIEW HISTOGRAMSERVICECLASSES AS
  SELECT DISTINCT SUBSTR(HISTOGRAM_TYPE,1,24) AS HISTOGRAM_TYPE,
         SUBSTR(PARENTSERVICECLASSNAME,1,24) AS SERVICE_SUPERCLASS,
         SUBSTR(SERVICECLASSNAME,1,24) AS SERVICE_SUBCLASS
  FROM HISTOGRAMBIN_DB2STATISTICS AS H,
       SYSCAT.SERVICECLASSES AS S
  WHERE H.SERVICE_CLASS_ID = S.SERVICECLASSID
```

3 番目のビューは、特定のサービス・クラスに関する特定のタイプのヒストグラムが収集された時間をすべてリストします。histogramserviceclasses ビューと同様に、このビューも HISTOGRAMBIN_DB2STATISTICS 表を SERVICECLASSES カタログ表と結合します。違う点は、このビューには列の 1 つとして STATISTICS_TIMESTAMP 列が組み込まれていることです。

```
CREATE VIEW HISTOGRAMTIMES AS
  SELECT DISTINCT SUBSTR(HISTOGRAM_TYPE,1,24) AS HISTOGRAM_TYPE,
         SUBSTR(PARENTSERVICECLASSNAME,1,24) AS SERVICE_SUPERCLASS,
         SUBSTR(SERVICECLASSNAME,1,24) AS SERVICE_SUBCLASS,
         STATISTICS_TIMESTAMP AS TIMESTAMP
  FROM HISTOGRAMBIN_DB2STATISTICS AS H,
       SYSCAT.SERVICECLASSES AS S
  WHERE H.SERVICE_CLASS_ID = S.SERVICECLASSID
```

4 番目と最後のビューは、ヒストグラム自体を表示するのに使用します。また、ヒストグラムを処理する際に頻繁に行う必要がある、長期にわたるヒストグラムの集約についても説明します。このビューは、各ビンの最上位と、各ビンに対してカウントされたアクティビティーの数を表示します。この演習の 3 つのヒストグラムの

場合、BIN_TOP フィールドはアクティビティーの存続時間、実行時間、またはキュー時間のミリ秒数を測定します。存続時間ヒストグラムの場合に、BIN_TOP が約 3000 ミリ秒で、直前のビンの BIN_TOP が 2000 ミリ秒で、NUMBER_IN_BIN が 10 のときには、10 個のアクティビティーの存続時間が 2 秒から 3 秒までの間であったことを示しています。

```
CREATE VIEW HISTOGRAMS(HISTOGRAM_TYPE,
                       SERVICE_SUPERCLASS,
                       SERVICE_SUBCLASS,
                       BIN_TOP,
                       NUMBER_IN_BIN) AS
  SELECT DISTINCT SUBSTR(HISTOGRAM_TYPE,1,24) AS HISTOGRAM_TYPE,
         SUBSTR(PARENTSERVICECLASSNAME,1,24) AS SERVICE_SUPERCLASS,
         SUBSTR(SERVICECLASSNAME,1,24) AS SERVICE_SUBCLASS,
         TOP AS BIN_TOP,
         SUM(NUMBER_IN_BIN) AS NUMBER_IN_BIN
  FROM HISTOGRAMBIN_DB2STATISTICS AS H,
       SYSCAT.SERVICECLASSES AS S
  WHERE H.SERVICE_CLASS_ID = S.SERVICECLASSID
  GROUP BY HISTOGRAM_TYPE, PARENTSERVICECLASSNAME, SERVICECLASSNAME, TOP
```

ステップ 2: ヒストグラムの収集をオンにする

サービス・サブクラスに関する基本的な集約アクティビティー・データ収集オプションを使用可能にすると、このサブクラスに関するアクティビティーの存続時間、キュー時間、および実行時間のヒストグラムが収集されます。COLLECT AGGREGATE ACTIVITY DATA 節を使用して、デフォルトのユーザー・スーパー・クラスの下のデフォルト・サブクラスに関する基本的な集約アクティビティー・データの収集を使用可能にします。

前の演習の終わりにユーザー定義のワークロードをすべて使用不可にしたので、デフォルトのユーザー・サービス・クラス中ですべてのアクティビティーが実行されることに注意してください。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS
  UNDER SYSDEFAULTUSERCLASS
  COLLECT AGGREGATE ACTIVITY DATA BASE
```

ステップ 3: 統計イベント・モニターのアクティブ化

以前に作成した統計イベント・モニターをアクティブにして、集約データが収集されたら必ず受信できるようにします。

```
SET EVENT MONITOR DB2STATISTICS STATE 1
```

ステップ 4: アクティビティーの実行および統計の統計イベント・モニターへの送信

この時点で、一部のアクティビティーを実行できます。アクティビティーの終了後に、WLM_COLLECT_STATS ストアード・プロシージャーを呼び出して、(デフォルトのユーザー・サービス・クラスに関するアクティビティーの存続時間、実行時間、およびキュー時間のヒストグラムを含む) 統計をアクティブな統計イベント・モニターに送信します。これらのヒストグラムには、集約アクティビティー統計を使用可能にした以降にデフォルトのユーザー・サービス・クラス中で実行されたすべてのアクティビティーに関するデータが含まれます。このストアード・プロシージャーを呼び出すと、統計のリセットも行われます。長期にわたるデータベース・

アクティビティーの変更を示すために、3 つの収集間隔が作成されます。最初の間隔では、work1.db2 および work2.db2 の 2 つのスクリプトを実行してから、統計を収集してリセットします。

```
db2 -o -tvf work1.db2
db2 -o -tvf work2.db2

CONNECT TO SAMPLE

CALL WLM_COLLECT_STATS()
```

2 番目の間隔では、work1.db2 スクリプトのみ 1 回実行してから、統計を収集してリセットします。

```
db2 -o -tvf work1.db2

CONNECT TO SAMPLE

CALL WLM_COLLECT_STATS()
```

3 番目の間隔では、work1.db2 を 2 回実行し、work2.db2 スクリプトを 1 回実行してから、統計を収集してリセットします。

```
db2 -o -tvf work1.db2
db2 -o -tvf work2.db2
db2 -o -tvf work1.db2

CONNECT TO SAMPLE

CALL WLM_COLLECT_STATS()
```

このようにデータを定期的に収集すると、長期にわたってシステム上の作業の変更内容を監視できます。

追加情報: 手動操作でデータを定期的に収集する必要はありません。WLM_COLLECT_INT データベース構成パラメーターを使用すると、間隔を分数で設定でき、その間隔の後に統計の収集とリセットが自動的に行われます。

ステップ 5: 統計を表示するための照会ビュー

統計が収集されたので、以前に作成したビューを使用して統計を表示できます。HISTOGRAMTYPES ビューは単に使用可能なヒストグラムのタイプを戻します。

```
SELECT * FROM HISTOGRAMTYPES

HISTOGRAM_TYPE
------------------------
CoordActExecTime
CoordActLifetime
CoordActQueueTime

  3 record(s) selected.
```

サービス・クラスの変更時に BASE オプションを使用したので、存続時間、実行時間、およびキュー時間の 3 つのヒストグラムがあります。HISTOGRAMSERVICECLASSES ビューを使用すると、ヒストグラムの収集対象のサービス・クラスを表示できます。以下の例は、出力を、CoordActLifetime ヒストグラムの出力のみに制限します。デフォルト・ユーザー・サービス・クラスのデフ

ォルト・サブクラスのみに関する集約アクティビティーの収集をオンにしたので、HISTOGRAMSERVICECLASSES ビューからの選択時にこのクラスのみ表示されます。

```
SELECT * FROM HISTOGRAMSERVICECLASSES
  WHERE HISTOGRAM_TYPE = 'CoordActLifetime'
  ORDER BY SERVICE_SUPERCLASS, SERVICE_SUBCLASS

HISTOGRAM_TYPE           SERVICE_SUPERCLASS       SERVICE_SUBCLASS
------------------------ ------------------------ ------------------------
-
CoordActLifetime         SYSDEFAULTUSERCLASS      SYSDEFAULTSUBCLASS

  1 record(s) selected.
```

HISTOGRAMTIMES ビューは、ヒストグラムが収集された回数を表示します。WLM_COLLECT_STATS プロシージャーは 3 回実行されたので、下記の存続時間ヒストグラムには 3 つのタイム・スタンプがあります。

```
SELECT * FROM HISTOGRAMTIMES
  WHERE HISTOGRAM_TYPE = 'CoordActLifetime'
    AND SERVICE_SUPERCLASS = 'SYSDEFAULTUSERCLASS'
    AND SERVICE_SUBCLASS = 'SYSDEFAULTSUBCLASS'
  ORDER BY TIMESTAMP

HISTOGRAM_TYPE   SERVICE_SUPERCLASS  SERVICE_SUBCLASS   TIMESTAMP
---------------- ------------------- ------------------ -----------------
---------
CoordActLifetime SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2007-08-05-
20.44.51.519380
CoordActLifetime SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2007-08-05-
21.04.27.131281
CoordActLifetime SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2007-08-05-
21.08.27.474168

  3 record(s) selected.
```

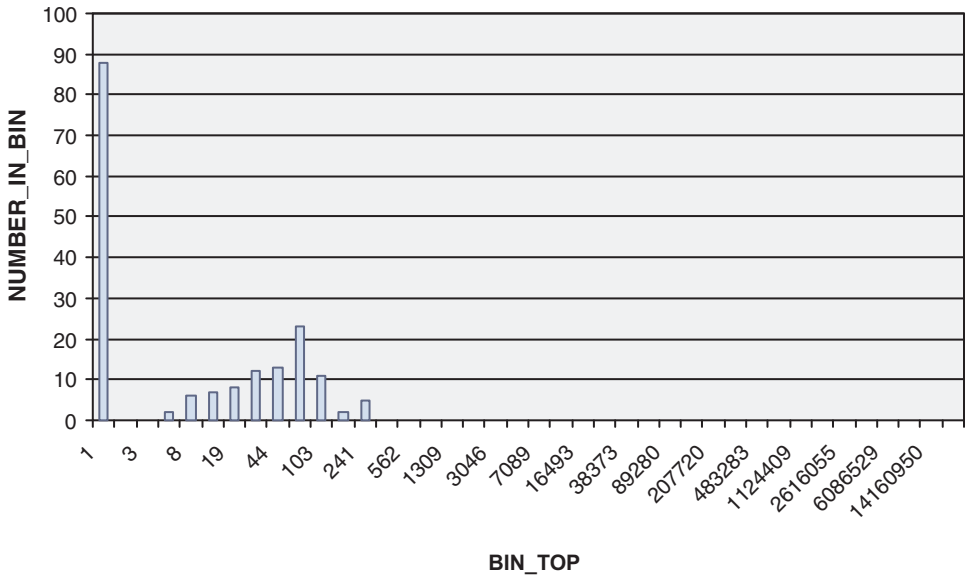
最後のビューの HISTOGRAMS は、ヒストグラム自体を表示します。HISTOGRAMTIMES ビューは各収集間隔を独自の行としてリストしますが、このビューはそれとは違って複数の間隔の間でヒストグラム・データを集約し、特定のサービス・クラスに関する特定のタイプのヒストグラムを 1 つ作成します。

```
SELECT BIN_TOP, NUMBER_IN_BIN FROM HISTOGRAMS
  WHERE HISTOGRAM_TYPE = 'CoordActLifetime'
    AND SERVICE_SUPERCLASS = 'SYSDEFAULTUSERCLASS'
    AND SERVICE_SUBCLASS = 'SYSDEFAULTSUBCLASS'
  ORDER BY BIN_TOP

BIN_TOP              NUMBER_IN_BIN
-------------------- --------------------
                  -1                    0
                   1                   88
                   2                    0
                   3                    0
                   5                    2
                   8                    6
                  12                    7
                  19                    8
                  29                   12
                  44                   13
                  68                   23
                 103                   11
                 158                    2
                 241                    5
                 369                    0
```

```
          562                   0
          858                   0
         1309                   0
         1997                   0
         3046                   0
         4647                   0
         7089                   0
        10813                   0
        16493                   0
        25157                   0
        38373                   0
        58532                   0
        89280                   0
       136181                   0
       207720                   0
       316840                   0
       483283                   0
       737162                   0
      1124409                   0
      1715085                   0
      2616055                   0
      3990325                   0
      6086529                   0
      9283913                   0
     14160950                   0
     21600000                   0

  41 record(s) selected.
```

次に、このヒストグラムからの出力をグラフ作成ツールへの入力として使用し、グラフを生成できます。以下の図は、Gruff Graphs と呼ばれる Ruby Graphing Library を使用して作成したグラフを示しています。

SYSDEFAULTUSERCLASS の存続時間ヒストグラム (CoordActLifetime):



アクティビティーの存続時間はシステムのパフォーマンスに応じて異なるので、このビューの照会を実行して作成される出力は、前の部分の存続時間ヒストグラム・グラフで示されているものとまったく同じにはならないはずです。前述の出力では、41 個のビンがあり、最大のビンはすべて空です。最上部には、BIN_TOP が -1 のビンがあります。このビンは、存続時間が長すぎてヒストグラムに収まらいす

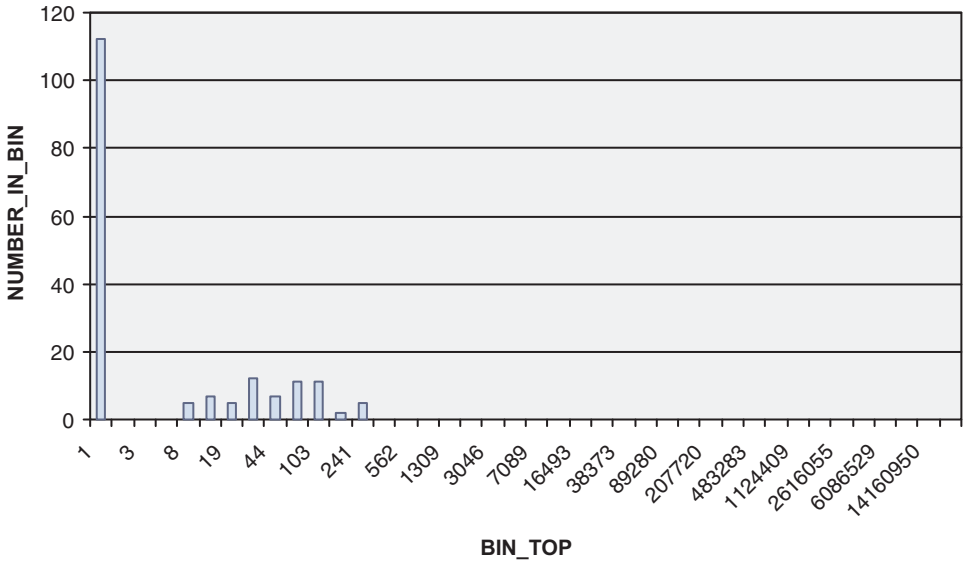
べてのアクティビティーを表します。BIN_TOP が -1 の際に、NUMBER_OF_BIN がゼロより大きい場合は、おそらくヒストグラムのビンの上限値を大きくする必要があることを示しています。前述の出力では、NUMBER_IN_BIN は 0 なので、このような変更を加える必要はありません。BIN_TOP が 1 の場合、ビン中で多数のアクティビティー (この演習では 88) がカウントされています。これはビンの下限で、88 個のアクティビティーの存続時間が 0 ミリ秒から 1 ミリ秒までの間であることを示しています。ヒストグラムから抜き出せる別の情報の部分としては、対応する NUMBER_IN_BIN がゼロ以外の BIN_TOP の最大値が 241 なので、このヒストグラム中に収集されたワークロード中のアクティビティーの最大存続時間が 158 ミリ秒から 241 ミリ秒の間だったということがあります。
SCSTATS_DB2STATISTICS 表の COORD_ACT_LIFETIME_TOP 列には、存続時間が最長のアクティビティーの、さらに正確な存続時間の測定が示されています。

CoordActLifetime の代わりに CoordActExecTime の histogram_type を使用しても同じ照会を繰り返すことができます。実行時間ヒストグラムは同じと予期されますが、存続時間ヒストグラムとは同一にはなりません。違う理由は、キューイングがない場合でも、実行時間には初期化の時間やカーソル・アイドル時間が含まれておらず、存続時間には含まれているからです。

```
SELECT BIN_TOP, NUMBER_IN_BIN FROM HISTOGRAMS
  WHERE HISTOGRAM_TYPE = 'CoordActExecTime'
    AND SERVICE_SUPERCLASS = 'SYSDEFAULTUSERCLASS'
    AND SERVICE_SUBCLASS = 'SYSDEFAULTSUBCLASS'
  ORDER BY BIN_TOP

BIN_TOP              NUMBER_IN_BIN
-------------------- --------------------
                  -1                    0
                   1                  112
                   2                    0
                   3                    0
                   5                    0
                   8                    5
                  12                    7
                  19                    5
                  29                   12
                  44                    7
                  68                   11
                 103                   11
                 158                    2
                 241                    5
                 369                    0
                 562                    0
                 858                    0
                1309                    0
                1997                    0
                3046                    0
                4647                    0
                7089                    0
               10813                    0
               16493                    0
               25157                    0
               38373                    0
               58532                    0
               89280                    0
              136181                    0
              207720                    0
              316840                    0
              483283                    0
              737162                    0
             1124409                    0
```

```
           1715085                    0
           2616055                    0
           3990325                    0
           6086529                    0
           9283913                    0
          14160950                    0
          21600000                    0

  41 record(s) selected.
```

SYSDEFAULTUSERCLASS の実行時間ヒストグラム (CoordActExecTime):



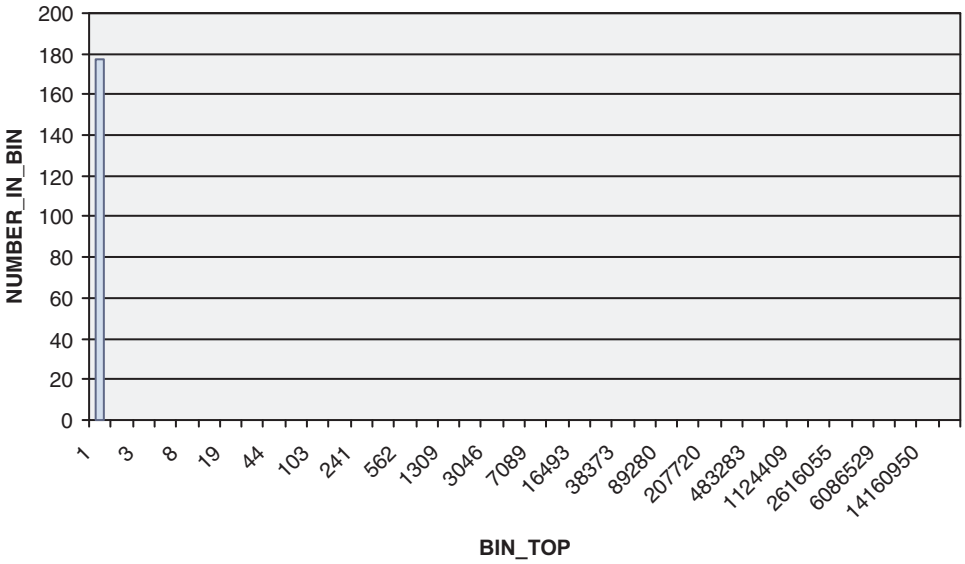
今回も、最初のビンでは多数のアクティビティーがカウントされ、アクティビティーの最長実行時間は最大で 241 ミリ秒です。

最後に、HISTOGRAMS ビューは CoordActQueueTime ヒストグラムを表示するのに使用します。この演習ではキューイングしきい値を作成したり使用可能にしたりしていないので、キューイングがないためにこのヒストグラムが最も単純になります。

```
SELECT BIN_TOP, NUMBER_IN_BIN FROM HISTOGRAMS
  WHERE HISTOGRAM_TYPE = 'CoordActQueueTime'
    AND SERVICE_SUPERCLASS = 'SYSDEFAULTUSERCLASS'
    AND SERVICE_SUBCLASS = 'SYSDEFAULTSUBCLASS'
  ORDER BY BIN_TOP

BIN_TOP              NUMBER_IN_BIN
-------------------- --------------------
                  -1                    0
                   1                  177
                   2                    0
                   3                    0
                   5                    0
                   8                    0
                  12                    0
                  19                    0
                  29                    0
                  44                    0
                  68                    0
                 103                    0
                 158                    0
                 241                    0
                 369                    0
```

```
           562                   0
           858                   0
          1309                   0
          1997                   0
          3046                   0
          4647                   0
          7089                   0
         10813                   0
         16493                   0
         25157                   0
         38373                   0
         58532                   0
         89280                   0
        136181                   0
        207720                   0
        316840                   0
        483283                   0
        737162                   0
       1124409                   0
       1715085                   0
       2616055                   0
       3990325                   0
       6086529                   0
       9283913                   0
      14160950                   0
      21600000                   0

  41 record(s) selected.
```

SYSDEFAULTUSERCLASS のキュー時間ヒストグラム (CoordActQueueTime):



すべてのアクティビティーがキューイングに費やす時間は 0 ミリ秒なので、すべてのアクティビティーが 0 ミリ秒から 1 ミリ秒までのビン中でカウントされています。

最後の複数の照会は、ビンに分割されたアクティビティーの存続時間、実行時間、およびキュー時間を表示していますが、複数の間隔にわたって集約しています。以下の照会は、同じ情報を別の観点から表示します。ヒストグラムの代わりに平均を表示し、間隔を結合する代わりに各間隔を個別に表示します。また、完了したアクティビティー数のカウントを報告します。このカウントは、各間隔で完了したアク

ティビティーの数を示します。この照会は、HISTOGRAMBIN_DB2STATISTICS 表ではなく SCSTATS_DB2STATISTICS 表を使用します。

```
SELECT STATISTICS_TIMESTAMP,
       COORD_ACT_LIFETIME_AVG AS LIFETIMEAVG,
       COORD_ACT_EXEC_TIME_AVG AS EXECTIMEAVG,
       COORD_ACT_QUEUE_TIME_AVG AS QUEUETIMEAVG,
       COORD_ACT_COMPLETED_TOTAL AS COMPLETED_TOTAL
FROM SCSTATS_DB2STATISTICS
WHERE SERVICE_SUPERCLASS_NAME = 'SYSDEFAULTUSERCLASS'
  AND SERVICE_SUBCLASS_NAME = 'SYSDEFAULTSUBCLASS'
ORDER BY STATISTICS_TIMESTAMP

STATISTICS_TIMESTAMP       LIFETIMEAVG EXECTIMEAVG QUEUETIMEAVG
COMPLETED_TOTAL
-------------------------- ----------- ----------- ------------ ---------
------
2007-08-07-14.07.44.511153         508         475            0
77
2007-08-07-14.07.46.537777         513         508            0
39
2007-08-07-14.07.51.882173         314         253            0
113

  3 record(s) selected.
```

この結果は、間隔ごとに平均存続時間の方がわずかに平均実行時間より長く、3 つすべてが 0.5 秒前後であることを示しています。平均キュー時間は、予期されているとおりゼロです。各間隔中の完了したアクティビティー数のカウントは予期したとおりです。なぜなら、最初の間隔ではワークロード 1 と 2 が実行された結果 77 のアクティビティーが収集され、2 番目の間隔ではワークロード 1 が単独で実行された結果 39 のアクティビティーが収集され、3 番目の間隔ではワークロード 1 が 2 回とワークロード 2 が 1 回実行された結果 113 のアクティビティーが収集されたからです。

ステップ 6: 次の演習のためのリセット

最後のステップでは、デフォルトのユーザー・サービス・クラス上の集約アクティビティーの収集をオフにして、ビューをドロップし、統計表中の情報を削除します。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS
  UNDER SYSDEFAULTUSERCLASS
  COLLECT AGGREGATE ACTIVITY DATA NONE

DROP VIEW histograms
DROP VIEW histogramtimes
DROP VIEW histogramserviceclasses
DROP VIEW histogramtypes

SET EVENT MONITOR DB2STATISTICS STATE 0

DELETE FROM HISTOGRAMBIN_DB2STATISTICS
DELETE FROM SCSTATS_DB2STATISTICS
```

演習 6: WLM 表関数を使用した遅延の調査

この演習では、DB2 WLM モニター機能を使用して、アプリケーションの低下の原因を判別する方法について説明します。

時間の見積もり: 10 分から 15 分

DB2 WLM モニター機能は、データベース内の処理に関する情報と統計を提供します。低下の原因を識別したら、状態に対処できます。

ステップ 1: アクティビティーの実行

この演習では、app1.db2 と app2.db2 の 2 つのアプリケーションを使用します。両方のアプリケーションとも SAMPLE データベースに対して DML 操作を実行します。1 つ目のウィンドウで app1.db2 スクリプトを実行した直後に 2 つ目のウィンドウで app2.db2 スクリプトを実行します。

```
db2 -tvf app1.db2
db2 -tvf app2.db2
```

ステップ 2: 現在アクティブなワークロード・オカレンスの表示

この時点で app2.db2 スクリプトはハングしているはずです。3 つ目のウィンドウから、表関数 WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES を発行して、データベース上で実行しているすべてのアプリケーションの状態を検索します。この例では、ワークロード・オカレンスをアプリケーションと同じものと見なしてかまいません。この表関数は、サービス・クラス内のワークロード・オカレンスすべてに関する情報を表示します。データベース中のすべてのワークロード・オカレンスを表示したいので、入力パラメーター *service_superclass_name* および *service_subclass_name* には、" で表されるワイルドカードを使用します。

```
CONNECT TO SAMPLE

SELECT INTEGER(APPLICATION_HANDLE) APPL_HANDLE,
   VARCHAR(CLIENT_APPLNAME, 15) AS APPL_NAME,
   VARCHAR(SYSTEM_AUTH_ID, 20) AS USER_ID
   FROM TABLE
   (WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', -2))
```

出力は、以下のようになります。

```
APPL_HANDLE APPL_NAME       USER_ID
----------- --------------- --------------------
         12 CLP app1.db2    DB2USR1
         17 CLP app2.db2    DB2USR1
         18 -               DB2USR1
         19 -               DB2USR1

  4 record(s) selected.
```

この出力から、app2.db2 のアプリケーション・ハンドルが 17 であることが分かります。

ステップ 3: アプリケーションのエージェントの検索

実行されている app2.db2 のエージェントを検出するには、WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用します。この表関数は、サービス・クラス内で作動しているエージェントに関する情報を表示します。アプリケ

ーション・ハンドル 17 に関する作動中のエージェントを表示するので、このハンドルを application_handle 入力パラメーターで指定します。この例の場合、特定のサービス・クラスのエージェントは対象ではないので、service_superclass_name および service_subclass_name 入力パラメーターにワイルドカードを指定します。

```
SELECT INTEGER(APPLICATION_HANDLE) AS APPL_HANDLE,
   UOW_ID, ACTIVITY_ID,
   VARCHAR(AGENT_TYPE, 15) AS AGENT_TYPE,
   VARCHAR(AGENT_STATE, 10) AS AGENT_STATE,
   VARCHAR(EVENT_TYPE, 10) AS EVENT_TYPE,
   VARCHAR(EVENT_OBJECT, 10) AS EVENT_OBJ,
   VARCHAR(EVENT_STATE, 10) AS EVENT_STATE
FROM TABLE
   (WLM_GET_SERVICE_CLASS_AGENTS('', '', 17, -2))
```

出力は、以下のようになります。

```
APPL_HANDLE UOW_ID      ACTIVITY_ID AGENT_TYPE      AGENT_STATE EVENT_TYPE
EVENT_OBJ  EVENT_STATE
----------- ----------- ----------- --------------- ----------- ----------
---------- -----------
         17           1           2 COORDINATOR     ACTIVE      ACQUIRE
LOCK       IDLE

  1 record(s) selected.
```

この出力から、アプリケーション 17 のコーディネーター・エージェントがアイドル状態で、ロックの獲得を待っていることが分かります。app2.db2 がハングしているように見えるのは、この理由によります。

ステップ 4: 問題アプリケーションの検索と問題の解決

アプリケーションがハングしている理由が分かったので、状態に対処できます。アプリケーションがロックを待っていることが分かっています。このアプリケーションが待っているロックと、そのロックを保持しているアプリケーションを検出するには、db2pd ツールを使用できます。最初にハングしているアプリケーションの現行トランザクション番号を検出する必要があります。アプリケーション・ハンドル 17 を対象に db2pd -transactions を発行します。

```
db2pd -db sample -transactions app=17
```

出力は、以下のようになります。

```
Address            AppHandl [nod-index]  TranHdl    Locks      State
Tflag      Tflag2     Firstlsn       Lastlsn        LogSpace
SpaceReserved   TID            AxRegCnt   GXID

0x07000000302A7080 17       [000-00017] 7          5          READ
0x00000000 0x00000000 0x000000000000 0x000000000000 0
0               0x000000000AC3 1          0
```

この出力から、アプリケーション 17 にトランザクション・ハンドル 7 があることが分かります。トランザクション・ハンドル 7 を対象に db2pd -locks コマンドを発行すると、このトランザクションが待っているロックを検索できます。

```
db2pd -db sample -locks 7 wait
```

出力は、以下のようになります。

```
Address            TranHdl    Lockname                   Type       Mode Sts
Owner      Dur HoldCount  Att  ReleaseFlg
0x07000000304013F0 7          00020010000000000640002D52 Row        .NS  W
2          1   0          0x00 0x00000002
```

この出力は、アプリケーションが行ロックを待っていることを示しています。ロックの所有者にはトランザクション・ハンドル 2 があります。このトランザクションがロックを保持しており、ハングの原因になっています。最終ステップではトランザクション・ハンドル 2 に対応するアプリケーション・ハンドルを判別します。トランザクション・ハンドル 2 を対象に db2pd -transactions コマンドを発行します。

```
db2pd -db sample -transactions 2
```

出力は、以下のようになります。

```
Address            AppHandl [nod-index] TranHdl    Locks      State
Tflag      Tflag2     Firstlsn       Lastlsn        LogSpace
SpaceReserved   TID            AxRegCnt   GXID
0x07000000302A2080 12       [000-00012] 2          6          WRITE
0x00000000 0x00000000 0x000002EE000C 0x000002EE005E 232
396             0x000000000ABB 1          0
```

この出力から、トランザクション・ハンドル 2 がアプリケーション・ハンドル 12 に対応していることが分かります。表関数 WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES の結果に戻って確認すると、アプリケーション 12 が app1.db2 を参照していることが分かります。このアプリケーションは、app2.db2 で必要な行ロックを保持しています。 app2.db2 を進めるには、app1.db2 を実行しているウィンドウから、作業単位またはプロセスのコミット、ロールバック、または終了を実行できます。 別の方法として、アプリケーション・ハンドル 12 に対して FORCE APPLICATION を発行して、app1.db2 を強制的にオフにすることもできます。

```
db2 force application (12)
```

追加情報: ロック競合のためにハングしているアプリケーションを診断する別の方法としては、SNAPSHOT_LOCKWAIT モニター表関数を使用できます。この表関数は、ロックの保持者と待機者に関する情報を提供します。この表関数を使用するには、データベースを開始する前に、DFT_MON_LOCK モニター・スイッチ構成パラメーターをオンにする必要があります。このスイッチは、インスタンス上のすべてのデータベースに影響があります。

演習 7: 継続中のアクティビティーの取り消し

この演習では、WLM_CANCEL_ACTIVITY プロシージャーを使用して、現在アクティブなアクティビティーを取り消す方法について説明します。

時間の見積もり: 5 分から 10 分

ステップ 1: 長期実行照会の発行

CLP ウィンドウから、長期実行照会を発行する、以下のスクリプトを実行します。

```
db2 -tvf longquery.db2
```

ステップ 2: アプリケーション・ハンドルの取得

別の CLP ウィンドウから、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES を呼び出して、カーソル・アクティビティーのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を取得します。

```
SELECT T.APPLICATION_HANDLE, T.UOW_ID, T.ACTIVITY_ID, T.ACTIVITY_TYPE
FROM SYSIBMADM.APPLICATIONS A,
     TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES
             (CAST(NULL AS BIGINT), -2)) T
     WHERE (A.AGENT_ID = T.APPLICATION_HANDLE) AND
           (A.COORD_MEMBER = T.COORD_MEMBER) AND
           (A.MEMBER = T.MEMBER) AND
           (T.MEMBER = T.COORD_MEMBER) AND
           (A.TPMON_CLIENT_APP = 'CLP longquery.db2')
```

この表関数の結果を APPLICATIONS 管理ビューと組み合わせると、longquery.db2 内から実行されるカーソル・アクティビティーを検索できます。出力は、以下のようになります。

```
APPLICATION_HANDLE   UOW_ID      ACTIVITY_ID ACTIVITY_TYPE
-------------------- ----------- ----------- --------------------------
----
                 267           1           1 READ_DML

  1 record(s) selected.
```

ステップ 3: アクティビティーの取り消し

同じ CLP ウィンドウから、前のステップで入手したアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を使用し、WLM_CANCEL_ACTIVITY ストアード・プロシージャーを呼び出して前述のカーソル・アクティビティーを取り消します。

```
CONNECT TO SAMPLE

CALL WLM_CANCEL_ACTIVITY (267, 1, 1)

CONNECT RESET
```

事例ごとにアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID が違うことに注意してください。

longquery.db2 で発行された長期実行照会から以下の出力が戻され、最初の CLP ウィンドウに表示されます。

```
SQL4725N  The activity has been cancelled. SQLSTATE=57014
```

演習 8: システムで実行中のアクティビティー・タイプの発見

この演習では、DB2 ワークロード管理モニター表関数とワーク・アクション・セットを使用して、システム上で実行しているアクティビティーのタイプを発見する方法について説明します。

時間の見積もり: 15 分から 20 分

例えば、システム上で並行して実行されている大きなアクティビティーまたはロード・ユーティリティーの数を知ることができます。作業のタイプが違うとリソース

要件も異なり、システム・パフォーマンスに影響するので、システム上で実行されている作業のタイプを知ることは重要です。

ステップ 1: システム上で実行しているタイプごとのアクティビティー数の判別

始める前に、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用して、現在実行されている特定のタイプのアクティビティーの数を表示することができます。

```
CONNECT TO SAMPLE

SELECT ACTIVITY_TYPE,
     COUNT(*) AS NUMBER_RUNNING
     FROM TABLE (
     WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(CAST(NULL AS BIGINT), -2)) AS T
GROUP BY ACTIVITY_TYPE
```

この照会からの出力は、以下のようになります。

```
ACTIVITY_TYPE                NUMBER_RUNNING
---------------------------- ----------------------
READ_DML                      1
```

特定の期間システム上で実行されたさまざまなタイプのアクティビティーに関する情報を取得するには、ワーク・クラス・セットとワーク・アクションを使用できます。

ステップ 2: COUNT ACTIVITY ワーク・アクションを使用したデータベース・ワーク・アクション・セットの作成

ある期間に特定のタイプのアクティビティーが実行された回数をカウントするには、ワーク・アクション・セットを作成する必要があります。この例では、システム全体で実行されるアクティビティーが対象なので、ワーク・アクション・セットをデータベース・レベルで作成し、演習 4 のステップ 1 で作成した all_class_types ワーク・クラス・セットと関連付けます。このワーク・クラス・セットにはすべてのタイプの認識されているアクティビティーに関するワーク・クラスが含まれています。特定のサービス・クラス中で実行されているアクティビティーのみを対象とする場合は、サービス・クラス・レベルでワーク・アクション・セットを作成することになります。この例の場合、すべてのタイプのアクティビティーに関する情報も対象なので、ワーク・アクション・セットには、all_class_types ワーク・クラス・セットにあるそれぞれのワーク・クラスに対する COUNT ACTIVITY ワーク・アクションが含まれます。

```
CREATE WORK ACTION SET work1_was FOR DATABASE
   USING WORK CLASS SET all_class_types
    (WORK ACTION count_read_wa ON WORK CLASS read_wc COUNT ACTIVITY,
    WORK ACTION count_write_wa ON WORK CLASS write_wc COUNT ACTIVITY,
    WORK ACTION count_ddl_wa ON WORK CLASS ddl_wc COUNT ACTIVITY,
    WORK ACTION count_call_wa ON WORK CLASS call_wc COUNT ACTIVITY,
    WORK ACTION count_load_wa ON WORK CLASS load_wc COUNT ACTIVITY,
    WORK ACTION count_all_wa on WORK CLASS all_wc COUNT ACTIVITY)
```

追加情報: ワーク・クラスに対応するアクティビティーに、1 つ以上のワーク・アクションが適用されるたびに、そのワーク・クラスのカウンターが 1 ずつ増分されます。COUNT ACTIVITY ワーク・アクションにより、このカウンターは効率的な仕方で更新されます。このタイプの実行されたアクティビティー数のカウント以外の

アクションをアクティビティーに対して実行しない場合は、COUNT ACTIVITY ワーク・アクションが最善の方法です。

ステップ 3: 一部のアクティビティーの実行

1 回 work1.db2 スクリプトを実行します。

```
db2 -tvf work1.db2
```

ステップ 4: ワーク・アクション・セット統計の表示

WLM_GET_WORK_ACTION_SET_STATS 表関数を使用して、メモリー内のワーク・アクション・セット統計にアクセスし、特定のアクティビティー・タイプが実行された回数を取得できます。例えば、以下の照会は、ワーク・アクションが関連付けられているワーク・クラス・セット内にある各ワーク・クラスに割り当てられたアクティビティーの数を示します。

```
CONNECT TO SAMPLE

SELECT SUBSTR(WORK_ACTION_SET_NAME, 1, 12) AS WORK_ACTION_SET_NAME,
               SUBSTR(WORK_CLASS_NAME, 1, 12) AS WORK_CLASS_NAME,
               LAST_RESET,
               SUBSTR(CHAR(ACT_TOTAL), 1, 12) AS TOTAL_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS('', -2)) AS WASSTATS
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME, LAST_RESET
```

追加情報: ステートメントに組み込まれているブランクは、引数によって結果が制限されていないことを意味します (この例では、すべてのワーク・アクション・セットに関する情報が必要です)。最後の引数 member の値はワイルドカード文字 -2 であり、これはすべてのデータベース・メンバーからのデータが返されることを意味します。

この照会からの出力は、以下のようになります。「*」は、定義されているワーク・クラスに該当しないか、またはワーク・アクションのないワーク・クラスに該当するすべてのアクティビティーを表します。

```
WORK_ACTION_SET_NAME WORK_CLASS_NAME LAST_RESET
TOTAL_ACTS
-------------------- -------------- -------------------------- ---------
-
WORK1_WAS            *              2007-08-14-13.55.30.725886 0
WORK1_WAS            ALL_WC         2007-08-14-13.55.30.725886 2
WORK1_WAS            CALL_WC        2007-08-14-13.55.30.725886 4
WORK1_WAS            DDL_WC         2007-08-14-13.55.30.725886 12
WORK1_WAS            LOAD_WC        2007-08-14-13.55.30.725886 1
WORK1_WAS            READ_WC        2007-08-14-13.55.30.725886 12
WORK1_WAS            WRITE_WC       2007-08-14-13.55.30.725886 6

  7 record(s) selected.
```

ステップ 5: タイプなどの属性以上によるアクティビティーの区別

単なるタイプ以上のものでアクティビティーを分離できます。例えば、実行されている大きな照会の数を知ることができます。

ワーク・クラス・セットに変更を加えて、大きな照会を表す新しい読み取りワーク・クラスを追加します。この例の場合、カーディナリティーが 40 を超える照会が大きな照会になります。

```
ALTER WORK CLASS SET all_class_types
   ADD WORK CLASS large_wc WORK TYPE READ FOR CARDINALITY FROM 41 POSITION AT 1
```

追加情報: このワーク・クラスの位置を 1 にしたことに注意してください。POSITION AT 節を指定しないと、このワーク・クラスの位置はワーク・クラス・セットの下部になります。アクティビティーが属するワーク・クラスを決める際には、ワーク・クラスが位置順にチェックされ、アクティビティーの属性と一致する属性のある最初のワーク・クラスが、アクティビティーの割り当て先のクラスになります。この演習では、large_wc の位置がリストの末尾の場合、read_wc since の位置が large_wc の前なので、大きなアクティビティーは read_wc since に割り当てられています。

ワーク・アクション・セットに変更を加えて、COUNT ACTIVITY ワーク・アクションを追加し、新しいワーク・クラスに適用します。

```
ALTER WORK ACTION SET work1_was
    ADD WORK ACTION count_large_reads ON WORK CLASS large_wc COUNT ACTIVITY
```

ステップ 6: 統計のリセットと一部のアクティビティーの実行

WLM_COLLECT_STATS ストアード・プロシージャーを呼び出して、メモリー内に保管されている統計をリセットすることにより、フレッシュを開始できるようにして、メモリー内に保管されているワークロード管理統計情報の照会を選択する際に、この時点から実行されたアクティビティーに関する情報が含まれるようにします。

```
CALL WLM_COLLECT_STATS()
```

1 回 work1.db2 スクリプトを実行します。

```
db2 -tvf work1.db2
```

ステップ 7: ワーク・アクション・セット統計の表示

再度 WLM_GET_WORK_ACTION_SET_STATS 表関数を使用して、メモリー内のワーク・アクション・セット統計にアクセスし、特定のアクティビティー・タイプを実行した回数を取得します。

```
CONNECT TO SAMPLE

SELECT SUBSTR(WORK_ACTION_SET_NAME, 1, 12) AS WORK_ACTION_SET_NAME,
       SUBSTR(CHAR(MEMBER), 1, 4) AS MEMB,
       SUBSTR(WORK_CLASS_NAME, 1, 12) AS WORK_CLASS_NAME,
       LAST_RESET,
       SUBSTR(CHAR(ACT_TOTAL), 1, 12) AS TOTAL_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS('', -2)) AS WASSTATS
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME, MEMB
```

出力は、以下のようになります。

```
WORK_ACTION_SET_NAME MEMB WORK_CLASS_NAME LAST_RESET
TOTAL_ACTS
-------------------- ---- -------------- -------------------------- ----
------
WORK1_WAS            0    *              2007-08-14-13.55.35.650685 0
WORK1_WAS            0    ALL_WC         2007-08-14-13.55.35.650685 2
WORK1_WAS            0    CALL_WC        2007-08-14-13.55.35.650685 4
WORK1_WAS            0    DDL_WC         2007-08-14-13.55.35.650685 12
WORK1_WAS            0    LARGE_WC       2007-08-14-13.55.35.650685 4
WORK1_WAS            0    LOAD_WC        2007-08-14-13.55.35.650685 1
```

```
WORK1_WAS            0    READ_WC         2007-08-14-13.55.35.650685 8
WORK1_WAS            0    WRITE_WC        2007-08-14-13.55.35.650685 6

  8 record(s) selected.
```

今回は、スクリプトからの 4 つのアクティビティーが大きなアクティビティーと見なされていることに注意してください。

ステップ 8: 次の演習のためのリセット

以下のように、ワーク・アクション・セットをドロップします。

```
ALTER WORK ACTION SET WORK1_WAS
 ALTER WORK ACTION COUNT_READ_WA DISABLE
 ALTER WORK ACTION COUNT_WRITE_WA DISABLE
 ALTER WORK ACTION COUNT_DDL_WA DISABLE
 ALTER WORK ACTION COUNT_CALL_WA DISABLE
 ALTER WORK ACTION COUNT_LOAD_WA DISABLE
 ALTER WORK ACTION COUNT_ALL_WA DISABLE
 ALTER WORK ACTION COUNT_LARGE_READS DISABLE;
ALTER WORK ACTION SET WORK1_WAS DISABLE;
DROP WORK ACTION SET WORK1_WAS;
```

演習 9: 実行中のアクティビティーに関する詳細情報のキャプチャー

この演習では、WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを使用して、後で履歴分析を行うために、現在実行中のアクティビティーに関する詳細情報をキャプチャーする方法について説明します。

時間の見積もり: 5 分から 10 分

キャプチャーしたアクティビティー情報は、アクティビティーに関するアクティブなイベント・モニターに送信されます。これまでのタスクでは、ワークロード、サービス・クラス、ワーク・アクション、およびしきい値に関して COLLECT ACTIVITY DATA 節を使用して、詳細なアクティビティー情報をキャプチャーする方法が示されました。アクティビティーの実行が開始され、そのアクティビティーの完了時にアクティビティー・イベント・モニターに情報が送信される前に、この節をあらかじめ指定する必要があります。
WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを使用すると、すでに進行中のアクティビティーに関する問題に気付いた時点で、反応的に情報をキャプチャーできます。このプロシージャーを使用すると、アクティビティーに関する情報は即時にアクティビティー・イベント・モニターに送信されます。基本情報とステートメント・アクティビティー情報が両方とも収集されますが、入力データは収集されません。

ステップ 1: アクティビティー・イベント・モニターの使用可能化

演習 1 で作成したアクティビティーに関する既存のイベント・モニターを使用可能にします。

```
CONNECT TO SAMPLE

SET EVENT MONITOR DB2ACTIVITIES STATE 1
```

ステップ 2: 長期実行照会の発行

CLP から、問題のあるカーソルを使用して長期実行照会を発行する、以下のスクリプトを実行します。

```
db2 -tvf longquery.db2
```

ステップ 3: アプリケーション・ハンドルの取得

2 つ目の CLP ウィンドウから、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES を呼び出して、このアクティビティーに関するアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を入手します。この表関数の結果を APPLICATIONS 管理ビューと JOIN することで、longquery.db2 内から実行されているカーソル・アクティビティーを検索できます。

```
CONNECT TO SAMPLE

SELECT T.APPLICATION_HANDLE, T.UOW_ID, T.ACTIVITY_ID, T.ACTIVITY_TYPE
FROM SYSIBMADM.APPLICATIONS A,
     TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES
          (CAST(NULL AS BIGINT), -2)) T
WHERE (A.AGENT_ID = T.APPLICATION_HANDLE) AND
          (A.COORD_MEMBER = T.COORD_MEMBER) AND
          (A.MEMBER = T.MEMBER) AND
          (T.MEMBER = T.COORD_MEMBER) AND
          (A.TPMON_CLIENT_APP = 'CLP longquery.db2')
```

出力は、以下のようになります。

```
APPLICATION_HANDLE   UOW_ID      ACTIVITY_ID ACTIVITY_TYPE
-------------------- ----------- ----------- ----------------------------
----
                 267           1           1 READ_DML

1 record(s) selected.
```

ステップ 4: アクティビティーに関する情報のキャプチャー

同じ CLP ウィンドウから、前のステップで入手したアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を使用して、WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを呼び出します。

```
CONNECT TO SAMPLE

CALL WLM_CAPTURE_ACTIVITY_IN_PROGRESS (267, 1, 1)

CONNECT RESET
```

このステップでは、アクティビティーに関する情報が、アクティビティーに関するアクティブなイベント・モニターに送信されます。事例ごとに、指定するアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID が違う可能性があることに注意してください。

ステップ 5: アクティビティー情報の表示

以下のようなステートメントを使用して、アクティビティーに関する収集済みの情報を表示します。

```
SELECT VARCHAR(A.APPL_NAME, 15) as APPL_NAME,
     VARCHAR(A.TPMON_CLIENT_APP, 20) AS CLIENT_APP_NAME,
     VARCHAR(A.APPL_ID, 30) as APPL_ID,
     A.ACTIVITY_ID,
     A.UOW_ID,
     A.PARTIAL_RECORD,
     A.TIME_STARTED,
     A.TIME_COMPLETED,
     VARCHAR(S.STMT_TEXT, 300) AS STMT_TEXT
FROM ACTIVITY_DB2ACTIVITIES AS A,
     ACTIVITYSTMT_DB2ACTIVITIES AS S
WHERE A.APPL_ID = S.APPL_ID AND
     A.ACTIVITY_ID = S.ACTIVITY_ID AND
     A.UOW_ID = S.UOW_ID
```

出力は、以下のようになります。

```
APPL_NAME       CLIENT_APP_NAME      APPL_ID
ACTIVITY_ID          UOW_ID      PARTIAL_RECORD TIME_STARTED
TIME_COMPLETED             STMT_TEXT
--------------- -------------------- ----------------------------- -----
--------------- ----------- ------------- ------------------------- ---
---------------------- --------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------
db2bp           CLP longquery.db2    *LOCAL.swalkty.070928151408
1           1              1 2007-09-28-11.14.09.334636 0000-00-00-
00.00.00.000000 SELECT COUNT(*) FROM SYSCAT.TABLES, SYSCAT.TABLES,
SYSCAT.TABLES, SYSCAT.TABLES, SYSCAT.TABLES
```

注: WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを使用してキャプチャーされるアクティビティーに関する情報は、COLLECT ACTIVITY DATA 節を使用した場合より幾分少なくなります。その理由は、アクティビティーの実行が完了する前にキャプチャーされるからです。特に、完了タイム・スタンプ (ゼロのみ表示される) や SQLCODE などのフィールドは適用されません。ACTIVITY 表の PARTIAL_RECORD 列を参照すると、WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを使用してアクティビティーが収集されたかどうかを判別できます。PARTIAL_RECORD 列の値が (前述の出力で示されているように) 1 の場合は、WLM_CAPTURE_ACTIVITY_IN_PROGRESS を使用してアクティビティー情報が収集されたことを示します。PARTIAL_RECORD 列の値が 0 の場合は、COLLECT ACTIVITY DATA 節を使用して完了後にアクティビティー情報が収集されたことを示します。

演習 10: 履歴データおよびレポートの生成

この演習では、WLM Historical Analysis Tool サンプルの使用法について説明します。

時間の見積もり: 20 分から 25 分

DB2 データベース製品には、履歴分析のために、WLM アクティビティー・イベント・モニターによってキャプチャーされる情報を提供する Perl のサンプル・スクリプトが含まれています。これらのスクリプトをレビューまたは変更して、ニーズに合った追加の履歴分析レポートを作成することができます。 Perl スクリプトは、次のとおりです。

- wlmhist.pl: 履歴データを生成します。
- wlmhistrep.pl: 履歴データからレポートを作成します。

ステップ 1: Explain 表の作成

履歴データを生成するには、ツールを実行するユーザーのスキーマの下に Explain 表がなければなりません。Explain 表を作成するには、/sqllib/misc ディレクトリーに進んで、以下のコマンドを実行します。

```
db2 CONNECT TO SAMPLE

db2 -tvf EXPLAIN.DDL
```

ステップ 2: アクティビティー・データを収集するようにサービス・クラスを変更する

対象となる WLM オブジェクト上で COLLECT ACTIVITY DATA 節を使用して、アクティビティーの収集を使用可能にします。この演習の場合、デフォルト・ユーザー・サービス・スーパー・クラスのデフォルト・サービス・サブクラス中で実行されるアクティビティーに関する履歴データを生成することができます。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
       COLLECT ACTIVITY DATA ON COORDINATOR WITH DETAILS
```

ステップ 3: アクティビティー・イベント・モニターの使用可能化

演習 1 のステップ 1 でアクティビティー・イベント・モニターを作成したので、まだ使用可能にしていない場合はこの時点で使用可能にします。

```
SET EVENT MONITOR DB2ACTIVITIES STATE 1
```

ステップ 4: 一部のアクティビティーの実行

一部のアクティビティーを実行して、履歴データの生成対象のアクティビティー・データを収集できるようにします。

```
db2 -tvf work1.db2
db2 -tvf work2.db2
```

ステップ 5: アクティビティーに対するイベント・モニターの無効化

履歴データを生成する前に、アクティビティーに関するイベント・モニターをオフにすることを強くお勧めします。オフにしないと、履歴データ生成プログラムの結果として実行される DML アクティビティーもキャプチャーされ、DB2 イベント・モニター・アクティビティー表に挿入される可能性があるので、アクティビティー・データの生成対象の実際のアクティビティーの数が非常に多くなります。

```
CONNECT TO SAMPLE

SET EVENT MONITOR DB2ACTIVITIES STATE 0
```

ステップ 6: 履歴データの生成

履歴データ生成プログラム・スクリプト wlmhist.pl を実行して、アクティビティー・イベント・モニター表にキャプチャーされるアクティビティーに関する履歴データを生成します。形式は次のようになります。

```
wlmhist.pl dbname user password [fromTime toTime workloadid
        serviceClassName serviceSubclassName activityTable activityStmtTable]
```

オプション・パラメーターをバイパスするには、ダッシュ (-) を使用します。

追加情報: 履歴データ生成プログラム (wlmhist.pl) スクリプトは、DML に関する履歴データのみ生成します。以前に履歴データ生成プログラム (wlmhist.pl) スクリプトを 1 回以上実行した場合は、再度実行する前に、データが重複しないように activityTable 表と activityStmtTable 表をクリアすることをお勧めします。これらの 2 つの表をクリアしないことを選択する場合は、fromTime および toTime 入力パラメーターを必ず使用して、すでにデータが生成されているアクティビティーに関する履歴データが生成されることがないようにしてください。

この演習の場合、アクティビティー・イベント・モニターでキャプチャーされたすべてのアクティビティーに関する履歴データを生成します。

```
Perl wlmhist.pl sample db2inst1 password
```

以下のようなエラーが通知されることもあります。

```
Error running explain [IBM][CLI Driver][DB2/LINUXX8664] SQL0418N  A
statement contains a use of a parameter marker that is not valid. SQLSTATE=42610
 for statement VALUES (TABLE_SCHEMA(:H00002 , :H00003   )) INTO :H00007

DBD::DB2::db do failed: [IBM][CLI Driver][DB2/LINUXX8664] SQL0418N  A
statement
contains a use of a parameter marker that is not valid. SQLSTATE=42610
```

履歴データを生成する際には、実際のステートメント上で Explain が実行されます。パラメーター・マーカーのある一部のステートメント上で Explain を実行できず、エラーが戻される場合もあります。この種のエラーが表示されるアクティビティーに関する履歴データは生成されません。

ツールによる履歴データの生成が完了すると、履歴データが正常に生成されたアクティビティーの数が通知されます。

ステップ 7: 履歴データ・レポートの生成

履歴データ・レポート・スクリプト wlmhistrep.pl を実行して、ステップ 1 で生成したデータに基づいてレポートを生成します。形式は次のようになります。

```
wlmhistrep.pl dbAlias userId passwd [outputFile report schemaName fromTime toTime submitter]
```

オプション・パラメーターをバイパスするには、ダッシュ (-) を使用します。

report パラメーターは、以下の文字の組み合わせにすることができます。

- A: ヒットした表
- B: ヒットしなかった表
- C: ヒットした索引
- D: ヒットしなかった索引
- E: サブミッター

userId パラメーターの指定値が、wlmhist 表の作成時に wlmhist.pl スクリプトの実行に使用した値と同じでない場合は、正しい schemaName を指定しなければなりま

せん。 **fromTime** および **toTime** パラメーターはタイム・スタンプ形式で指定しなければなりません (例えば、2007-06-06-17.00.00)。

この演習の場合、ヒットした表とヒットしなかった索引に関するレポートを生成します。

```
Perl wlmhistrep.pl sample db2inst1 password - AD
```

出力は、以下のようになります。

```
                TABLES HIT REPORT FOR DATABASE sample
                ____________________________________

TABLE NAME                TABLE SCHEMA        % HITS        TOTAL HITS
__________________________ ___________________ _____________ ___________

EMPLOYEE                  KARENAM              7.14285714           2
INVENTORY                 KARENAM             14.28571429           4

ORG                       KARENAM             28.57142857           8
SALES                     KARENAM             14.28571429           4
SYSROUTINES               SYSIBM               7.14285714           2
SYSTABLES                 SYSIBM              21.42857143           6
SYSTABLESPACES            SYSIBM               7.14285714           2

              INDEXES NOT HIT REPORT FOR DATABASE sample
              __________________________________________

TABLE NAME         TABLE SCHEMA    INDEX NAME         INDEX SCHEMA    INDEX TYPE
_________________ _______________ _________________ _______________ __________
EXPLAIN_ARGUMENT   KARENAM         ARG_I1             KARENAM         REG
HMON_ATM_INFO      SYSTOOLS        ATM_UNIQ           SYSTOOLS        REG
CUSTOMER           KARENAM         CUST_CID_XMLIDX    KARENAM         XVIL
CUSTOMER           KARENAM         CUST_NAME_XMLIDX   KARENAM         XVIL
CUSTOMER           KARENAM         CUST_PHONES_XMLIDX KARENAM         XVIL
CUSTOMER           KARENAM         CUST_PHONET_XMLIDX KARENAM         XVIL
EXPLAIN_DIAGNOSTIC KARENAM         EXP_DIAG_DAT_I1    KARENAM         REG
HMON_COLLECTION    SYSTOOLS        HI_OBJ_UNIQ        SYSTOOLS        REG
ADVISE_INDEX       KARENAM         IDX_I1             KARENAM         REG
ADVISE_INDEX       KARENAM         IDX_I2             KARENAM         REG
SYSATTRIBUTES      SYSIBM          INDATTRIBUTES01    SYSIBM          REG
SYSATTRIBUTES      SYSIBM          INDATTRIBUTES02    SYSIBM          REG
:
:
```

ステップ 8: 次の演習のためのリセット

デフォルト・ユーザー・サービス・スーパー・クラスのデフォルト・サービス・サブクラスに関するアクティビティー収集を使用不可にして、アクティビティー表をクリーンアップします。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
       COLLECT ACTIVITY DATA NONE

DELETE FROM ACTIVITY_DB2ACTIVITIES
DELETE FROM ACTIVITYSTMT_DB2ACTIVITIES
```

演習 11: サービス・クラスに関する拡張集約の使用

この演習では、サービス・クラスに対して COLLECT AGGREGATE ACTIVITY DATA EXTENDED オプションを使用して、コーディネーター・アクティビティーの到着間隔時間と見積コストのヒストグラムを作成する方法について説明します。

時間の見積もり: 25 分から 30 分

*到着間隔時間*とは、システムへの 1 つのアクティビティーの到着から次のアクティビティーの到着までの間の時間間隔です。アクティビティーの*見積コスト*は、アクティビティーの実行中に使用されるシステム・リソースの SQL コンパイラーによる見積もりを表し、DML アクティビティーのみに適用されます。

到着間隔時間のヒストグラムまたは見積コストのヒストグラムは、相互に、または存続時間ヒストグラムと、あるいは他の存続時間統計と相関して、存続時間ヒストグラムまたは存続時間統計に生じた変更が、下記のイベントのいずれかによるものかどうかを判別します。

- ワークロードの複雑さの変更 (見積コストの分布の変更など)
- アクティビティー到着率の変更 (到着間隔時間の分布から判別する)
- システム自体の変更 (新しいしきい値の導入、サービス・クラスに付与された優先順位の変更、またはハードウェアの変更など)

ヒストグラムについて詳しくは、 292 ページの『ワークロード管理のヒストグラム』を参照してください。

ヒストグラムには、統計イベント・モニターによってアクセスします。この演習では、演習 1 のステップ 1 で作成した統計イベント・モニターを再利用します。

ステップ 1: ヒストグラム統計の表示用のビューの作成

複数のビューを作成して、HISTOGRAMBIN_DB2STATISTICS 表の照会を容易にします。最初のビューは、使用可能なすべてのヒストグラム・タイプをリストします。この演習では、存続時間、実行時間、およびキュー時間の 3 つの基本タイプのみ報告します。

```
CREATE VIEW HISTOGRAMTYPES AS
  SELECT DISTINCT SUBSTR(HISTOGRAM_TYPE,1,24) HISTOGRAM_TYPE
  FROM HISTOGRAMBIN_DB2STATISTICS
```

2 番目のビューは、ヒストグラムの収集対象のサービス・クラスの検出を容易にします。HISTOGRAMBIN_DB2STATISTICS 表は、サービス・クラス ID を指定してヒストグラムを収集する対象のサービス・クラスを報告します。この表と SERVICECLASSES カタログ表を結合すると、サービス・クラス情報を、サービス・クラス ID ではなくサービス・スーパー・クラス名およびサービス・サブクラス名と共に表示できます。

```
CREATE VIEW HISTOGRAMSERVICECLASSES AS
  SELECT DISTINCT SUBSTR(HISTOGRAM_TYPE,1,24) HISTOGRAM_TYPE,
         SUBSTR(PARENTSERVICECLASSNAME,1,24) SERVICE_SUPERCLASS,
         SUBSTR(SERVICECLASSNAME,1,24) SERVICE_SUBCLASS
  FROM HISTOGRAMBIN_DB2STATISTICS H,
       SYSCAT.SERVICECLASSES S
  WHERE H.SERVICE_CLASS_ID = S.SERVICECLASSID
```

3 番目のビューは、特定のサービス・クラスに関する特定のタイプのヒストグラムが収集された時間をすべてリストします。HISTOGRAMSERVICECLASSES ビューと同様に、HISTOGRAMBIN_DB2STATISTICS 表を SERVICECLASSES カタログ表と結合します。違う点は、このビューには列の 1 つとして STATISTICS_TIMESTAMP 列が組み込まれていることです。

```
CREATE VIEW HISTOGRAMTIMES AS
  SELECT DISTINCT SUBSTR(HISTOGRAM_TYPE,1,24) HISTOGRAM_TYPE,
         SUBSTR(PARENTSERVICECLASSNAME,1,24) SERVICE_SUPERCLASS,
         SUBSTR(SERVICECLASSNAME,1,24) SERVICE_SUBCLASS,
         STATISTICS_TIMESTAMP TIMESTAMP
  FROM HISTOGRAMBIN_DB2STATISTICS H,
       SYSCAT.SERVICECLASSES S
  WHERE H.SERVICE_CLASS_ID = S.SERVICECLASSID
```

4 番目と最後のビューは、ヒストグラム自体を表示するのに使用します。また、ヒストグラムを処理する際の共通タスクである、長期にわたるヒストグラムの集約についても説明します。このビューは、各ビンの最上位と、各ビンに対してカウントされたアクティビティーの数を表示します。以下の 2 つのヒストグラムのうち、BIN_TOP フィールドはアクティビティー到着間隔時間のミリ秒数と見積コストの timeron の数を測定します。例えば、到着間隔時間ヒストグラムの場合に、BIN_TOP が 3000 ミリ秒で、直前のビンの BIN_TOP が 2000 ミリ秒で、NUMBER_IN_BIN が 10 のときには、10 個のアクティビティーがあり、各アクティビティーはその前のアクティビティーの到着後 2 秒から 3 秒までの間にシステムに到着したことを示しています。

```
CREATE VIEW HISTOGRAMS(HISTOGRAM_TYPE, SERVICE_SUPERCLASS,
  SERVICE_SUBCLASS, BIN_TOP, NUMBER_IN_BIN) AS
  SELECT DISTINCT SUBSTR(HISTOGRAM_TYPE,1,24) HISTOGRAM_TYPE,
         SUBSTR(PARENTSERVICECLASSNAME,1,24) SERVICE_SUPERCLASS,
         SUBSTR(SERVICECLASSNAME,1,24) SERVICE_SUBCLASS,
         TOP AS BIN_TOP,
         SUM(NUMBER_IN_BIN) AS NUMBER_IN_BIN
  FROM HISTOGRAMBIN_DB2STATISTICS H,
       SYSCAT.SERVICECLASSES S
  WHERE H.SERVICE_CLASS_ID = S.SERVICECLASSID
  GROUP BY HISTOGRAM_TYPE, PARENTSERVICECLASSNAME, SERVICECLASSNAME, TOP
```

ステップ 2: ヒストグラムの収集をオンにする

デフォルト・ユーザー・サービス・クラスに関するヒストグラムの収集をオンにするには、EXTENDED オプションを指定して集約アクティビティー・データを収集するようにデフォルト・サブクラスを変更します。こうすると、BASE オプションで使用可能な 3 つのヒストグラム (存続時間、実行時間、およびキュー時間) と、EXTENDED オプションの使用時のみ使用可能な 2 つのヒストグラム (到着間隔時間および見積コスト) の両方が提供されます。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
   COLLECT AGGREGATE ACTIVITY DATA EXTENDED
```

ステップ 3: 統計イベント・モニターのアクティブ化

以前に作成したイベント・モニターがまだアクティブでない場合はアクティブにして、集約データが収集されたら必ず受信できるようにします。

```
SET EVENT MONITOR DB2STATISTICS STATE 1
```

ステップ 4: アクティビティーの実行および統計の統計イベント・モニターへの送信

最初に、一部のアクティビティーを実行します。アクティビティーの終了後に、WLM_COLLECT_STATS ストアード・プロシージャーを呼び出して、サービス・クラス統計をアクティブな統計イベント・モニターに送信します (デフォルトのユーザー・サービス・クラスに関するアクティビティーの存続時間、実行時間、キュー時間、到着間隔時間、および見積コストのヒストグラムを含む)。これらのヒストグラムには、集約アクティビティー統計を使用可能にした以降にデフォルトのユーザー・サービス・クラス中で実行されたすべてのアクティビティーに関するデータが含まれます。このストアード・プロシージャーを呼び出すと、統計のリセットも行われます。長期にわたるデータベース・アクティビティーの変更を示すために、3 つの収集間隔が作成されます。

最初の間隔では、work1.db2 および work2.db2 の 2 つのスクリプトを実行してから、統計を収集してリセットします。

```
db2 -o- -tvf work1.db2
db2 -o- -tvf work2.db2

CONNECT TO SAMPLE

CALL WLM_COLLECT_STATS
```

2 番目の間隔では、work1.db2 スクリプトのみ 1 回実行してから、統計を収集してリセットします。

```
db2 -o- -tvf work1.db2

CONNECT TO SAMPLE

CALL WLM_COLLECT_STATS
```

3 番目の間隔では、work1.db2 スクリプトを 2 回実行し、work2.db2 スクリプトを 1 回実行してから、統計を収集してリセットします。

```
db2 -o- -tvf work1.db2
db2 -o- -tvf work2.db2
db2 -o- -tvf work1.db2

CONNECT TO SAMPLE

CALL WLM_COLLECT_STATS
```

このようにデータを定期的に収集すると、長期にわたってシステム上の作業の変更内容を監視できます。

追加情報: 手動操作でデータを定期的に収集する必要はありません。WLM_COLLECT_INT データベース構成パラメーターを使用すると、間隔を分数で設定でき、その間隔の後に統計の収集とリセットが行われます。

ステップ 5: 統計を表示するための照会ビュー

統計が収集されたので、以前に作成したビューを使用して統計を表示できます。HISTOGRAMTYPES ビューは単に使用可能なヒストグラムのタイプを戻します。

```
SELECT * FROM HISTOGRAMTYPES

HISTOGRAM_TYPE
```

```
-----------------------
CoordActEstCost
CoordActExecTime
CoordActInterArrivalTime
CoordActLifetime
CoordActQueueTime

  3 record(s) selected.
```

サービス・クラスの変更時に EXTENDED オプションを使用したので、5 つのヒストグラムがあります。

HISTOGRAMSERVICECLASSES ビューを使用すると、ヒストグラムの収集対象のサービス・クラスを表示できます。以下の例は、出力を、CoordActInterArrivalTime ヒストグラムの出力のみに制限します。デフォルト・ユーザー・サービス・クラスのデフォルト・サブクラスのみに関する集約アクティビティーの収集をオンにしたので、HISTOGRAMSERVICECLASSES ビューからの選択時にこのクラスのみ表示されます。

```
SELECT * FROM HISTOGRAMSERVICECLASSES
  WHERE HISTOGRAM_TYPE = 'CoordActInterArrivalTime'
  ORDER BY SERVICE_SUPERCLASS, SERVICE_SUBCLASS

HISTOGRAM_TYPE           SERVICE_SUPERCLASS       SERVICE_SUBCLASS
------------------------ ------------------------ ------------------------
-
CoordActInterArrivalTime SYSDEFAULTUSERCLASS      SYSDEFAULTSUBCLASS

  1 record(s) selected.
```

HISTOGRAMTIMES ビューは、ヒストグラムが収集された回数を表示します。WLM_COLLECT_STATS プロシージャーは 3 回実行されたので、下記の到着間隔時間ヒストグラムには 3 つのタイム・スタンプがあります。

```
SELECT * FROM HISTOGRAMTIMES
  WHERE HISTOGRAM_TYPE = 'CoordActInterArrivalTime'
    AND SERVICE_SUPERCLASS = 'SYSDEFAULTUSERCLASS'
    AND SERVICE_SUBCLASS = 'SYSDEFAULTSUBCLASS'
  ORDER BY TIMESTAMP

HISTOGRAM_TYPE           SERVICE_SUPERCLASS       SERVICE_SUBCLASS
TIMESTAMP
------------------------ ------------------------ ------------------------
- --------------------------
CoordActInterArrivalTime SYSDEFAULTUSERCLASS      SYSDEFAULTSUBCLASS
2007-08-08-13.41.38.870298
CoordActInterArrivalTime SYSDEFAULTUSERCLASS      SYSDEFAULTSUBCLASS
2007-08-08-13.41.42.802855
CoordActInterArrivalTime SYSDEFAULTUSERCLASS      SYSDEFAULTSUBCLASS
2007-08-08-13.41.53.577835
```

最後のビューの HISTOGRAMS は、ヒストグラム自体を表示します。HISTOGRAMTIMES ビューは各収集間隔を独自の行としてリストしますが、このビューはそれとは違って複数の間隔の間でヒストグラム・データを集約し、特定のサービス・クラスに関する特定のタイプのヒストグラムを 1 つ作成します。

```
SELECT BIN_TOP, NUMBER_IN_BIN FROM HISTOGRAMS
  WHERE HISTOGRAM_TYPE = 'CoordActInterArrivalTime'
    AND SERVICE_SUPERCLASS = 'SYSDEFAULTUSERCLASS'
    AND SERVICE_SUBCLASS = 'SYSDEFAULTSUBCLASS'
  ORDER BY BIN_TOP
```

```
BIN_TOP              NUMBER_IN_BIN
-------------------- --------------------
                  -1                    0
                   1                   10
                   2                    6
                   3                    7
                   5                   14
                   8                    7
                  12                   32
                  19                    2
                  29                    9
                  44                   24
                  68                   11
                 103                    8
                 158                    8
                 241                    9
                 369                    1
                 562                   10
                 858                    5
                1309                    5
                1997                    0
                3046                    0
                4647                    0
                7089                    0
               10813                    2
               16493                    2
               25157                    0
               38373                    0
               58532                    0
               89280                    0
              136181                    0
              207720                    0
              316840                    0
              483283                    0
              737162                    0
             1124409                    0
             1715085                    0
             2616055                    0
             3990325                    0
             6086529                    0
             9283913                    0
            14160950                    0
            21600000                    0

  41 record(s) selected.
```

アクティビティーの到着間隔時間はシステムのパフォーマンスに応じて異なるので、この照会を実行して作成される出力は、前述のものとまったく同じにはなりません。前述の出力では、41 個のビンがあり、最大のビンはすべて空です。最上部には、BIN_TOP が -1 のビンがあります。このビンは、到着間隔時間が長すぎてヒストグラムに収まらないすべてのアクティビティーを表します。BIN_TOP が -1 の際に、NUMBER_OF_BIN がゼロより大きい場合は、おそらくヒストグラムのビンの上限値を大きくする必要があることを示しています。前述の出力では、NUMBER_IN_BIN は 0 なので、このような変更を加える必要はありません。1309 ミリ秒未満に到着したアクティビティーの大多数は、相互に異なります。7089 ミリ秒から 16493 ミリ秒までの間に到着した 4 つのアクティビティーは異なります。

CoordActInterArrivalTime の代わりに CoordActEstCost の histogram_type を使用しても同じ照会を繰り返すことができます。

```
SELECT BIN_TOP, NUMBER_IN_BIN FROM HISTOGRAMS
  WHERE HISTOGRAM_TYPE = 'CoordActEstCost'
    AND SERVICE_SUPERCLASS = 'SYSDEFAULTUSERCLASS'
```

```
    AND SERVICE_SUBCLASS = 'SYSDEFAULTSUBCLASS'
  ORDER BY BIN_TOP

BIN_TOP              NUMBER_IN_BIN
-------------------- --------------------
                  -1                    0
                   1                   39
                   2                    0
                   3                    0
                   5                    0
                   8                   30
                  12                    0
                  19                   30
                  29                    0
                  44                    0
                  68                    0
                 103                    0
                 158                    0
                 241                    0
                 369                    0
                 562                    0
                 858                    0
                1309                    0
                1997                    0
                3046                    0
                4647                    0
                7089                    0
               10813                    0
               16493                    0
               25157                    0
               38373                    0
               58532                    0
               89280                    0
              136181                    0
              207720                    0
              316840                    0
              483283                    0
              737162                    0
             1124409                    0
             1715085                    0
             2616055                    0
             3990325                    0
             6086529                    0
             9283913                    0
            14160950                    0
            21600000                    0

  41 record(s) selected.
```

このようなヒストグラムは、ワークロードが小さい場合に標準的なものです。ワークロードが小さい場合、アクティビティーのサイズには大きな変化はないので、3 種類のビンのみに対するアクティビティーがカウントされています。60% をわずかに超えるアクティビティーのコストの見積もりは 5 timeron から 19 timeron までの間で、残りのアクティビティーのコストの見積もりは 1 timeron 未満です。

ステップ 6: 他の演習のためのリセット

最後のステップでは、デフォルトのユーザー・サービス・クラス上の集約アクティビティーの収集をオフにして、ビューをドロップし、統計表中の情報を削除します。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
   COLLECT AGGREGATE ACTIVITY DATA NONE
```

```
DROP VIEW HISTOGRAMS
DROP VIEW HISTOGRAMTIMES
DROP VIEW HISTOGRAMSERVICECLASSES
DROP VIEW HISTOGRAMTYPES

SET EVENT MONITOR DB2STATISTICS STATE 0

DELETE FROM HISTOGRAMBIN_DB2STATISTICS
DELETE FROM SCSTATS_DB2STATISTICS
```

第 7 章 ワークロード管理のシナリオ

ワークロード管理のサンプル・アプリケーション

包括的なワークロード管理フィーチャーが DB2 ワークロード管理とともに DB2 データ・サーバーに組み込まれたことで、アクティビティー、リソース、およびパフォーマンスのより細かな制御が可能になるとともに、システムの実行状況の洞察を深めることができるようになりました。現在、ワークロード管理のサンプル・アプリケーションは developerWorks® で入手できます。

ワークロード管理のサンプル・アプリケーションは、DB2 ワークロード管理フィーチャーを使用して以下の目標を達成する方法を説明します。

照会の暴走からシステムを保護する

照会の暴走は多くのコストを費やし、パフォーマンス低下の原因となります。ワークロード管理のサンプル・アプリケーションは暴走する可能性のある照会を識別し、そのような照会が指定されたしきい値に違反した後、その実行を停止させます。

個別のアプリケーションによるリソースの同時使用量を制限する

サンプル・アプリケーションは、DB2 ワークロード管理フィーチャーを使用して、大量の並行作業をサブミットするアプリケーションが他のアプリケーションのパフォーマンスにマイナスの影響を与えないようにする方法を示します。

特定の応答時間を達成する

ワークロード管理フィーチャーによって、他のどんなアクティビティーがシステム上で同時に実行されているかにかかわりなく「アプリケーション Y からのトランザクション X は 90% のケースにおいて 1 秒以内で完了する」といった形式の特定の回答時間の目標を達成することができます。サンプル・アプリケーションは回答時間の目標を達成する方法を示します。

短い照会の一貫性のある応答時間

一般的に応答時間が 1 秒未満の照会は、他のどんなワークロードがシステム上で実行されているかにかかわりなく、応答時間が比較的に一貫性のあるものとなります。サンプル・アプリケーションは照会実行時間のヒストグラムを使用して一貫性をモニターします。

ピーク要求の期間中にシステムを保護する

ワークロード管理ポリシー・フィーチャーは、いったんシステムに十分な負荷がかかると作業をキューイングして、ピーク要求のバースト中にシステムをキャパシティーの過負荷から保護します。

抽出、トランスフォーム、およびロード (ETL) のバッチ処理とユーザー照会を並行して使用可能にする

ワークロード管理フィーチャーを使用すれば、(例えば表へのデータのロードなどの) ETL ジョブを実行しながら、同時に照会を実行するユーザーのパフォーマンスへの影響を制御できます。

サンプル・アプリケーションを入手するには、developerWorks のワークロード管理のサンプルを参照してください。

シナリオ: ワークロード関連のシステム・スローダウンの調査

システム・スローダウン (例、一部のアプリケーションを完了するのに予想以上の時間がかかる、など) に気付き、その問題がワークロードの構成に関連しているかどうかが不確かな場合、表関数のデータを使用して問題を調べたり、必要に応じて訂正したりすることができます。

最初に、WLM_GET_SERVICE_SUBCLASS_STATS 表関数からのデータを使用して、複数のサービス・クラスとデータベース・メンバーにまたがるデータを集約する照会を作成します。すべてのデータベース・メンバーのすべてのサービス・クラスにおいてデータを収集することを指定するには、最初および 2 番目の引数を空ストリングに設定し、3 番目の引数を -2 に設定します。

照会は次のようになります。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(SUM(COORD_ACT_COMPLETED_TOTAL)),1,13) AS ACTSCOMPLETED,
       SUBSTR(CHAR(SUM(COORD_ACT_ABORTED_TOTAL)),1,11) AS ACTSABORTED,
       SUBSTR(CHAR(MAX(CONCURRENT_ACT_TOP)),1,6) AS ACTSHW,
       CAST(CASE WHEN SUM(COORD_ACT_COMPLETED_TOTAL) = 0 THEN 0
                 ELSE SUM(COORD_ACT_COMPLETED_TOTAL * COORD_ACT_LIFETIME_AVG)
                 / SUM(COORD_ACT_COMPLETED_TOTAL) END / 1000 AS DECIMAL(9,3))
       AS ACTAVGLIFETIME
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '', -2)) AS SCSTATS
GROUP BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME
```

```
SUPERCLASS_NAME     SUBCLASS_NAME      ACTSCOMPLETED ACTSABORTED ACTSHW ACTAVGLIFETIME
------------------- ------------------ ------------- ----------- ------ --------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 20            0           1               3.750
SUP1                SUB1               40            0           8               7.223
```

先述の例のデータでは、SUP1 サービス・スーパークラスの SUB1 サービス・サブクラスは、通常よりも多くのアクティビティーを同時に実行しています。さらに調査する場合、このサービス・クラスにマップするワークロードの統計を調べることもできます。照会は次のようになります。

```
SELECT SUBSTR(WLSTATS.WORKLOAD_NAME,1,22) AS WL_NAME,
       SUBSTR(CHAR(WLSTATS.MEMBER),1,4) AS MEMB,
       CONCURRENT_WLO_TOP AS WLO_HIGH_WTRMRK,
       CONCURRENT_WLO_ACT_TOP AS WLO_ACT_HIGH_WTRMRK
FROM TABLE(WLM_GET_WORKLOAD_STATS('', -2)) AS WLSTATS,
     TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', -2)) AS SCWLOS
WHERE WLSTATS.WORKLOAD_NAME = SCWLOS.WORKLOAD_NAME
AND SCWLOS.SERVICE_SUPERCLASS_NAME = 'SUP1'
AND SCWLOS.SERVICE_SUBCLASS_NAME = 'SUB1'
ORDER BY WL_NAME, MEMB;
```

```
WL_NAME                MEMB WLO_HIGH_WTRMRK WLO_ACT_HIGH_WTRMRK
---------------------- ---- --------------- -------------------
LYNNSALES                 0               2                   8
LYNNSALES                 1               0                   0
SYSDEFAULTUSERWORKLOAD    0               1                   1
SYSDEFAULTUSERWORKLOAD    1               0                   0
```

出力は、LYNNSALES ワークロードのアプリケーションが 8 個のアクティビティーを並行してサブミットしたことを示しています。各ワークロード・オカレンスのコーディネーター・アクティビティーの並行性を制限するには、しきい値の追加を考慮してください。

シナリオ: 子アクティビティー全体におけるアクティビティー・メトリックの集約

アクティビティー・イベント・モニターの各行には、特定のアクティビティーの実行を記述する情報およびモニター・メトリックが含まれています。アクティビティー情報に含まれるのは、**parent_uow_id** モニター・エレメントおよび **parent_activity_id** モニター・エレメントです。アクティビティーが他のアクティビティーの子で (ネストされていて)、かつエレメントによって親アクティビティーが特定される場合、これらのモニター・エレメントはゼロ以外の値です。

再帰的 SQL を使用して、すべての子アクティビティーにおける、アクティビティー・イベント・モニターのモニター・メトリックを、その親アクティビティーに集約することができます。親に集約されたメトリックを表示することは、ストアード・プロシージャーのどの部分が調整によって益を受ける可能性があるかを識別するのに役立ちます。例えば、ストアード・プロシージャー P1 がストアード・プロシージャー P2 および P3 を呼び出す場合、P1 の呼び出しにおける CPU 使用量の 90% が、ネストされたプロシージャー P3 のステートメントの処理中に使用されることが、集約されたメトリックによって示されるかもしれません。その結果、それに応じて重点的な調査を行うことができます。

ネストされたメトリックの集約をアクティビティー・イベント・モニターから得る方法として、以下の例を使用することができます。

注: この例では、ステートメントの終了文字として @ 文字が使用されています。次のコマンドの例で示されているとおり、以下のステートメントをファイル (例えば、test.clp) にコピーし、 db2 -td@ -f *filename* を使用して実行することができます。

```
db2 -td@ -f test.clp
```

この例では、以下の表およびストアード・プロシージャーが存在すると仮定します。

```
CREATE TABLE T1 (ONE INT)@

DROP PROCEDURE TEST.P1@
DROP PROCEDURE TEST.P2@
DROP PROCEDURE TEST.P3@
DROP PROCEDURE TEST.P4@

CREATE PROCEDURE TEST.P4()
LANGUAGE SQL
BEGIN
  INSERT INTO T1 VALUES(5);
  INSERT INTO T1 VALUES(6);
  INSERT INTO T1 VALUES(7);
END@

CREATE PROCEDURE TEST.P3()
LANGUAGE SQL
BEGIN
```

```
  DECLARE V INTEGER;

  INSERT INTO T1 VALUES(1);
  CALL TEST.P4();
  SELECT COUNT(*) INTO V FROM T1;
END@

CREATE PROCEDURE TEST.P2()
LANGUAGE SQL
BEGIN
  INSERT INTO T1 VALUES(2);
  INSERT INTO T1 VALUES(3);
END@

CREATE PROCEDURE TEST.P1()
LANGUAGE SQL
BEGIN
  CALL TEST.P3();
  CALL TEST.P2();
  INSERT INTO T1 VALUES(4);
END@
```

プロシージャー

1. アクティビティー・イベント・モニターを作成して、アクティビティー・キャプチャーを有効にします。この例では、WLM_SET_CONN_ENV プロシージャーを使用して、現在の接続に対してアクティビティー・キャプチャーが有効にされます。同じ接続を使用して TEST.P1 プロシージャーを実行してから、アクティビティー・イベント・モニターを無効にします。 TEST.P1 プロシージャーおよびその子アクティビティーすべてについてアクティビティー情報がキャプチャーされます。

   ```
   CREATE EVENT MONITOR A FOR ACTIVITIES WRITE TO TABLE@
   SET EVENT MONITOR A STATE 1@

   CALL WLM_SET_CONN_ENV(NULL, '<collectactdata>WITH DETAILS</collectactdata>
   <collectactpartition>ALL</collectactpartition>')@

   CALL TEST.P1()@

   SET EVENT MONITOR A STATE 0@

   CALL WLM_SET_CONN_ENV(NULL, '<collectactdata>NONE</collectactdata>')@
   ```

2. 以下の照会を実行して、親の ID 情報、個別の CPU 使用量、および集約 CPU 使用量と共にキャプチャーされたステートメントを表示します。すべての子アクティビティーにおける集約 CPU 使用量が再帰的に合計されます。この照会は、アクティビティー・イベント・モニターによってキャプチャーされる任意のモニター・メトリックをサポートするように、容易に拡張できます。

   注: 読みやすくするため、ここでは CPU 消費量のみを報告しています。

```
WITH ACT( APPL_ID,
          UOW_ID,
          ACTIVITY_ID,
          PARENT_UOW_ID,
          PARENT_ACTIVITY_ID,
          CPU )
  AS ( SELECT APPL_ID,
              UOW_ID,
              ACTIVITY_ID,
              MAX(PARENT_UOW_ID),
              MAX(PARENT_ACTIVITY_ID),
              SUM(METRICS.TOTAL_CPU_TIME)
```

```
        FROM ACTIVITY_A AS A,
            XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmls/prod/db2/mon'),
                '$actmetrics/activity_metrics' PASSING XMLPARSE(DOCUMENT A.DETAILS_XML) as "actmetrics"
                COLUMNS
                  TOTAL_CPU_TIME BIGINT PATH 'total_cpu_time' ) AS METRICS
        WHERE A.PARTIAL_RECORD = 0
        GROUP BY APPL_ID,
                UOW_ID,
                ACTIVITY_ID ),
    TMP( BASE_APPL_ID,
          BASE_UOW_ID,
          BASE_ACTIVITY_ID,
          APPL_ID,
          UOW_ID,
          ACTIVITY_ID,
          PARENT_UOW_ID,
          PARENT_ACTIVITY_ID,
          CPU,
          LEVEL )
  AS ( SELECT APPL_ID,
              UOW_ID,
              ACTIVITY_ID,
              APPL_ID,
              UOW_ID,
              ACTIVITY_ID,
              PARENT_UOW_ID,
              PARENT_ACTIVITY_ID,
              CPU,
              1
        FROM ACT
       UNION ALL
        SELECT T.BASE_APPL_ID,
              T.BASE_UOW_ID,
              T.BASE_ACTIVITY_ID,
              A.APPL_ID,
              A.UOW_ID,
              A.ACTIVITY_ID,
              A.PARENT_UOW_ID,
              A.PARENT_ACTIVITY_ID,
              A.CPU,
              T.LEVEL + 1
        FROM ACT AS A, TMP AS T
        WHERE A.APPL_ID = T.APPL_ID AND
              A.PARENT_UOW_ID = T.UOW_ID AND
              A.PARENT_ACTIVITY_ID = T.ACTIVITY_ID AND
              T.LEVEL < 128 ),
    AGG( APPL_ID,
          UOW_ID,
          ACTIVITY_ID,
          CPU )
  AS ( SELECT BASE_APPL_ID,
              BASE_UOW_ID,
              BASE_ACTIVITY_ID,
              SUM(CPU)
        FROM TMP
        GROUP BY BASE_APPL_ID,
                BASE_UOW_ID,
                BASE_ACTIVITY_ID )
SELECT
      A.UOW_ID,
      A.ACTIVITY_ID,
      A.PARENT_UOW_ID,
      A.PARENT_ACTIVITY_ID,
      A.CPU AS STMT_CPU,
      B.CPU AS AGG_CPU,
      SUBSTR(CONCAT(REPEAT('  ',C.STMT_NEST_LEVEL),
              C.STMT_TEXT),
```

```
             1, 30) AS STMT_TEXT
FROM ACT AS A,
    AGG AS B,
    ACTIVITYSTMT_A AS C,
    ACTIVITY_A AS D
WHERE A.APPL_ID = B.APPL_ID AND
      A.UOW_ID = B.UOW_ID AND
      A.ACTIVITY_ID = B.ACTIVITY_ID AND
      D.COORD_PARTITION_NUM = D.PARTITION_NUMBER AND
      A.APPL_ID = C.APPL_ID AND
      A.UOW_ID = C.UOW_ID AND
      A.ACTIVITY_ID = C.ACTIVITY_ID AND
      A.APPL_ID = D.APPL_ID AND
      A.UOW_ID = D.UOW_ID AND
      A.ACTIVITY_ID = D.ACTIVITY_ID AND
      D.PARTIAL_RECORD = 0
ORDER BY D.TIME_CREATED ASC@
```

この照会によって以下の出力が生成されます。STMT_CPU 列には、子アクティビティーによる CPU の使用は含めないで、ステートメントの CPU 使用量が報告されます。AGG_CPU 列には、アクティビティーとその子すべての集約 CPU 使用量が報告されます。

注: 照会があまりに速く実行されて測定できない場合、STMT_CPU および AGG_CPU はゼロになる可能性があります。

```
UOW_ID      ACTIVITY_ID          PARENT_UOW_ID PARENT_ACTIVITY_ID  STMT_CPU             AGG_CPU              STMT_TEXT
----------- -------------------- ------------- ------------------- -------------------- -------------------- --------------------------------
        576                    1             0                   0                 5353                84064 CALL TEST.P1()
        576                    2           576                   1                 7444                52043 CALL TEST.P3()
        576                    3           576                   2                 1869                 1869 INSERT INTO T1 VALUES(1)
        576                    4           576                   2                11727                26935 CALL TEST.P4()
        576                    5           576                   4                 2017                 2017 INSERT INTO T1 VALUES(5)
        576                    6           576                   4                 6602                 6602 INSERT INTO T1 VALUES(6)
        576                    7           576                   4                 6589                 6589 INSERT INTO T1 VALUES(7)
        576                    8           576                   2                15795                15795 SELECT COUNT(*) INTO :HV00
        576                    9           576                   1                11727                20314 CALL TEST.P2()
        576                   10           576                   9                 1941                 1941 INSERT INTO T1 VALUES(2)
        576                   11           576                   9                 6646                 6646 INSERT INTO T1 VALUES(3)
        576                   12           576                   1                 6354                 6354 INSERT INTO T1 VALUES(4)

  12 record(s) selected.
```

シナリオ: 完了に時間がかかり過ぎているアクティビティーの識別

ワークロード管理表関数を使用すると、データ・サーバー内の特定のアクティビティーを識別したり、必要に応じてアプリケーション全体を終了させることなくそのアクティビティーを取り消したりするタスクを単純化することができます。

完了に時間がかかり過ぎているアクティビティーの識別

以下は、長時間実行照会の識別の例です。 SalesReport アプリケーションを実行中の販売部のユーザーから、アプリケーションの完了に時間がかかり過ぎているとの苦情が報告されたとします。

アプリケーション・ハンドルを識別した後、
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用して、このアプリケーションで現在実行中のすべてのアクティビティーを調べます。例えば、アプリケーション・ハンドルが 1 の場合、照会は以下のようになります。

```
SELECT SUBSTR(CHAR(COORD_MEMBER),1,5) AS COORD,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
       SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
```

```
      SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
      SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PARACTID,
      SUBSTR(ACTIVITY_TYPE,1,8) AS ACTTYPE,
      SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
   FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(1, -2))
      AS WLOACTS
   ORDER BY MEMB, UOWID, ACTID

COORD MEMB UOWID ACTID PARUOWID PARACTID ACTTYPE NESTING
----- ---- ----- ----- -------- -------- -------- -------
    0    0     2     3        -        -     CALL       0
    0    0     2     5        2        3 READ_DML       1
```

該当するアクティビティーは、作業単位 ID が 2、アクティビティー ID が 5 であると識別されます。その後、WLM_GET_SERVICE_CLASS_AGENTS 表関数を以下のように使用して、このアクティビティーを扱うエージェントが何を実行しているのかを把握できます。

```
SELECT APPLICATION_HANDLE, UOW_ID, ACTIVITY_ID,
       SUBSTR(REQUEST_TYPE,1,8) AS REQUEST_TYPE,
       SUBSTR(EVENT_TYPE,1,8) AS EVENT_TYPE,
       SUBSTR(EVENT_OBJECT,1,8) AS EVENT_OBJECT
   FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', CAST(NULL AS BIGINT),-2))
      AS AGENTS
   WHERE APPLICATION_HANDLE = 1
   AND UOW_ID = 2
   AND ACTIVITY_ID = 5
```

例えば、このアクティビティーがキューに入れられる場合や、実行中、またはロックで待機中の場合があります。アクティビティーがキューに入れられた場合の結果は、次のようになります。

```
APPLICATION_HANDLE UOW_ID ACTIVITY_ID REQUEST_TYPE EVENT_TYPE EVENT_OBJECT
------------------ ------ ----------- ------------ ---------- ------------
                 1      2           5         OPEN       WAIT    WLM_QUEUE
```

アクティビティーが実行中の場合の結果は、次のようになります。

```
APPLICATION_HANDLE UOW_ID ACTIVITY_ID REQUEST_TYPE EVENT_TYPE EVENT_OBJECT
------------------ ------ ----------- ------------ ---------- ------------
                 1      2           5          OPEN   PROCESS      REQUEST
```

アクティビティーがロック待機中の場合の結果は、次のようになります。

```
APPLICATION_HANDLE UOW_ID ACTIVITY_ID REQUEST_TYPE EVENT_TYPE EVENT_OBJECT
------------------ ------ ----------- ------------ ---------- ------------
                 1      2           5         OPEN    ACQUIRE         LOCK
```

アクティビティーが現在行っている事柄を確認したなら、それに応じて以下のように続行できます。

- アクティビティーがキューに存在する場合、照会の実行時間が長すぎてもはやユーザーが結果に関心がない、または照会が消費しているリソース量が多すぎると判断するのであれば、その照会を取り消せます。
- 重要なアクティビティーがキューに存在する場合、現在実行中の作業のうちそれほど重要でない他のものを取り消すことを考慮するか (並行性を減らして重要なアクティビティーがキューに入れられたままにする)、また、その作業がハングしているわけではなく、待機しているだけであることが分かるとユーザーは納得する場合もあります。

- アクティビティーがロックを待機している場合、スナップショット・モニターを使用してアプリケーションが待機しているロックについて調査できます。
- アクティビティーが待機しているロックが優先度の低いアクティビティーによって保持されているのであれば、その優先度の低いアクティビティーを取り消すことを考慮してください。

アクティビティー 5 が実行中の DML ステートメントについて把握することが役立つ場合もあります。アクティブ・アクティビティー・イベント・モニターがあると仮定します。WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを実行すると、アクティビティー 5 が実行中の DML ステートメントに関する情報および他の情報をキャプチャーできます。WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを使用すると、ステートメント・イベント・モニターとは異なり、その時点で実行されているすべてのステートメントではなく特定の照会に関する情報をキャプチャーできます。また MON_GET_ACTIVITY_DETAILS を使用すると、ステートメント・テキストを取得できます。

アクティビティーを取り消す必要があると判断する場合、WLM_CANCEL_ACTIVITY ルーチンを以下のように使用すると、アクティビティーを発行したアプリケーションを終了させなくてもそのアクティビティーを取り消せます。

```
CALL WLM_CANCEL_ACTIVITY (1, 2, 5)
```

アクティビティーを発行したアプリケーションは SQL4725N エラーを受け取ります。負の SQL コードを処理するアプリケーションはこの SQL コードを処理できます。

ロック競合によるアクティビティー・ハングの識別

あるユーザーから時間がかかり過ぎているアプリケーションに関して苦情が報告されたとします。さらに、その長期実行アプリケーションのアプリケーション名または許可 ID のいずれかが分かっていると想定してください。この情報を用いて、**LIST APPLICATIONS** コマンドを使用すると、アプリケーション・ハンドルを取得できます。**LIST APPLICATIONS** コマンドによって戻されたアプリケーション・ハンドルが 2 であるとすると、WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用して、このアクティビティーを操作しているエージェントを判別できます。照会は次のようになります。

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(CHAR(AGENT_TID),1,9) AS AGENT_TID,
       SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
       SUBSTR(EVENT_OBJECT,1,11) AS EVENTOBJECT,
       SUBSTR(REQUEST_TYPE,1,7) AS REQTYPE,
       SUBSTR(CHAR(UOW_ID),1,6) AS UOW_ID,
       SUBSTR(CHAR(ACTIVITY_ID),1,6) AS ACT_ID
   FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', 2, -2)) AS SCDETAILS
   ORDER BY APPHANDLE, MEMB, AGENT_TID

APPHANDLE MEMB AGENT_TID AGENTTYPE   EVENTOBJECT  REQTYPE UOW_ID ACT_ID
--------- ---- --------- ----------- -----------  ------- ------ ------
2         0    1         COORDINATOR REQUEST      OPEN    2      1
2         1    3         SUBAGENT    LOCK         -       2      1
```

エージェント 1 がリモート応答を待機中であることを結果は示しています。同じアクティビティーを操作しているリモート・メンバー上のエージェントを見ると、そのエージェントがロックを取得するために待機中であることが EVENTOBJECT フィールドから分かります。

次のステップでは、ロックの所有者を判別します。この情報は以下の例に示されているように、モニター・スイッチをオンにして、スナップショット・モニター表関数を使用すると取得できます。

```
SELECT AGENT_ID AS WAITING_FOR_LOCK,
       SUBSTR(APPL_ID_HOLDING_LK,1,40) AS HOLDING_LOCK,
       CAST(LOCK_MODE_REQUESTED AS SMALLINT) AS WANTED,
       CAST(LOCK_MODE AS SMALLINT) AS HELD
   FROM TABLE(SNAPSHOT_LOCKWAIT('SAMPLE',-1)) AS SLW

WAITING_FOR_LOCK     HOLDING_LOCK                                WANTED HELD
-------------------- ------------------------------------------ ------ ------
                   2 *LOCAL.DB2.060131021547                         9      5
```

また以下の一連のコマンドを使用しても、ロック所有者を判別できます。

```
db2pd -db database alias -locks
db2pd -db database alias -transactions
```

長期実行アクティビティーを取り消す場合には、WLM_CANCEL_ACTIVITY プロシージャーを使用できます。長期実行アプリケーションを正常終了させる方がロックを所有しているアプリケーションを正常終了させるよりも重要な場合には、ロックを所有しているアプリケーションを強制終了させます。

シナリオ: 1 時間を超えてキューに入れられているアクティビティーをキャンセルする方法

ここに記載する例のスクリプトを使用すると、 1 時間を超えてキューに入れられているアクティビティーをキャンセルするプロシージャーを作成できます。また、そのキュー内アクティビティー・キャンセル・プロシージャーを、 DB2 管理用タスク・スケジューラーを使用して 10 分間隔で実行するようにスケジュールするスクリプトの例も記載しています。

このキュー内アクティビティー・キャンセル・プロシージャーは、キャンセルされたアクティビティーに関する情報のキャプチャー (アクティビティー・イベント・モニターがアクティブである場合)、およびキャンセルされたアクティビティーに関する小さな履歴表の保守も行います。これらの情報コンポーネントは、どちらもオプションです。スクリプト例のコメントに、コンポーネントが不要な場合にコメント化する場所を示しています。

例のプロシージャー内に含まれているステートメントは、それ自身もアクティビティーであり、しきい値制御の対象となります (使用するシステムで構成されているしきい値の内容によります)。 この例のキュー内アクティビティー・キャンセル・プロシージャーは、キューイングしきい値を適用させないサービス・クラスで実行するように検討してください。

1. 1 時間を超えてキューに入れられているアクティビティーをキャンセルするプロシージャーを作成する、次の例のスクリプトを、ユーザーが作成したファイル (例えば x.clp という名前のファイル) にコピーします。

```
-- Simple history table to track cancelled
-- activities

CREATE TABLE SAMPLE.CANCELED_ACTIVITIES(
   APPLICATION_HANDLE BIGINT,
   UOW_ID BIGINT,
   ACTIVITY_ID BIGINT )@

-- Cancel any activities that have been queued
-- for more than 1 hour

CREATE PROCEDURE SAMPLE.CANCEL_QUEUED_ACTIVITIES()
  LANGUAGE SQL
  BEGIN
     DECLARE APPHANDLE   BIGINT;
     DECLARE UOWID       BIGINT;
     DECLARE ACTIVITYID  BIGINT;
     DECLARE QUEUETIME   BIGINT;
     DECLARE AT_END      INT DEFAULT 0;

     DECLARE QUEUEDAPPS CURSOR WITH HOLD FOR SELECT APPLICATION_HANDLE,
         UOW_ID, ACTIVITY_ID
         FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(NULL,-2)) AS T
         WHERE ACTIVITY_STATE = 'QUEUED' AND LOCAL_START_TIME IS NULL;

     DECLARE QTIMECUR CURSOR FOR SELECT TIMESTAMPDIFF(8, CHAR
         (CURRENT TIMESTAMP - TIMESTAMP(VALUE)))
         FROM TABLE(WLM_GET_ACTIVITY_DETAILS(APPHANDLE ,
         UOWID ,  ACTIVITYID , -2)) AS T WHERE NAME = 'ENTRY_TIME';

     DECLARE CONTINUE HANDLER FOR NOT FOUND
        SET AT_END = 1;

     -- Ignore errors for activity not found and activity event
     -- monitor does not exist.
     DECLARE CONTINUE HANDLER FOR SQLSTATE '5U035', SQLSTATE '01H53'
        BEGIN
        END;

     -- Find all activities that are queued by WLM
     -- thresholds where (ACTIVITY_STATE = 'QUEUED')
     OPEN QUEUEDAPPS;
     FETCH QUEUEDAPPS INTO APPHANDLE, UOWID, ACTIVITYID;

     WHILE AT_END = 0 DO

     -- Now use activity entry time to estimate the time spend queued.
     -- Queuing occurs before an activity begins execution, so queue
     -- time is approximated using current time - entry time
        OPEN QTIMECUR;
        FETCH QTIMECUR INTO QUEUETIME;
        CLOSE QTIMECUR;

        IF ( QUEUETIME >= 1 ) THEN

     -- Optional: Insert a record into a table to record the
     -- cancellation of the statement (for monitoring purposes, to
     -- understand how many statements were cancelled). Modify this
     -- insert as required to capture more info such as the name of
     -- the application that submitted the cancelled query. Comment out
     -- these 2 lines if the monitoring is not important to you.
        INSERT INTO SAMPLE.CANCELED_ACTIVITIES VALUES ( APPHANDLE,
                       UOWID, ACTIVITYID );

     -- Optional: Send details about activity to any activity activities
     -- event monitor before cancelling. Comment out
     -- this line if you don't care about the details of the
```

```
    -- statements that were cancelled
       CALL WLM_CAPTURE_ACTIVITY_IN_PROGRESS( APPHANDLE, UOWID,
           ACTIVITYID );

    -- Cancel the activity
       CALL WLM_CANCEL_ACTIVITY( APPHANDLE, UOWID, ACTIVITYID );

    -- Explicit commit, required for the insert statement above. The
    -- admin task scheduler will not perform a commit. Comment out this
    -- line if the insert statement is removed.
       COMMIT;

       END IF;

       FETCH QUEUEDAPPS INTO APPHANDLE, UOWID, ACTIVITYID;

    END WHILE;

    CLOSE QUEUEDAPPS;

  END@
```

2. 次のコマンドによりスクリプト x.clp を実行して、キュー内アクティビティー・キャンセル・プロシージャーを作成します。

```
db2 -td@ -f x.clp
```

3. 次のコマンドを実行して、キュー内アクティビティー・キャンセル・プロシージャーを実行します。

```
db2 "call sample.cancel_queued_activities()"
```

   1 時間を超えてキューに入れられているアクティビティーが、キャンセルされます。

4. キュー内アクティビティー・キャンセル・プロシージャーを、 DB2 管理用タスク・スケジューラーを使用して 10 分間隔で実行するようにスケジュールするスクリプトの例を、次に示します。この例のスクリプトを、ユーザーが作成したファイル (例えば y.clp という名前のファイル) にコピーします。

```
----------------------------------------
-- Enable DB2 Admin Task Scheduler if
-- not already enabled.
----------------------------------------

!db2set DB2_ATS_ENABLE=YES@

----------------------------------------
-- Create SYSTOOLSPACE tablespace.
-- Enable if SYSTOOLSPACE does not already
-- exist on your database.
----------------------------------------

-- CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP MANAGED BY AUTOMATIC STORAGE
-- EXTENTSIZE 4@

----------------------------------------
-- Add a task to automatically cancel
-- activities that have been queued
-- for more than 1 hour. Task is scheduled
-- to run every 10 minutes. Adjust the
-- schedule as necessary using the
-- schedule input parameter (specified in
-- cron format).
----------------------------------------
```

```
CALL SYSPROC.ADMIN_TASK_ADD(
   'CANCEL ACTIVITIES QUEUED FOR MORE 1 HOUR',
   NULL,
   NULL,
   NULL,
   '*/10 * * * *',
   'SAMPLE',
   'CANCEL_QUEUED_ACTIVITIES',
   NULL,
   NULL,
   NULL )@
```

5. 次のコマンドによりスクリプト y.clp を実行して、キュー内アクティビティー・キャンセル・プロシージャーを 10 分間隔で実行するようにスケジュールします。

```
db2 -td@ -f y.clp
```

シナリオ: コストを低く見積もられた、実行時間の長いアクティビティーの識別

次の例は、ワーク・クラス、ワーク・アクション・セット、しきい値、およびアクティビティー・コレクションを使用して、見積もりコストは低いが実行時間の長いアクティビティーを識別できる方法を示しています。このような状況は、表と索引の統計が古いために、見積もりコスト (timeron 単位) が不正確であることを示している場合があります。

最初のステップは、ワーク・クラス・セット及びワーク・クラスを作成することです。これは、見積もりコストが低いアクティビティーを識別するために使用されます。以下に例を示します。

```
CREATE WORK CLASS SET WCS1
(WORK CLASS SMALLDML WORK TYPE DML FOR TIMERONCOST FROM 0 TO 500)
```

次に、データベース・ワーク・アクション・セット及び、ワーク・アクションを作成します。これは、アクティビティー合計時間しきい値を SMALLDML ワーク・クラスに適用します。しきい値アクションは CONTINUE で、しきい値に違反したアクティビティーが完了時にアクティビティー・イベント・モニターに送信されるように、次のようにして COLLECT ACTIVITY DATA オプションを指定します。

```
CREATE WORK ACTION SET WAS1 FOR DATABASE USING WORK CLASS SET WCS1
(WORK ACTION WA1 ON WORK CLASS SMALLDML WHEN ACTIVITYTOTALTIME > 15 MINUTES
COLLECT ACTIVITY DATA WITH DETAILS CONTINUE)
```

最後に、次のようにして、しきい値違反イベント・モニターとアクティビティー・イベント・モニターを作成し、活動化します。

```
CREATE EVENT MONITOR THVIOLATIONS FOR THRESHOLD VIOLATIONS WRITE TO TABLE
SET EVENT MONITOR THVIOLATIONS STATE 1

CREATE EVENT MONITOR DB2ACTIVITIES FOR ACTIVITIES WRITE TO TABLE
SET EVENT MONITOR DB2ACTIVITIES STATE 1
```

見積もりコストが 500 timeron 未満の DML アクティビティーが 15 分を超えて実行すると、THVIOLATIONS イベント・モニターに (合計時間しきい値に違反したことを示す) しきい値違反レコードが書き込まれ、DML アクティビティーの完了時にそれについての詳細が収集されて、DB2ACTIVITIES イベント・モニターに送信されます。DB2ACTIVITIES イベント・モニターのアクティビティーについて収集さ

れた情報を使用して、さらに調査を進めることができます。例えば、照会で EXPLAIN ステートメントを実行して、アクセス・プランを調べることができます。また、アクティビティーの収集時のシステム負荷とキューイングも考慮する必要があります。これは、存続時間が長いのは、システム・リソースが不足したり、アクティビティーがキューに入れられたためである可能性があるからです。存続時間が長いからといって、必ずしも統計が古いことを示しているとは限りません。

シナリオ: サービス・サブクラスで実行されているすべてのアクティビティーのキャンセル

特定のサービス・サブクラスで現在実行されているすべてのアクティビティーをキャンセルする際に使用できるストアード・プロシージャー例を以下に示します。

CANCELALL プロシージャーを作成するには、以下のステップを実行してください。

1. 次の CREATE PROCEDURE ステートメントをファイル (例えば、cancelall.ddl) にコピーします。

```
CREATE PROCEDURE CANCELALL ( IN INSCID BIGINT )
        SPECIFIC CANCELALL
        LANGUAGE SQL
BEGIN
   DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
   DECLARE APPHNDL BIGINT;
   DECLARE UOWID INTEGER;
   DECLARE ACTIVITYID INTEGER;

   DECLARE C1 CURSOR FOR  (SELECT APPLICATION_HANDLE,
        UOW_ID, ACTIVITY_ID
        FROM TABLE(SYSPROC.WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES
                                                ( NULL, -2 ))
        AS T WHERE T.SERVICE_CLASS_ID = INSCID);

   DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
     BEGIN
     END;
   OPEN C1;

   FETCH_LOOP:
     LOOP

   FETCH C1 INTO APPHNDL, UOWID, ACTIVITYID;

   IF (SQLSTATE <> '00000')
   THEN LEAVE FETCH_LOOP;
   END IF;

   CALL WLM_CANCEL_ACTIVITY( APPHNDL, UOWID, ACTIVITYID );

   END LOOP FETCH_LOOP;

END@
```

2. 次の CLP コマンドを実行します。

```
db2 -td@ -f cancelall.ddl
```

プロシージャーの作成後、次のステートメントを使用してこのプロシージャーを実行します (例えば、ID = 15 のサービス・サブクラス内にあるすべてのアクティビティーをキャンセルします)。

```
CALL CANCELALL( 15 )
```

注: CANCELALL プロシージャーは、入力として渡されるターゲットとは別のサービス・サブクラスで実行する必要があります。同じサービス・サブクラスで実行すると、このプロシージャー自体がキャンセルされることになります。

シナリオ: サービス・クラスにマップされるすべてのアプリケーションまたはサービス・クラス内でアクティビティーを実行しているすべてのアプリケーションの切断

特定のサービス・クラスにマップされるすべてのアプリケーション、または特定のサービス・クラスで現在アクティビティーを実行しているすべてのアプリケーションを切断 (強制終了) する際に使用できるストアード・プロシージャー例を以下に示します。

FORCEALLINSC プロシージャーを作成するには、以下のステップを実行してください。

1. 次の CREATE PROCEDURE ステートメントをファイル (例えば、forceall.ddl) にコピーします。

```
CREATE PROCEDURE FORCEALLINSC ( IN INSCID BIGINT )
        SPECIFIC FORCEALLINSC
        LANGUAGE SQL
BEGIN
   DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
   DECLARE APPHNDL BIGINT;
   DECLARE UOWID INTEGER;
   DECLARE ACTIVITYID INTEGER;

   DECLARE C1 CURSOR FOR  (SELECT APPLICATION_HANDLE
        FROM TABLE(SYSPROC.WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES
                                                    ( NULL, -2 ))
        AS T WHERE T.SERVICE_CLASS_ID = INSCID);


   DECLARE C2 CURSOR FOR  (SELECT APPLICATION_HANDLE
        FROM TABLE(SYSPROC.WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
                                                   (NULL,NULL, -2 ))
        AS T, SYSCAT.SERVICECLASSES AS S
        WHERE T.SERVICE_SUPERCLASS_NAME = S.PARENTSERVICECLASSNAME AND
        T.SERVICE_SUBCLASS_NAME = S.SERVICECLASSNAME AND
        S.SERVICECLASSID = INSCID);

   DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
     BEGIN
     END;

-- First force any applications that have an activity running in
-- the specified service class

   OPEN C1;

   FETCH_LOOP:
     LOOP

   FETCH C1 INTO APPHNDL;

   IF (SQLSTATE <> '00000')
   THEN LEAVE FETCH_LOOP;
   END IF;
```

```
    CALL ADMIN_CMD( 'FORCE APPLICATION (' || CHAR(APPHNDL) || ')' );

    END LOOP FETCH_LOOP;

  -- Now force any connections that are mapped to the service class, but which
  -- don't currently have any activities running

    OPEN C2;

    FETCH_LOOP2:
      LOOP

    FETCH C2 INTO APPHNDL;

    IF (SQLSTATE <> '00000')
    THEN LEAVE FETCH_LOOP2;
    END IF;

    CALL ADMIN_CMD( 'FORCE APPLICATION (' || CHAR(APPHNDL) || ')' );

    END LOOP FETCH_LOOP2;

  END@
```

2. 次の CLP コマンドを実行します。

```
db2 -td@ -f forceall.ddl
```

プロシージャーの作成後、次のステートメントを使用してこのプロシージャーを実行します (例として、ID = 15 の特定のサービス・クラスにマップされるすべてのアプリケーション、またはそのようなサービス・クラスでアクティビティーを現在実行しているすべてのアプリケーションを切断します)。

```
CALL FORCEALLINSC( 15 )
```

注: FORCEALLINSC プロシージャーは、入力として渡されるターゲットとは別のサービス・クラスで実行する必要があります。同じサービス・クラスで実行すると、このプロシージャー自体がキャンセルされることになります。

シナリオ: キャパシティー・プランニング・データが使用可能な場合の DB2 ワークロード管理構成の調整

キャパシティー・プランニングを実行した場合、ユーザーのタイプおよびその予測される応答時間に関する情報があるはずです。この情報を使用して、DB2 ワークロード管理構成の構築、構成の有効性の判別、および構成の調整を行えます。

キャパシティー・プランニングを実行したと仮定し、以下の表のデータは、作業タイプおよび応答時間目標に関するその実行結果を表しているとします。

表 63. キャパシティー・プランニングの結果

作業タイプ	アプリケーション	目標	重要度	予想されるスループット
受注	orderentryapp.exe	平均応答時間を < 1 秒にする	高	1 日につき 10 000 (挿入と更新の両方)
ビジネス・インテリジェンス照会	businessobjects.exe	平均応答時間を < 10 秒にする	高	1 日につき 100 の照会

表 63. キャパシティー・プランニングの結果 (続き)

作業タイプ	アプリケーション	目標	重要度	予想される スループット
バッチ処理	batchapp.exe	スループットを最大化する	低	1 日につき 5000 の更新
その他	その他すべてのアプリケーション	ベスト・エフォート	低	1 日につき 100 のアクティビティー

前述の表のデータに基づいて、3 つのサービス・クラス (ORDER_ENTRY_SC、BI_QUERIES_SC、BATCH_SC) およびそれらのサービス・クラスに作業を割り当てる 3 つのワークロード (ORDER_ENTRY_WL、BI_QUERIES_WL、BATCH_WL) を作成できます。サービス・クラスとワークロードの作成後、統計イベント・モニターを作成して、各サービス・クラスのアクティビティー存続期間ヒストグラムなどの集約アクティビティー情報を収集できます。以下の表は、各サービス・クラスのアクティビティーの日々の平均カウント (アクティビティー存続期間ヒストグラムから算出) を、キャパシティー・プランニング実行時に予想されていた量と比較したデータであると仮定します。

表 64. 日々のアクティビティー

サービス・クラス	1 日当たりの予想される アクティビティー	1 日当たりの実際のアクティビティー
ORDER_ENTRY_SC	10 000	9700
BI_QUERIES_SC	100	115
BATCH_SC	5000	5412
SYSDEFAULTUSERCLASS	100	85

測定されたデータは、キャパシティー・プランニングによる見積が正しかったことを示しています。以下の表は、平均アクティビティー存続期間 (アクティビティー存続期間ヒストグラムから取得) を、キャパシティー・プランニング時に判別された応答時間目標と比較したデータで、BI_QUERIES_SC サービス・クラス内のアクティビティーが応答時間の目標を満たしていないことを示しています。

表 65. 応答時間

サービス・クラス	応答時間目標	実際の平均存続期間
ORDER_ENTRY_SC	< 1 秒	0.8 秒
BI_QUERIES_SC	< 10 秒	30 秒
BATCH_SC		2 秒
SYSDEFAULTUSERCLASS		10 分

DB2 ワークロード管理を使用すると、ビジネス・インテリジェンス照会が応答時間目標を満たさないという問題を処理する際に、以下のような様々な方法を使用できます。

- 重要度の低いサービス・クラスの並行性を制限する
- オペレーティング・システムのワークロード・マネージャーによって、重要度の低いサービス・クラスにはプロセッサー・リソースの割り当てを少なくする

- サービス・クラスのエージェントおよび入出力プリフェッチャーの優先度を変更する
- 前述の 3 つの方法を組み合わせて使用する

ビジネス・インテリジェンス照会がその目標に達しない原因となっているリソースがプロセッサー時間であるとします。さらに、オペレーティング・システムのワークロード・マネージャーを使用して、SYSDEFAULTUSERCLASS サービス・クラスには他のサービス・クラスと比較して少ないプロセッサー・リソースを割り当てていると想定します。その場合、一定の日数に渡り、集約アクティビティー情報をキャプチャーして、CPU 割り振りに対して行った変更によって期待した結果が得られたかどうかを調べます。以下の表は別の比較を示しているデータで、応答時間目標と、オペレーティング・システムのワークロード・マネージャーに変更を加えた後にヒストグラムから算出された実際の平均存続期間を比較したものです。今度はすべてのサービス・クラスが応答時間の目標を達成し、プロセッサー時間の再割り振りを行ったので、SYSDEFAULTUSERCLASS サービス・クラス内のアクティビティーの応答時間は 2 倍になりました。

表 66. 再構成後の応答時間

サービス・クラス	応答時間目標	実際の平均存続期間
ORDER_ENTRY_SC	< 1 秒	0.6 秒
BI_QUERIES_SC	< 10 秒	9.5 秒
BATCH_SC		1.5 秒
SYSDEFAULTUSERCLASS		20 分

シナリオ: キャパシティー・プランニング情報が使用できない場合の DB2 ワークロード管理構成のチューニング

DB2 ワークロード管理ツールを使用すると、構成を設計するために使用するキャパシティー分析データがない場合であっても、ワークロード管理構成を設計、モニター、およびチューニングするのに役立ちます。

次のような状況を想定してください。システムのワークロードに関して十分な知識がないか、安定した実行結果を得るために必要なワークロードをまだ知らないために、作成するワークロードおよびサービス・クラスが最初は分からないとします。また、一部のアプリケーションには応答時間要件があることは知っているものの、スピードを重要視するアプリケーションなど、他の幾つほどのアプリケーションがリソースを得るために競っているかは知らないとします。ワークロード管理モニター機能を使用すると、これが判別できます。

モニター・データを基礎として使用して、DB2 ワークロード管理構成をセットアップするには、以下のようにします。

1. 重要であると分かっているアプリケーションを分類します。そうしたアプリケーションを取り分けて、それに見合うシステム・リソースの部分を割り振ってください。
2. その他のワークロードに関しては、ワークロード内で最大のアクティビティーに関する統計を収集します。そうしたアクティビティーは、アクティビティー単位でシステムに最大の影響を与えるものだからです。

3. ステップ 2 で収集したアクティビティー情報を分析します。
4. ワークロードのまだ分類されていない部分に関して、ステップ 1 から 3 までを繰り返します。未分類の作業を分類する必要はないと判断するまで、このステップを繰り返します。

以下のセクションでは、こうしたステップを実行する方法について記します。

ステップ 1: 重要なことが分かっているアプリケーションを取り分けて、それに見合うリソースの部分を指定する

BI1 と BI2 という 2 つの重要なビジネス・インテリジェンス・アプリケーションがあり、それらのアプリケーションの応答時間を最小化する必要があると想定します。これら 2 つのアプリケーション用のワークロードを作成し、システム・リソースを割り当てることのできる MOSTIMPORTANT というサービス・クラスにそれらのワークロードをマップできます。

AIX オペレーティング・システムでは、AIX ワークロード・マネージャーを使用して MOSTIMPORTANT というサービス・クラスを作成し、このサービス・クラスに保証されたリソースの集合を割り振ることができます。

DB2 データ・サーバーでは、必要なサービス・クラスおよびワークロードを以下のようにして作成します。

```
CREATE SERVICE CLASS MOSTIMPORTANT OUTBOUND CORRELATOR 'MOSTIMPORTANT'
CREATE WORKLOAD BI1WORKLOAD APPLNAME ('BI1') SERVICE CLASS MOSTIMPORTANT
CREATE WORKLOAD BI2WORKLOAD APPLNAME ('BI2') SERVICE CLASS MOSTIMPORTANT
```

この例での意図を明白にするため、認識済みアプリケーションについて把握した後でさえ、システム・ワークロードの大部分が使途不明であるとします。そのため、このワークロードに関して理解を深め、場合によっては制御する必要があります。

ステップ 2: その他の未分類のワークロードに関して、ワークロード内の最大のアクティビティーに関する統計を収集する

長時間実行アクティビティーは、短時間実行アクティビティーに比べてシステムに与える個々の影響が大きくなります。長時間実行アクティビティーは、長期間に渡ってシステム・リソースを占有するからです。しかし、長期間実行アクティビティーに関する情報を収集しても、短時間実行アクティビティーに関する情報を収集することと比較して、オーバーヘッドが大きくなるということはありません。このため、ワークロードで最も大きな比率を占める部分に関する情報を収集する最善の方法は、上位 30% の長時間実行アクティビティーに関する情報を最初に収集することです。

アクティビティー情報を収集するアクティビティー存続期間を最初に決定して、アクティビティー情報の収集を開始します。このタスクを単純化するには、収集する未分類のアクティビティーの部分 (30 % など) を選択してから、こうしたアクティビティーのアクティビティー存続期間ヒストグラムを調査できます。システムが統計を更新し、WLM_COLLECT_STATS プロシージャーを実行してその統計をアクティブ統計イベント・モニターに送信できるようにします。

以下の照会を使用して、SYSDEFAULTUSERCLASS サービス・クラスのアクティビティー存続期間ヒストグラムを表として取得します。その表は、各存続期間範囲に

分類される合計アクティビティーの比率を表しています。この照会は、データベースに複数のメンバーがないという前提で書かれています。

```
WITH TOTAL AS (
SELECT PARENTSERVICECLASSNAME,
       SERVICECLASSNAME,
       HIST.HISTOGRAM_TYPE,
       SUM(NUMBER_IN_BIN) AS NUMBER_IN_BIN
FROM HISTOGRAMBIN_DB2STATISTICS AS HIST,
     SYSCAT.SERVICECLASSES SC
WHERE
     HIST.SERVICE_CLASS_ID = SC.SERVICECLASSID
     AND HIST.TOP >= 0
     AND SC.PARENTSERVICECLASSNAME = 'SYSDEFAULTUSERCLASS'
     AND SC.SERVICECLASSNAME = 'SYSDEFAULTSUBCLASS'
     AND HIST.HISTOGRAM_TYPE = 'CoordActLifetime'
GROUP BY PARENTSERVICECLASSNAME, SERVICECLASSNAME, HISTOGRAM_TYPE)
SELECT CAST(CAST(TOP AS DOUBLE) / 60000 AS DECIMAL(14,3)) AS TOP_IN_MINUTES,
     CAST(100 * CAST(SUM(HIST.NUMBER_IN_BIN) AS DOUBLE) / TOTAL.NUMBER_IN_BIN AS DECIMAL(4,2))
     AS PERCENT_IN_BIN
FROM HISTOGRAMBIN_DB2STATISTICS AS HIST,
     SYSCAT.SERVICECLASSES SC,
     TOTAL
WHERE HIST.SERVICE_CLASS_ID = SC.SERVICECLASSID
      AND HIST.TOP >= 0
      AND TOTAL.NUMBER_IN_BIN > 0
      AND SC.PARENTSERVICECLASSNAME = 'SYSDEFAULTUSERCLASS'
      AND SC.SERVICECLASSNAME = 'SYSDEFAULTSUBCLASS'
      AND HIST.HISTOGRAM_TYPE = 'CoordActLifetime'
      AND TOTAL.PARENTSERVICECLASSNAME = SC.PARENTSERVICECLASSNAME
      AND TOTAL.SERVICECLASSNAME = SC.SERVICECLASSNAME
      AND TOTAL.HISTOGRAM_TYPE = HIST.HISTOGRAM_TYPE
GROUP BY TOP, SC.PARENTSERVICECLASSNAME, SC.SERVICECLASSNAME, HIST.HISTOGRAM_TYPE, TOTAL.NUMBER_IN_BIN;


TOP_IN_MINUTES   PERCENT_IN_BIN
---------------- --------------
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.001           0.00
           0.001           0.00
           0.002           0.00
           0.004           0.00
           0.006           0.00
           0.009           0.00
           0.014           0.00
           0.021           0.00
           0.033           0.00
           0.050           0.00
           0.077           0.00
           0.118           0.00
           0.180           0.00
           0.274           0.00
           0.419           0.00
           0.639           0.00
           0.975           0.00
           1.488           0.00
           2.269           0.00
           3.462           0.00
           5.280           0.00
```

```
  8.054          0.00
 12.286          0.00
 18.740          0.00
 28.584         10.00
 43.600         15.00
 66.505         45.00
101.442         23.00
154.731          5.00
236.015          2.00
360.000          0.00
```

以下の図は、前述の照会の結果をグラフとして作図したものです。

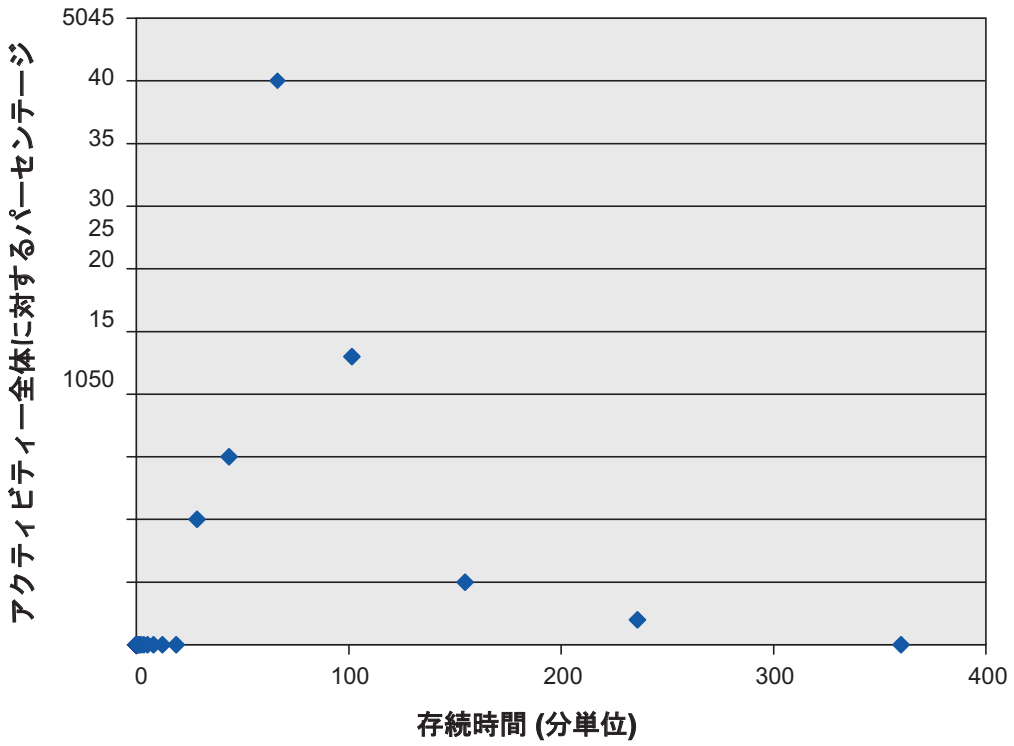


図 56. 未分類アクティビティーのアクティビティー存続期間ヒストグラム

この例では、アクティビティーの 30% が 101 分以上の存続期間範囲に該当します。こうしたアクティビティーの情報をキャプチャーするには、以下の例に示されているように CONTINUE オプションと COLLECT ACTIVITY DATA オプションを使用して、100 分のアクティビティー存続期間しきい値を作成します。このしきい値に違反すると、アクティビティー情報がアクティブ・アクティビティー・イベント・モニターに送信されます。

```
CREATE THRESHOLD COLLECTLONGESTRUNNING30PERCENT
FOR SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
ACTIVITIES ENFORCEMENT DATABASE ENABLE
WHEN ACTIVITYTOTALTIME > 100 MINUTES COLLECT ACTIVITY DATA CONTINUE
```

データを収集できるようにシステムを実行します。

この上位 30% の長時間実行アクティビティーに関する情報の収集に伴うオーバーヘッドを許容できるとすれば、データ収集を数時間または数日間にわたって続行できます。収集したデータを使用して、まだ未分類である、DML の上位 30% の長時間実行アクティビティーを生成しているユーザーとアプリケーションを判別できま

す。こうしたアクティビティーには、スピードを重視するものが含まれている場合があります。優先度の低いアプリケーションがかなりの数の大きなアクティビティーを実行しているなど、予想外なことが分かることもあります。データの収集と分析が終わったら、アクティビティー存続期間に対するしきい値を削除できます。

ステップ 3: 前のステップで収集したアクティビティーに関する情報を分析する

アクティビティーをサブミットしたアプリケーション毎に、前のステップで収集したアクティビティーに関する情報を分析できます。以下の照会を指定できます。

```
SELECT SUBSTR (APPL_NAME, 1,16) APPLICATION_NAME,
       AVG(TIMESTAMPDIFF(4, CHAR(TIME_COMPLETED - TIME_CREATED)))
       AS AVG_LIFETIME_MINUTES
       COUNT(*) AS ACTIVITY_COUNT
FROM ACTIVITY_DB2ACTIVITIES
GROUP BY APPL_NAME
ORDER BY APPL_NAME

APPLICATION_NAME AVG_LIFETIME_MINUTES ACTIVITY_COUNT
================ =================== ==============
MOSTLYSMALL1                      120             21
MOSTLYSMALL2                      110             15
UNIMPORTANTAPP                    150          10213
```

サブミットしたアプリケーション毎にアクティビティーを分析すると、上位 30% の長時間実行アクティビティーの大多数は UNIMPORTANTAPP アプリケーションによってサブミットされたことがわかります。ワークロードを使用してこのアプリケーションを他の未分類のアプリケーションから分離して、BESTEFFORT というサービス・クラスにマップします。このサービス・クラスは、他のすべてのアクティビティーがそのリソースの必要性を満たした場合にのみリソースを受け取ります。

前述の結果からすると、デフォルトのサービス・クラス内の残りのアプリケーションは大規模なアクティビティーをわずかしかサブミットしていないように思われます。長時間実行アクティビティーの収集を制限せずに、デフォルトのサービス・クラス内で実行されているアクティビティーを収集するプロセスを繰り返すことが役立つ場合もあります。

ステップ 4: ワークロード内に残っている未分類の作業を分類する必要がなくなるまで、ワークロードのまだ未分類の部分に関して、ステップ 1 から 3 までを繰り返す

この時点で、2 つの重要なアプリケーションが MOSTIMPORTANT サービス・クラスで実行されていて、重要でないアプリケーションは BESTEFFORT サービス・クラスで実行され、デフォルトのユーザー・サービス・クラスではさらに重要度の低い作業が実行されています。この状況では、このサービス・クラス内のすべてのアクティビティーに関する情報を収集するのはそれほど大変なことではありません。あるいは、作業をさらに細分化する必要がなければ、ここで終了することもできます。万一、残りのワークロードに想定外の事柄が含まれている場合に備えて、そうした残りのアクティビティーに関する情報を収集できます。以下のようにして、デフォルトのユーザー・サービス・クラスに COLLECT ACTIVITY DATA を設定して、アクティビティー・イベント・モニターを作成すると、このタスクを行えます。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
COLLECT ACTIVITY DATA ON COORDINATOR WITHOUT DETAILS
```

データを収集できるようにシステムを実行します。ステップ 3 の結果を分析できます。

```
SELECT SUBSTR (APPL_NAME,1,16) APPLICATION_NAME,
       AVG(TIMESTAMPDIFF(4, CHAR(TIME_COMPLETED - TIME_CREATED)))
       AS AVG_LIFETIME_MINUTES
       COUNT(*) AS ACTIVITY_COUNT
FROM ACTIVITY_DB2ACTIVITIES
GROUP BY APPL_NAME
ORDER BY APPL_NAME

APPLICATION_NAME AVG_LIFETIME_MINUTES ACTIVITY_COUNT
================ ==================== ==============
MOSTLYSMALL1                        5           1501
MOSTLYSMALL2                        7            124
ONLYSMALL                           2          10123
```

ONLYSMALL アプリケーションが未分類のアクティビティーの大多数を生成していることを、この結果は示しています。最も大規模なアクティビティーに関する情報を収集した際の結果にはこのアプリケーションが含まれていなかったので、データ収集の期間中には ONLYSMALL は大規模な照会を生成していなかったと考えることができます。

第 8 章 リファレンス

プロシージャーおよび表関数

WLM_CANCEL_ACTIVITY - アクティビティーのキャンセル

このプロシージャーは、指定されたアクティビティーをキャンセルします。キャンセルが行われる場合、キャンセルされたアクティビティーをサブミットしたアプリケーションにエラー・メッセージが戻されます。

構文

スキーマは SYSPROC です。

プロシージャー・パラメーター

application_handle
アクティビティーがキャンセルされるアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

uow_id
キャンセルされるアクティビティーの作業単位 ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

activity_id
キャンセルされる作業単位内のアクティビティーを一意的に識別するアクティビティー ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

管理者は WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用して、アクティビティーのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を検索できます。アプリケーション・ハンドル 1、作業単位 ID 2、およびアクティビティー ID 3 のアクティビティーをキャンセルするには、次のようにします。

```
CALL WLM_CANCEL_ACTIVITY(1, 2, 3)
```

使用上の注意

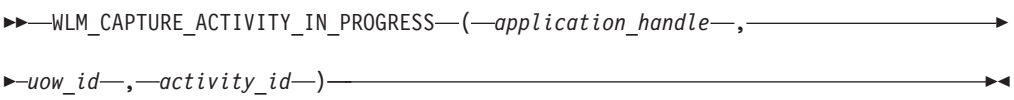
- アクティビティーが見つからない場合、SQLSTATE 5U035 の SQL4702N が戻されます。
- アクティビティーが正しい状態でない (初期化されていない) ためにキャンセルできない場合、SQLSTATE 5U016 の SQL4703N (理由コード 1) が戻されます。
- アクティビティーが正常にキャンセルされた場合、SQLSTATE 57014 の SQL4725N がキャンセルされたアプリケーションに戻されます。
- キャンセル時に、コーディネーターが別のアクティビティーの要求を処理しているかまたはアイドル状態である場合、アクティビティーは CANCEL_PENDING 状態になり、コーディネーターが次の要求を処理するとキャンセルされます。

WLM_CAPTURE_ACTIVITY_IN_PROGRESS - アクティビティー・イベント・モニターのアクティビティー情報の収集

WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーは、指定されたアクティビティーに関する情報を収集し、その情報をアクティブなアクティビティー・イベント・モニターに書き込みます。

子アクティビティーを持つアクティビティーにこのプロシージャーを適用する場合、プロシージャーはそれぞれの子アクティビティーのレコードを再帰的に生成します。この情報は、プロシージャーを呼び出すときに収集されて送信されます。プロシージャーは、親アクティビティーによる実行の完了を待機しません。イベント・モニター内のアクティビティーのレコードは部分レコードとしてマークが付けられます。

構文



スキーマは SYSPROC です。

プロシージャー・パラメーター

以下のパラメーターがすべて指定されないと、アクティビティーは検出されず、SQL4702N が SQLSTATE 5U035 とともに返されます。

application_handle
そのアクティビティー情報がキャプチャーされるアプリケーションのハンドルを指定する、タイプ BIGINT の入力引数。

uow_id
その情報がキャプチャーされるアクティビティーの作業単位 ID を指定する、タイプ INTEGER の入力引数。

activity_id
その情報がキャプチャーされる作業単位内のアクティビティーを一意的に識別するアクティビティー ID を指定する、タイプ INTEGER の入力引数。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

ストアード・プロシージャー MYSCHEMA.MYSLOWSTP の実行がいつもより遅いように感じる、とユーザーが苦情を言うとします。管理者はスローダウンの原因の調査に乗り出します。ストアード・プロシージャーを実行しながらの調査は実際的とは言えないので、管理者はストアード・プロシージャー・アクティビティーおよびその中にネストされたすべてのアクティビティーに関する情報をキャプチャーすることにします。

DB2ACTIVITIES という名前の DB2 アクティビティーのイベント・モニターがアクティブ化されています。管理者は WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 関数を使用して、このストアード・プロシージャーの呼び出しに関するアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を取得します。ここで、管理者がアクティビティーがアプリケーション・ハンドル 1、作業単位 ID 2、およびアクティビティー ID 3 で識別されていると想定し、WLM_CAPTURE_ACTIVITY_IN_PROGRESS への呼び出しを次のように発行できます。

```
CALL WLM_CAPTURE_ACTIVITY_IN_PROGRESS(1,2,3)
```

プロシージャーが完了した後、管理者は次の表関数を使用してアクティビティーが時間を要した場所を発見することができます。関数は、DB2ACTIVITIES イベント・モニターから情報を取得します。

```
CREATE FUNCTION SHOWCAPTUREDACTIVITY(APPHNDL BIGINT,
                                     UOWID INTEGER,
                                     ACTIVITYID INTEGER)
  RETURNS TABLE (UOW_ID INTEGER, ACTIVITY_ID INTEGER, STMT_TEXT VARCHAR(40),
    LIFE_TIME DOUBLE)
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
```

```
  RETURN WITH RAH (LEVEL, APPL_ID, PARENT_UOW_ID, PARENT_ACTIVITY_ID,
          UOW_ID, ACTIVITY_ID, STMT_TEXT, ACT_EXEC_TIME) AS
  (SELECT 1, ROOT.APPL_ID, ROOT.PARENT_UOW_ID,
          ROOT.PARENT_ACTIVITY_ID, ROOT.UOW_ID, ROOT.ACTIVITY_ID,
          ROOTSTMT.STMT_TEXT, ACT_EXEC_TIME
   FROM ACTIVITY_DB2ACTIVITIES ROOT, ACTIVITYSTMT_DB2ACTIVITIES ROOTSTMT
   WHERE ROOT.APPL_ID = ROOTSTMT.APPL_ID AND ROOT.AGENT_ID = APPHNDL
     AND ROOT.UOW_ID = ROOTSTMT.UOW_ID AND ROOT.UOW_ID = UOWID
     AND ROOT.ACTIVITY_ID = ROOTSTMT.ACTIVITY_ID AND ROOT.ACTIVITY_ID = ACTIVITYID
  UNION ALL
   SELECT PARENT.LEVEL +1, CHILD.APPL_ID, CHILD.PARENT_UOW_ID,
          CHILD.PARENT_ACTIVITY_ID, CHILD.UOW_ID,
          CHILD.ACTIVITY_ID, CHILDSTMT.STMT_TEXT, CHILD.ACT_EXEC_TIME
   FROM RAH PARENT, ACTIVITY_DB2ACTIVITIES CHILD,
        ACTIVITYSTMT_DB2ACTIVITIES CHILDSTMT
   WHERE PARENT.APPL_ID = CHILD.APPL_ID AND
         CHILD.APPL_ID = CHILDSTMT.APPL_ID AND
         PARENT.UOW_ID = CHILD.PARENT_UOW_ID AND
         CHILD.UOW_ID = CHILDSTMT.UOW_ID AND
         PARENT.ACTIVITY_ID = CHILD.PARENT_ACTIVITY_ID AND
         CHILD.ACTIVITY_ID = CHILDSTMT.ACTIVITY_ID AND
         PARENT.LEVEL < 64
  )
SELECT UOW_ID, ACTIVITY_ID, SUBSTR(STMT_TEXT,1,40),
               ACT_EXEC_TIME AS
               LIFE_TIME
FROM RAH
```

以下のサンプル照会では、表関数を使用します。

```
  SELECT * FROM TABLE(SHOWCAPTUREDACTIVITY(1, 2, 3))
    AS ACTS ORDER BY UOW_ID, ACTIVITY_ID
```

使用上の注意

アクティブなアクティビティー・イベント・モニターがない場合、SQLSTATE 01H53 の SQL1633W が戻されます。

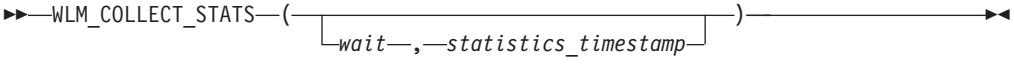
アクティビティー情報は、アクティビティーのコーディネーター・メンバーでのみ収集されます。

WLM_COLLECT_STATS - ワークロード管理統計の収集およびリセット

WLM_COLLECT_STATS プロシージャーは、サービス・クラス、ワークロード、作業クラス、およびしきい値キューの統計を収集し、統計イベント・モニターに書き込みます。また、このプロシージャーは、サービス・クラス、ワークロード、作業クラス、およびしきい値キューの統計のリセットも行います。

アクティブな統計イベント・モニターがない場合、プロシージャーは統計のリセットのみを行います。

構文



スキーマは SYSPROC です。

プロシージャー・パラメーター

wait
このプロシージャーが統計収集とリセットを開始した直後に戻るかどうかを指定する、タイプ CHAR のオプション入力引数。 'Y' を指定した場合、プロシージャーは、すべての統計が統計イベント・モニター表に書き込まれてフラッシュされるまで戻りません。それ以外の場合、プロシージャーは統計収集およびリセットを開始した直後に戻ります。

statistics_timestamp
統計収集の開始のタイム・スタンプ値を戻すタイプ TIMESTAMP のオプション出力引数。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

例 1: WLM_COLLECT_STATS を呼び出して統計収集とリセットを開始します。

```
CALL WLM_COLLECT_STATS()
```

以下はこの照会の出力例です。

```
Return Status = 0
```

例 2: WLM_COLLECT_STATS を呼び出して、統計を収集およびリセットしますが、データが統計イベント・モニター表に書き込まれるまで戻らないようにします。

```
CALL WLM_COLLECT_STATS('Y', ::collect_timestamp)
```

以下はこの照会の出力例です。

```
Return Status = 0
```

例 3: 別の呼び出しが進行中に、WLM_COLLECT_STATS を呼び出して、統計を収集およびリセットする。

```
CALL WLM_COLLECT_STATS()
```

以下はこの照会の出力例です。

```
SQL1632W The collect and reset statistics request was ignored because
another collect and reset statistics request is already in progress.
```

使用上の注意

WLM_COLLECT_STATS プロシージャーは、**wlm_collect_int** データベース構成パラメーターで定義された間隔で自動的に行われるものと同じ収集操作 (アクティブな統計イベント・モニターへの統計の送信) およびリセット操作を実行します。

別の収集およびリセット要求の進行中 (例えばプロシージャーの別の呼び出しの実行中または自動収集の発生中) にプロシージャーを呼び出すと、SQL1632W が SQLSTATE 01H53 とともに返され、新しい要求は無視されます。

非同期モードでは、WLM_COLLECT_STATS プロシージャーは、収集およびリセット・プロセスのみ開始します。このプロシージャーは、すべての統計がアクティブな統計イベント・モニターに書き込まれる前に呼び出し元に戻る場合があります。統計の収集およびリセットが発生する頻度に応じて、WLM_COLLECT_STATS プロシージャーの呼び出し (これ自体がアクティビティー) は統計において、前の収集間隔または直前に開始された新規の収集間隔のいずれかでカウントされます。

同期モードでは、WLM_COLLECT_STATS プロシージャーは、統計収集が完了し、すべての統計がアクティブな統計イベント・モニターの表に書き込まれるまで戻りません。統計収集が開始した時のタイム・スタンプは、*statistics_timestamp* 出力パラメーターを介して戻されます。

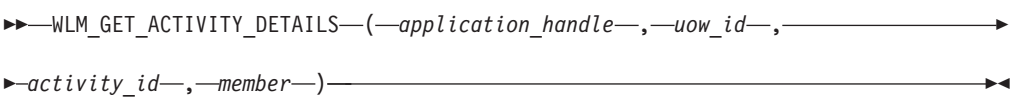
WLM_GET_ACTIVITY_DETAILS - 特定のアクティビティーに関する詳細情報を戻す

この関数は、そのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID によって識別される特定のアクティビティーに関する詳細情報を戻します。この情報には、アクティビティーが違反したしきい値に関する詳細が含まれます。

注: この表関数は使用すべきではなく、 に置き換えられました。 MON_GET_ACTIVITY_DETAILS 表関数.

この関数は、1 つ以上のサービス・サブクラスの基本統計を戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

application_handle
有効なアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL の場合、行はこの関数から戻されません。引数が NULL の場合、SQL171N エラーが戻されます。

uow_id
アプリケーション内で固有の有効な作業単位 ID を指定する、タイプ INTEGER

の入力引数。引数が NULL の場合、行はこの関数から戻されません。引数が NULL の場合、SQL171N エラーが戻されます。

activity_id
作業単位内で固有の有効なアクティビティー ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、行はこの関数から戻されません。引数が NULL の場合、SQL171N エラーが戻されます。

member
この関数を呼び出すときに現在接続されているデータベースと同じインスタンス内の有効なメンバー番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・メンバーの場合は -1、すべてのデータベース・メンバーの場合は -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

個々のアクティビティーに関する詳細情報は、WLM_GET_ACTIVITY_DETAILS 表関数を使用して取得できます。この表関数は、メンバーごとにアクティビティー情報を、名前と値のペアで戻します。この例は、アプリケーション・ハンドル 1、作業単位 ID 1、アクティビティー ID 5 で識別されるアクティビティーのメンバーごとに、名前と値のペアの 11 個のメンバー・サブセットのみを示すことに限定しています。名前と値のペアの完全なリストについては、418 ページの表 68および 420 ページの表 69を参照してください。

```
SELECT SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(NAME, 1, 20) AS NAME,
       SUBSTR(VALUE, 1, 30) AS VALUE
FROM TABLE(WLM_GET_ACTIVITY_DETAILS(1, 1, 5, -2)) AS ACTDETAIL
WHERE NAME IN ('APPLICATION_HANDLE',
               'COORD_PARTITION_NUM',
               'LOCAL_START_TIME',
               'UOW_ID',
               'ACTIVITY_ID',
               'PARENT_UOW_ID',
               'PARENT_ACTIVITY_ID',
               'ACTIVITY_TYPE',
               'NESTING_LEVEL',
               'INVOCATION_ID',
               'ROUTINE_ID')
ORDER BY PART
```

以下はこの照会の出力例です。

```
PART NAME                 VALUE
---- -------------------- ------------------------------
0    APPLICATION_HANDLE   1
0    COORD_PARTITION_NUM  0
0    LOCAL_START_TIME     2005-11-25-18.52.49.343000
0    UOW_ID               1
0    ACTIVITY_ID          5
0    PARENT_UOW_ID        1
0    PARENT_ACTIVITY_ID   3
0    ACTIVITY_TYPE        READ_DML
0    NESTING_LEVEL        0
0    INVOCATION_ID        1
0    ROUTINE_ID           0
1    APPLICATION_HANDLE   1
1    COORD_PARTITION_NUM  0
1    LOCAL_START_TIME     2005-11-25-18.52.49.598000
1    UOW_ID               1
1    ACTIVITY_ID          5
1    PARENT_UOW_ID
1    PARENT_ACTIVITY_ID
1    ACTIVITY_TYPE        READ_DML
1    NESTING_LEVEL        0
1    INVOCATION_ID        1
1    ROUTINE_ID           0
```

使用上の注意

ACTIVITY_STATE が QUEUED である場合、コーディネーター・アクティビティーがカタログ・メンバーに対する RPC を行ってしきい値チケットを取得したが、まだ応答を受け取っていないことを意味します。この状態が表示されることは、アクティビティーが WLM によってキューに入れられていることを示すか、または短期間にわたって、アクティビティーがそのチケットを取得する処理中であることを示すことがあります。アクティビティーが本当にキューに入れられているかどうかについてもっと正確な実態を把握するために、どのエージェントが (WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用して) アクティビティーで作業しているかを判別し、カタログ・メンバーにあるこのエージェントの event_object の値が WLM_QUEUE であるかどうかを検出することができます。

戻される情報

表 67. WLM_GET_ACTIVITY_DETAILS について戻される情報

列名	データ・タイプ	説明
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
NAME	VARCHAR(256)	エレメント名。考えられる値については、表 68および 420 ページの表 69を参照してください。
VALUE	VARCHAR(1024)	エレメントの値。考えられる値については、表 68および 420 ページの表 69を参照してください。

表 68. 戻されるエレメント

エレメント名	説明
ACTIVITY_ID	activity_id - アクティビティー ID モニター・エレメント
ACTIVITY_STATE	activity_state - アクティビティーの状態モニター・エレメント

表 68. 戻されるエレメント (続き)

エレメント名	説明
ACTIVITY_TYPE	activity_type - アクティビティー・タイプ・モニター・エレメント
APPLICATION_HANDLE	application_handle - アプリケーション・ハンドル・モニター・エレメント
COORD_PARTITION_NUM	coord_partition_num - コーディネーター・パーティション番号モニター・エレメント
DATABASE_WORK_ACTION_SET_ID	このアクティビティーがデータベースに適用されている作業アクション・セットにマップされている場合、この列にはその作業アクション・セットの ID が入っています。アクティビティーが、データベースに適用されている作業アクション・セットにマップされていない場合、この列には 0 が入っています。
DATABASE_WORK_CLASS_ID	このアクティビティーがデータベースに適用されている作業アクション・セットにマップされている場合、この列にはこのアクティビティーの作業クラスの ID が入っています。アクティビティーが、データベースに適用されている作業アクション・セットにマップされていない場合、この列には 0 が入っています。
EFFECTIVE_ISOLATION	effective_isolation - 有効な分離モニター・エレメント
EFFECTIVE_LOCK_TIMEOUT	effective_lock_timeout - 有効なロック・タイムアウト・モニター・エレメント
EFFECTIVE_QUERY_DEGREE	effective_query_degree - 有効な照会の度合いモニター・エレメント
ENTRY_TIME	entry_time - エントリー時間モニター・エレメント
INVOCATION_ID	invocation_id - 呼び出し ID モニター・エレメント
LAST_REFERENCE_TIME	last_reference_time - 最終参照時刻モニター・エレメント
LOCAL_START_TIME	local_start_time - ローカル開始時刻モニター・エレメント
NESTING_LEVEL	nesting_level - ネスティング・レベル・モニター・エレメント
PACKAGE_NAME	package_name - パッケージ名モニター・エレメント
PACKAGE_SCHEMA	package_schema - パッケージ・スキーマ・モニター・エレメント
PACKAGE_VERSION_ID	package_version_id - パッケージ・バージョン・モニター・エレメント
PARENT_ACTIVITY_ID	parent_activity_id - 親アクティビティー ID モニター・エレメント
PARENT_UOW_ID	アプリケーション内の固有の作業単位 ID。このアクティビティーの親アクティビティーが開始された元の作業単位を表します。アクティビティーに親アクティビティーがない場合、またはそれがリモート・メンバーにある場合は、空ストリングを戻します。
QUERY_COST_ESTIMATE	query_cost_estimate - 照会コストの見積もりモニター・エレメント
ROUTINE_ID	routine_id - ルーチン ID モニター・エレメント

表 68. 戻されるエレメント (続き)

エレメント名	説明
ROWS_FETCHED	rows_fetched - フェッチ行数モニター・エレメント
ROWS_MODIFIED	rows_modified - 変更された行数モニター・エレメント
SECTION_NUMBER	section_number - セクション番号モニター・エレメント
SERVICE_CLASS_ID	service_class_id - サービス・クラス ID モニター・エレメント
SERVICE_CLASS_WORK_ACTION_SET_ID	このアクティビティーがサービス・クラスに適用されている作業アクション・セットにマップされている場合、この列にはその作業アクション・セットの ID が入っています。アクティビティーがサービス・クラスに適用されている作業アクション・セットにマップされていない場合、この列には 0 が入っています。
SERVICE_CLASS_WORK_CLASS_ID	このアクティビティーがサービス・クラスに適用されている作業アクション・セットにマップされている場合、この列にはこのアクティビティーの作業クラスの ID が入っています。アクティビティーが、サービス・クラスに適用されているワーク・アクション・セットにマップされていない場合、この列には 0 が入っています。
STMT_PKG_CACHE_ID	stmt_pkgcache_id - ステートメント・パッケージ・キャッシュ ID モニター・エレメント
STMT_TEXT	stmt_text - SQL ステートメント・テキスト・モニター・エレメント
SYSTEM_CPU_TIME	system_cpu_time - システム CPU 時間モニター・エレメント
UOW_ID	uow_id - 作業単位 ID モニター・エレメント
USER_CPU_TIME	user_cpu_time - ユーザー CPU 時間モニター・エレメント
UTILITY_ID	utility_id - ユーティリティー ID モニター・エレメント

重要: WLM_GET_ACTIVITY_DETAILS 表関数は、現在アクティビティーに適用されているしきい値のみを示します。

以下のエレメントは、対応するしきい値がアクティビティーに適用される場合にのみ戻されます。

表 69. 適用される場合に戻されるエレメント

エレメント名	説明
ACTIVITYTOTALTIME_THRESHOLD_ID	activitytotaltime_threshold_id - アクティビティー合計時間しきい値 ID モニター・エレメント
ACTIVITYTOTALTIME_THRESHOLD_VALUE	activitytotaltime_threshold_value - アクティビティー合計時間しきい値モニター・エレメント
ACTIVITYTOTALTIME_THRESHOLD_VIOLATED	activitytotaltime_threshold_violated - アクティビティー合計時間しきい値の違反モニター・エレメント
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_ID	concurrentdbcoordactivities_db_threshold_id - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値 ID モニター・エレメント

表 69. 適用される場合に戻されるエレメント (続き)

エレメント名	説明
CONCURRENTDBCOORDACTIVITIES_DB_ THRESHOLD_QUEUED	concurrentdbcoordactivities_db_threshold_queued - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値によるキュー待機モニター・エレメント
CONCURRENTDBCOORDACTIVITIES_DB_ THRESHOLD_VALUE	concurrentdbcoordactivities_db_threshold_value - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値モニター・エレメント
CONCURRENTDBCOORDACTIVITIES_DB_ THRESHOLD_VIOLATED	concurrentdbcoordactivities_db_threshold_violated - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値の違反モニター・エレメント
CONCURRENTDBCOORDACTIVITIES_ SUBCLASS_THRESHOLD_ID	concurrentdbcoordactivities_subclass_threshold_id - 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 ID モニター・エレメント
CONCURRENTDBCOORDACTIVITIES_ SUBCLASS_THRESHOLD_QUEUED	「Yes」は、アクティビティーが CONCURRENTDBCOORDACTIVITIES_SUBCLASS しきい値によってキューに入れられたことを示します。「No」は、アクティビティーがキューに入れられなかったことを示します。
CONCURRENTDBCOORDACTIVITIES_ SUBCLASS_THRESHOLD_VALUE	concurrentdbcoordactivities_subclass_threshold_value - 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 : モニター・エレメント
CONCURRENTDBCOORDACTIVITIES_ SUBCLASS_THRESHOLD_VIOLATED	「Yes」は、アクティビティーが CONCURRENTDBCOORDACTIVITIES_SUBCLASS しきい値に違反したことを示します。「No」は、アクティビティーがまだしきい値に違反していないことを示します。
CONCURRENTDBCOORDACTIVITIES_ SUPERCLASS_THRESHOLD_ID	アクティビティーに適用された CONCURRENTDBCOORDACTIVITIES _SUPERCLASS しきい値の ID。
CONCURRENTDBCOORDACTIVITIES_ SUPERCLASS_THRESHOLD_QUEUED	「Yes」は、アクティビティーが CONCURRENTDBCOORDACTIVITIES _SUPERCLASS しきい値によってキューに入れられたことを示します。「No」は、 アクティビティーがキューに入れられなかったことを示します。
CONCURRENTDBCOORDACTIVITIES_ SUPERCLASS_THRESHOLD_VALUE	アクティビティーに適用された CONCURRENTDBCOORDACTIVITIES _SUPERCLASS しきい値の 上限。
CONCURRENTDBCOORDACTIVITIES_ SUPERCLASS_THRESHOLD_VIOLATED	「Yes」は、アクティビティーが CONCURRENTDBCOORDACTIVITIES _SUPERCLASS しきい値に違反したことを示します。「No」は、 アクティビティーがまだしきい値に違反していないことを示します。

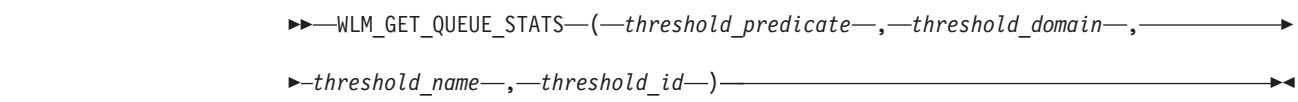
表 69. 適用される場合に戻されるエレメント (続き)

エレメント名	説明
CONCURRENTDBCOORDACTIVITIES_ WORK_ACTION_SET_THRESHOLD_ID	アクティビティーに適用された CONCURRENTDBCOORDACTIVITIES_WORK _ACTION_SET しきい値の ID。
CONCURRENTDBCOORDACTIVITIES_ WORK_ACTION_SET_THRESHOLD_QUEUED	「Yes」は、アクティビティーが CONCURRENTDBCOORDACTIVITIES _WORK_ACTION_SET しきい値によってキューに入れられたことを示します。「No」は、アクティビティーがキューに 入れられなかったことを示します。
CONCURRENTDBCOORDACTIVITIES_ WORK_ACTION_SET_THRESHOLD_VALUE	アクティビティーに適用された CONCURRENTDBCOORDACTIVITIES_WORK _ACTION_SET しきい値の 上限。
CONCURRENTDBCOORDACTIVITIES_ WORK_ACTION_SET_THRESHOLD_VIOLATED	「Yes」は、アクティビティーが CONCURRENTDBCOORDACTIVITIES_WORK _ACTION_SET しきい値に違反したことを示します。「No」は、 アクティビティーがまだしきい値に違反していないことを示します。
CONCURRENTWORKLOADACTIVITIES _THRESHOLD_ID	アクティビティーに適用された CONCURRENTWORKLOADACTIVITIES しきい値の ID。
CONCURRENTWORKLOADACTIVITIES _THRESHOLD_VALUE	アクティビティーに適用された CONCURRENTWORKLOADACTIVITIES しきい値の上限。
CONCURRENTWORKLOADACTIVITIES _THRESHOLD_VIOLATED	「Yes」は、アクティビティーが CONCURRENTWORKLOADACTIVITIES しきい値に違反したことを示します。「No」は、アクティビティーがまだしきい値に違反していないことを示します。
ESTIMATEDSQLCOST_THRESHOLD_ ID	estimatedsqlcost_threshold_id - 見積もり SQL コストしきい値 ID モニター・エレメント
ESTIMATEDSQLCOST_THRESHOLD_ VALUE	estimatedsqlcost_threshold_value - 見積もり SQL コストしきい値モニター・エレメント
ESTIMATEDSQLCOST_THRESHOLD_ VIOLATED	estimatedsqlcost_threshold_violated - 見積もり SQL コストしきい値の違反モニター・エレメント
SQLROWSRETURNED_THRESHOLD_ ID	sqlrowsreturned_threshold_id - 戻される SQL 読み取り行数しきい値 ID モニター・エレメント
SQLROWSRETURNED_THRESHOLD_ VALUE	sqlrowsreturned_threshold_value - 戻される SQL 読み取り行数しきい値モニター・エレメント
SQLROWSRETURNED_THRESHOLD_ VIOLATED	sqlrowsreturned_threshold_violated - 戻される SQL 読み取り行数しきい値の違反モニター・エレメント
SQLTEMPSPACE_THRESHOLD_ID	sqltempspace_threshold_id - SQL 一時スペースしきい値 ID モニター・エレメント
SQLTEMPSPACE_THRESHOLD_ VALUE	sqltempspace_threshold_value - SQL 一時スペースしきい値モニター・エレメント
SQLTEMPSPACE_THRESHOLD_ VIOLATED	sqltempspace_threshold_violated - SQL 一時スペースしきい値の違反モニター・エレメント

WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す

WLM_GET_QUEUE_STATS 関数は、すべてのアクティブ・メンバー上の 1 つ以上のしきい値キューの基本統計を戻します。この関数は、しきい値キューごとに 1 行の統計を戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

threshold_predicate
しきい値述部を指定する、タイプ VARCHAR(27) の入力引数。可能な値は、以下のとおりです。

CONCDBC
並行データベース・コーディネーター・アクティビティーしきい値

DBCONN
データベース・メンバー接続合計しきい値

SCCONN
サービス・クラス・メンバー接続合計しきい値

引数が NULL または空ストリングである場合、他の基準を満たすすべてのしきい値についてデータが戻されます。

threshold_predicate の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDPREDICATE 列の値と一致します。

threshold_domain
しきい値ドメインを指定する、タイプ VARCHAR(18) の入力引数。可能な値は、以下のとおりです。

DB データベース

SB サービス・サブクラス

SP サービス・スーパークラス

WA 作業アクション・セット

引数が NULL または空ストリングである場合、他の基準を満たすすべてのしきい値についてデータが戻されます。

threshold_domain の値は、SYSCAT.THRESHOLDS ビューの DOMAIN 列の値と一致します。

threshold_name
しきい値の名前を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、他の基準を満たすすべてのしきい値についてデータが戻されます。 *threshold_name* の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDNAME 列の値と一致します。

threshold_id
しきい値 ID を指定する、タイプ INTEGER の入力引数。引数が NULL または -1 である場合、他の基準を満たすすべてのしきい値についてデータが戻されます。 *threshold_id* の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDID 列の値と一致します。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

以下の照会は、システム上のすべてのキューの基本統計を、すべてのメンバーにわたって表示します。

```
SELECT substr(THRESHOLD_NAME, 1, 6) THRESHNAME,
     THRESHOLD_PREDICATE,
     THRESHOLD_DOMAIN,
     MEMBER MEMB,
     QUEUE_SIZE_TOP,
     QUEUE_TIME_TOTAL,
     QUEUE_ASSIGNMENTS_TOTAL QUEUE_ASSIGN
FROM table(WLM_GET_QUEUE_STATS('', '', '', -1)) as QSTATS
```

出力例を以下に示します。

```
THRESHNAME THRESHOLD_PREDICATE         THRESHOLD_DOMAIN   ...
---------- --------------------------- ------------------ ...
LIMIT1     CONCDBC                     DB                 ...
LIMIT2     SCCONN                      SP                 ...
LIMIT3     DBCONN                      DB                 ...

... MEMB QUEUE_SIZE_TOP QUEUE_TIME_TOTAL QUEUE_ASSIGN
... ---- -------------- ---------------- ------------
... 0             12          1238540          734
... 0              4           741249           24
... 0              7           412785          128
```

使用上の注意

この関数は、(1 つのメンバー上の) キュー全体や (1 つ以上のキューの) メンバー全体のデータ集約は行いません。ただし、上の例で示された SQL 照会を使用すると、データを集約することができます。

戻される情報

表 70. WLM_GET_QUEUE_STATS について戻される情報

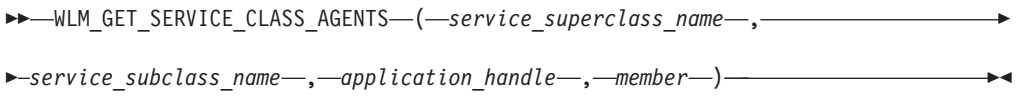
列名	データ・タイプ	説明
THRESHOLD_PREDICATE	VARCHAR(27)	threshold_predicate - しきい値述部モニター・エレメント
THRESHOLD_DOMAIN	VARCHAR(18)	threshold_domain - しきい値ドメイン・モニター・エレメント
THRESHOLD_NAME	VARCHAR(128)	threshold_name - しきい値名モニター・エレメント
THRESHOLD_ID	INTEGER	thresholdid - しきい値 ID モニター・エレメント
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name - サービス・スーパークラス名モニター・エレメント
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name - サービス・サブクラス名モニター・エレメント
WORK_ACTION_SET_NAME	VARCHAR(128)	work_action_set_name - 作業アクション・セット名モニター・エレメント
WORK_CLASS_NAME	VARCHAR(128)	work_class_name - 作業クラス名モニター・エレメント
WORKLOAD_NAME	VARCHAR(128)	workload_name - ワークロード名 : モニター・エレメント
LAST_RESET	TIMESTAMP	last_reset - 最後のリセット・タイム・スタンプ・モニター・エレメント
QUEUE_SIZE_TOP	INTEGER	queue_size_top - キュー・サイズの最上位モニター・エレメント
QUEUE_TIME_TOTAL	BIGINT	queue_time_total - キュー時間の合計モニター・エレメント
QUEUE_ASSIGNMENTS_TOTAL	BIGINT	queue_assignments_total - キュー割り当ての合計モニター・エレメント
QUEUE_SIZE_CURRENT	INTEGER	キュー内の接続またはアクティビティーの数。
QUEUE_TIME_LATEST	BIGINT	最後の接続またはアクティビティーがキューをそのままにしておくためにキューで費やした時間。単位はミリ秒です。
QUEUE_EXIT_TIME_LATEST	TIMESTAMP	最後の接続またはアクティビティーがキューをそのままにしておいた時間。
THRESHOLD_CURRENT_CONCURRENCY	INTEGER	しきい値に従って現在実行中の接続またはアクティビティーの数。
THRESHOLD_MAX_CONCURRENCY	INTEGER	しきい値によって現在実行中の接続またはアクティビティーの最大数。
MEMBER	SMALLINT	member - データベース・メンバー・モニター・エレメント

WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト

WLM_GET_SERVICE_CLASS_AGENTS 関数は、指定されたサービス・クラスで (または指定されたアプリケーションの代わりに) 稼働している指定されたメンバー上のエージェント、fenced モード・プロセス (db2fmp プロセス)、およびシステム・エンティティーのリストを戻します。システム・エンティティーは、非エージェント・スレッドおよびプロセス (ページ・クリーナーおよびプリフェッチャーなど) です。

戻される可能性のある情報の完全なリストは、 428 ページの表 71を参照してください。

構文



スキーマは SYSPROC です。

表関数パラメーター

service_superclass_name
現在接続されているデータベースのサービス・スーパークラスの名前を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、データベース内のすべてのスーパークラスについてデータが取得されます。

service_subclass_name
スーパークラス内の特定のサブクラスを参照する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、データベース内のすべてのサブクラスについてデータが取得されます。

application_handle
エージェント情報が戻されるアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL である場合、データベース内のすべてのアプリケーションについてデータが取得されます。アプリケーション・ハンドルが 0 の場合、システム・エンティティーのみ戻されます。

member
現在接続されているデータベースと同じインスタンス内のメンバー番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・メンバーには -1、すべてのデータベース・メンバーには -2 を指定します。NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限

- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例 1

以下の照会は、すべてのデータベース・メンバーについてアプリケーション・ハンドル 1 に関連付けられたエージェントのリストを戻します。 **LIST APPLICATIONS** コマンドまたは WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を使用して、アプリケーション・ハンドルを判別することができます。

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
  SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
  SUBSTR(CHAR(AGENT_TID),1,9) AS AGENT_TID,
  SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
  SUBSTR(AGENT_STATE,1,10) AS AGENTSTATE,
  SUBSTR(REQUEST_TYPE,1,12) AS REQTYPE,
  SUBSTR(CHAR(UOW_ID),1,6) AS UOW_ID,
  SUBSTR(CHAR(ACTIVITY_ID),1,6) AS ACT_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS(CAST(NULL AS VARCHAR(128)),
  CAST(NULL AS VARCHAR(128)), 1, -2)) AS SCDETAILS
ORDER BY APPHANDLE, MEMB, AGENT_TID
```

出力例を以下に示します。

```
APPHANDLE MEMB AGENT_TID AGENTTYPE   AGENTSTATE REQTYPE        UOW_ID ACT_ID
--------- ---- --------- ----------- ---------- -------------- ------ ------
1         0    3         COORDINATOR ACTIVE     FETCH          1      5
1         0    4         SUBAGENT    ACTIVE     SUBSECTION:1   1      5
1         1    2         SUBAGENT    ACTIVE     SUBSECTION:2   1      5
```

この出力は、UOW ID 1 およびアクティビティー ID 5 のアクティビティーの代わりに作動している、メンバー 0 上のコーディネーター・エージェントとサブエージェント、およびメンバー 1 上のサブエージェントを示しています。AGENTTYPE 列の値 COORDINATOR に対しては、REQTYPE 列に FETCH の値があります (これは、メインまたは初期要求タイプを示しています)。これは、要求のタイプがコーディネーター・エージェントに対するフェッチ要求であることを意味しています。

例 2

以下の照会は、エージェントがどのロックを待機しているかを判別します。

```
select event_object, event_type, event_state, varchar(event_object_name, 30)
  as event_object_name
  from table(WLM_GET_SERVICE_CLASS_AGENTS('','',cast(NULL as bigint), -1)) as t
```

出力例を以下に示します。

```
EVENT_OBJECT    EVENT_TYPE        EVENT_STATE         EVENT_OBJECT_NAME
--------------- ----------------- ------------------- --------------------------
REQUEST         PROCESS           EXECUTING           -
REQUEST         PROCESS           EXECUTING           -
REQUEST         PROCESS           EXECUTING           -
REQUEST         PROCESS           EXECUTING           -
REQUEST         PROCESS           EXECUTING           -
REQUEST         PROCESS           EXECUTING           -
REQUEST         PROCESS           EXECUTING           -
REQUEST         PROCESS           EXECUTING           -
```

```
REQUEST         WAIT            IDLE            -
LOCK            ACQUIRE         IDLE            02000500000000000000000054
ROUTINE         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -

  21 record(s) selected.
```

後で同じ照会を使用すると、WLM しきい値によってエージェントがキューに入れられたことが示されます。

```
EVENT_OBJECT    EVENT_TYPE      EVENT_STATE     EVENT_OBJECT_NAME
--------------- --------------- --------------- -------------------------
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
WLM_QUEUE       WAIT            IDLE            MYCONCDBCOORDTH
ROUTINE         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -
REQUEST         PROCESS         EXECUTING       -

  21 record(s) selected.
```

使用上の注意

これらのパラメーターの作用については、論理積 (AND) が取られます。つまり、矛盾する入力パラメーターを指定する (例えば、サービス・スーパークラス SUP_A とサブクラス SUB_B を、SUB_B が SUP_A のサブクラスにならないように指定する) 場合、行は戻されません。

戻される情報

表 71. WLM_GET_SERVICE_CLASS_AGENTS によって戻される情報

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR (128)	service_superclass_name - サービス・スーパークラス名モニター・エレメント
SERVICE_SUBCLASS_NAME	VARCHAR (128)	service_subclass_name - サービス・サブクラス名モニター・エレメント

表 71. WLM_GET_SERVICE_CLASS_AGENTS によって戻される情報 (続き)

列名	データ・タイプ	説明
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル・モニター・エレメント
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
ENTITY	VARCHAR (32)	以下の値のいずれか。 • エンティティーのタイプがエージェントである場合、値は db2agent です。 • エンティティーのタイプが fenced モード・プロセスである場合、値は db2fmp (*pid*) です。ここで、*pid* は fenced モード・プロセスのプロセス ID です。 • それ以外の場合、値はシステム・エンティティーの名前です。
WORKLOAD_NAME	VARCHAR (128)	workload_name - ワークロード名 : モニター・エレメント
WORKLOAD_OCCURRENCE_ID	INTEGER	workload_occurrence_id - ワークロード・オカレンス ID モニター・エレメント
UOW_ID	INTEGER	uow_id - 作業単位 ID モニター・エレメント
ACTIVITY_ID	INTEGER	activity_id - アクティビティー ID モニター・エレメント
PARENT_UOW_ID	INTEGER	parent_uow_id - 親作業単位 ID モニター・エレメント
PARENT_ACTIVITY_ID	INTEGER	parent_activity_id - 親アクティビティー ID モニター・エレメント
AGENT_TID	BIGINT	agent_tid - エージェント・スレッド ID モニター・エレメント
AGENT_TYPE	VARCHAR (32)	エージェント・タイプ。エージェント・タイプは以下のとおりです。 • COORDINATOR • OTHER • PDBSUBAGENT • SMPSUBAGENT 値が COORDINATOR である場合、エージェント ID はコンセントレーター環境で変わることがあります。
SMP_COORDINATOR	INTEGER	エージェントが SMP コーディネーターかどうかを示します。「はい」の場合は 1、「いいえ」の場合は 0。
AGENT_SUBTYPE	VARCHAR (32)	エージェント・サブタイプ。可能なサブタイプは以下のとおりです。 • DSS • OTHER • RPC • SMP

表 71. WLM_GET_SERVICE_CLASS_AGENTS によって戻される情報 (続き)

列名	データ・タイプ	説明
AGENT_STATE	VARCHAR (32)	エージェントが関連付けられているか、アクティブであるかを示します。可能な値は次のとおりです。 • ASSOCIATED • ACTIVE
EVENT_TYPE	VARCHAR (32)	このエージェントによって最後に処理されたイベントのタイプ。可能な値は、以下のとおりです。 • ACQUIRE • PROCESS • WAIT この列に使用できる値についての詳細は、 433 ページの表 72 を参照してください。
EVENT_OBJECT	VARCHAR (32)	このエージェントによって最後に処理されたイベントのオブジェクト。可能な値は、以下のとおりです。 • COMPRESSION_DICTIONARY_BUILD • IMPLICIT_REBIND • INDEX_RECREATE • LOCK • LOCK_ESCALATION • QP_QUEUE • REMOTE_REQUEST • REQUEST • ROUTINE • WLM_QUEUE この列に使用できる値についての詳細は、 433 ページの表 72 を参照してください。
EVENT_STATE	VARCHAR (32)	このエージェントによって最後に処理されたイベントの状態。可能な値は、以下のとおりです。 • EXECUTING • IDLE この列に使用できる値についての詳細は、 433 ページの表 72 を参照してください。
REQUEST_ID	VARCHAR (64)	要求 ID。この値は、*application_handle* の値と組み合わせて指定される場合のみ固有です。この組み合わせを使用して、長い時間を要する 1 つの要求と複数の要求とを区別することができます。例えば、1 つの長いフェッチと複数のフェッチを区別するなどです。

表 71. *WLM_GET_SERVICE_CLASS_AGENTS* によって戻される情報 (続き)

列名	データ・タイプ	説明
REQUEST_TYPE	VARCHAR (32)	要求のタイプ。可能な値は、以下のとおりです。 • コーディネーター・エージェントの場合: – CLOSE – COMMIT – COMPILE – DESCRIBE – EXCSQLSET – EXECIMMD – EXECUTE – FETCH – INTERNAL *number* (*number* は内部定数の値) – OPEN – PREPARE – REBIND – REDISTRIBUTE – REORG – ROLLBACK – RUNSTATS • AGENT_SUBTYPE が DSS または SMP であるサブエージェントの場合: – サブセクション番号がゼロ以外の場合は、SUBSECTION:*subsection number* の形式のサブセクション番号。そうでない場合は NULL を戻します。

表 71. *WLM_GET_SERVICE_CLASS_AGENTS* によって戻される情報 (続き)

列名	データ・タイプ	説明
REQUEST_TYPE (続く)	VARCHAR (32)	• AGENT_SUBTYPE が RPC であるサブエージェントの場合: – ABP – CATALOG – INTERNAL – REORG – RUNSTATS – WLM • SUBTYPE が OTHER であるサブエージェントの場合: – ABP – APP_RBSVPT – APP_RELSVPT – BACKUP – CLOSE – EXTERNAL_RBSVPT – EVMON – FORCE – FORCE_ALL – INTERNAL *number* (*number* は内部定数の値) – INTERRUPT – NOOP (要求がない場合) – QP – REDISTRIBUTE – STMT_RBSVPT – STOP_USING – UPDATE_DBM_CFG – WLM
NESTING_LEVEL	INTEGER	nesting_level - ネスティング・レベル・モニター・エレメント
INVOCATION_ID	INTEGER	invocation_id - 呼び出し ID モニター・エレメント
ROUTINE_ID	INTEGER	routine_id - ルーチン ID モニター・エレメント
EVENT_OBJECT_NAME	VARCHAR (1024)	イベント・オブジェクト名。 EVENT_OBJECT の値が LOCK である場合、この列の値は、エージェントが待機するロックの名前です。 EVENT_OBJECT の値が WLM_QUEUE である場合、この列の値は、エージェントがキューに入れられている WLM しきい値の名前です。それ以外の場合、値は NULL です。
APPLICATION_NAME	VARCHAR (128)	appl_name - アプリケーション名
APPLICATION_ID	VARCHAR (128)	appl_id - アプリケーション ID

表 71. WLM_GET_SERVICE_CLASS_AGENTS によって戻される情報 (続き)

列名	データ・タイプ	説明
CLIENT_PID	BIGINT	client_pid - クライアント・プロセス ID
SESSION_AUTH_ID	VARCHAR (128)	session_auth_id - セッション許可 ID
REQUEST_START_TIME	TIMESTAMP	エージェントか現在処理中の要求の処理を開始した時刻
AGENT_STATE_LAST _UPDATE_TIME	TIMESTAMP	エージェントによって処理されているイベントが最後に変更された時刻。エージェントによって現在処理されているイベントは、EVENT_TYPE、EVENT_OBJECT、および EVENT_STATE 列で示されます。
EXECUTABLE_ID	VARCHAR (32) FOR BIT DATA	executable_id - 実行可能 ID モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー・モニター・エレメント

注: EVENT_STATE、EVENT_TYPE、EVENT_OBJECT および EVENT_OBJECT_NAME 列値の可能な組み合わせを、以下の表にリストします。

表 72. EVENT_STATE、EVENT_TYPE、EVENT_OBJECT および EVENT_OBJECT_NAME 列値の可能な組み合わせ

イベント記述	EVENT_STATE 値	EVENT_TYPE 値	EVENT_OBJECT 値	EVENT_OBJECT_ NAME 値
ロックの獲得	IDLE	ACQUIRE	LOCK	ロック名
ロックのエスカレート	EXECUTING	PROCESS	LOCK_ESCALATION	NULL
要求の処理	EXECUTING	PROCESS	REQUEST	NULL
新規要求の待機	IDLE	WAIT	REQUEST	NULL
リモート・メンバーで処理される要求の待機	IDLE	WAIT	REMOTE_REQUEST	NULL
WLM threshold キューの待機	IDLE	WAIT	WLM_QUEUE	しきい値名
ルーチンの処理	EXECUTING	PROCESS	ROUTINE	NULL
索引の再作成	EXECUTING	PROCESS	INDEX_RECREATE	NULL
コンプレッション・ディクショナリーの作成	EXECUTING	PROCESS	COMP_DICT_BLD	NULL
暗黙的な再バインド	EXECUTING	PROCESS	IMPLICIT_REBIND	NULL

WLM_GET_SERVICE_CLASS _WORKLOAD_OCCURRENCES - ワークロード・オカレンスのリスト

WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 関数は、特定のメンバー上の指定されたサービス・クラスで実行されているすべてのワークロード・オカレンスのリストを戻します。ワークロード・オカレンスとは、属性がワークロードの定義と一致しており、そのためにワークロードに関連付けられた、またはワークロードに割り当てられた特定のデータベース接続です。

戻される可能性のある情報の完全なリストは、 435 ページの表 73を参照してください。

構文

スキーマは SYSPROC です。

表関数パラメーター

service_superclass_name
現在接続されているデータベースのサービス・スーパークラスの名前を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、他のパラメーターの値と一致する、データベース内のすべてのスーパークラスについてデータが取得されます。

service_subclass_name
ワークロード・オカレンスのターゲット・サービス・サブクラス。このワークロード・オカレンスによってサブミットされる作業は、別のサブクラスにマップまたは再マップされるアクティビティーを除いて、すべてターゲット・サービス・スーパークラスの下のこのサービス・サブクラスで実行されます。

member
現在接続されているデータベースと同じインスタンス内のメンバーの番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・メンバーには -1、すべてのデータベース・メンバーには -2 を指定します。NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

システム全体で実行されているワークロード・オカレンスを管理者が調べるには、*service_superclass_name* および *service_subclass_name* に NULL 値または空ストリングを、*member* に -2 をそれぞれ指定して、WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 関数を呼び出すことができます。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(CHAR(COORD_MEMBER),1,4) AS COORDMEMB,
```

```
       SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL,
       SUBSTR(WORKLOAD_NAME,1,22) AS WORKLOAD_NAME,
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID
  FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
       (CAST(NULL AS VARCHAR(128)), CAST(NULL AS VARCHAR(128)), -2))
       AS SCINFO
  ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB, APPHNDL,
       WORKLOAD_NAME, WLO_ID
```

システムに 4 つのデータベース・メンバーがあり、現時点で 2 つのワークロードを実行していると想定すると、上記の照会は以下のような結果を生成します。

```
SUPERCLASS_NAME     SUBCLASS_NAME      MEMB COORDMEMB ...
------------------- ------------------ ---- --------- ...
SYSDEFAULTMAINTENAN SYSDEFAULTSUBCLASS 0    0         ...
SYSDEFAULTSYSTEMCLA SYSDEFAULTSUBCLASS 0    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3    0         ...

... APPHNDL WORKLOAD_NAME          WLO_ID
... ------- ---------------------- ------
... -       -                      -
... -       -                      -
... 1       SYSDEFAULTUSERWORKLOAD 1
... 2       SYSDEFAULTUSERWORKLOAD 2
... 1       SYSDEFAULTUSERWORKLOAD 1
... 2       SYSDEFAULTUSERWORKLOAD 2
... 1       SYSDEFAULTUSERWORKLOAD 1
... 2       SYSDEFAULTUSERWORKLOAD 2
... 1       SYSDEFAULTUSERWORKLOAD 1
... 2       SYSDEFAULTUSERWORKLOAD 2
```

使用上の注意

これらのパラメーターの作用については、論理積 (AND) が取られます。つまり、矛盾する入力パラメーターを指定する (例えば、サービス・スーパークラス SUP_A とサブクラス SUB_B を、SUB_B が SUP_A のサブクラスにならないように指定する) 場合、行は戻されません。

注: ワークロード・オカレンスについて報告される統計 (例えば、coord_act_completed_total) が、対応するワークロード統計と結合されると、ワークロード・オカレンスについて報告される統計が各作業単位の初めにリセットされます。

戻される情報

表 73. WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES について戻される情報

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name - サービス・スーパークラス名モニター・エレメント
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name - サービス・サブクラス名モニター・エレメント

表 73. WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES について戻される情報 (続き)

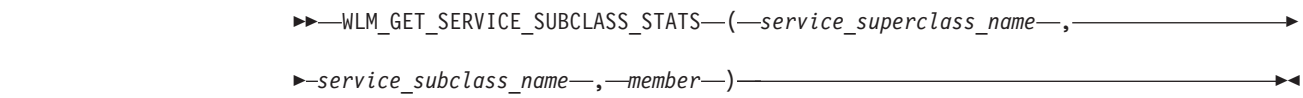
列名	データ・タイプ	説明
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
COORD_PARTITION_NUM	SMALLINT	coord_partition_num - コーディネーター・パーティション番号モニター・エレメント
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル・モニター・エレメント
WORKLOAD_NAME	VARCHAR(128)	workload_name - ワークロード名 : モニター・エレメント
WORKLOAD_OCCURRENCE_ID	INTEGER	workload_occurrence_id - ワークロード・オカレンス ID モニター・エレメント
UOW_ID	INTEGER	uow_id - 作業単位 ID モニター・エレメント
WORKLOAD_OCCURRENCE_STATE	VARCHAR(32)	workload_occurrence_state - ワークロード・オカレンスの状態モニター・エレメント
SYSTEM_AUTH_ID	VARCHAR(128)	system_auth_id - システム許可 ID モニター・エレメント
SESSION_AUTH_ID	VARCHAR(128)	session_auth_id - セッション許可 ID モニター・エレメント
APPLICATION_NAME	VARCHAR(128)	appl_name - アプリケーション名モニター・エレメント
CLIENT_WRKSTNNAME	VARCHAR(255)	client_wrkstnname - クライアント・ワークステーション名モニター・エレメント
CLIENT_ACCTNG	VARCHAR(255)	client_acctng - クライアント・アカウンティング・ストリング・モニター・エレメント
CLIENT_USER	VARCHAR(255)	このワークロード・オカレンスの CLIENT_USERID 特殊レジスターの現行値。
CLIENT_APPLNAME	VARCHAR(255)	client_applname - クライアント・アプリケーション名モニター・エレメント
COORD_ACT_COMPLETED_TOTAL	INTEGER	coord_act_completed_total - コーディネーター・アクティビティー完了総数モニター・エレメント
COORD_ACT_ABORTED_TOTAL	INTEGER	coord_act_aborted_total - コーディネーター・アクティビティー打ち切り総数モニター・エレメント
COORD_ACT_REJECTED_TOTAL	INTEGER	coord_act_rejected_total - コーディネーター・アクティビティー・リジェクト総数モニター・エレメント
CONCURRENT_ACT_TOP	INTEGER	concurrent_act_top - 並行アクティビティーの最上位モニター・エレメント
ADDRESS	VARCHAR(255)	address - 接続の開始元となった IP アドレス
APPL_ID	VARCHAR(128)	appl_id - アプリケーション ID
MEMBER	SMALLINT	member - データベース・メンバー・モニター・エレメント
COORD_MEMBER	SMALLINT	coord_member - コーディネーター・メンバー・モニター・エレメント

WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す

WLM_GET_SERVICE_SUBCLASS_STATS 関数は、1 つ以上のサービス・サブクラスの基本統計を戻します。

戻される可能性のある情報の完全なリストは、 439 ページの表 74を参照してください。

構文



スキーマは SYSPROC です。

表関数パラメーター

service_superclass_name
現在接続されているデータベースのサービス・スーパークラスの名前を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、データベース内のすべてのスーパークラスについてデータが取得されます。

service_subclass_name
現在接続されているデータベースのサービス・サブクラスの名前を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、データベース内のすべてのサブクラスについてデータが取得されます。

member
現在接続されているデータベースと同じインスタンス内の有効なメンバー番号を指定する、タイプ INTEGER の入力引数。現行のメンバーには -1、すべてのデータベース・メンバーには -2 を指定します。NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

例 1: すべてのアクティビティーは実行前に DB2 サービス・クラスに対してマップされる必要があるため、サービス・クラス統計表関数を使用し、すべてのメンバー上のすべてのサービス・クラスを照会して、システムの全体的な状態をモニターできます。以下の例では、NULL 値が *service_superclass_name* と *service_subclass_name* に渡されてすべてのサービス・クラスの統計を戻し、*member* には値 -2 が指定されてすべてのメンバーの統計を戻します。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3))
         AS AVGLIFETIME,
       CAST(COORD_ACT_LIFETIME_STDDEV / 1000 AS DECIMAL(9,3))
         AS STDDEVLIFETIME,
       SUBSTR(CAST(LAST_RESET AS VARCHAR(30)),1,16) AS LAST_RESET
  FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),
       CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
  ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB
```

ステートメントは、以下の出力例で示されているように、アクティビティー存続期間の平均および標準偏差などのサービス・クラス統計を秒単位で戻します。

```
SUPERCLASS_NAME     SUBCLASS_NAME      MEMB ...
------------------- ------------------ ---- ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3    ...

... AVGLIFETIME STDDEVLIFETIME LAST_RESET
... ----------- -------------- ----------------
...     691.242         34.322 2006-07-24-11.44
...     644.740         22.124 2006-07-24-11.44
...     612.431         43.347 2006-07-24-11.44
...     593.451         28.329 2006-07-24-11.44
```

例 2: また、同じ表関数が、各メンバー上のサービス・クラスで実行しているコーディネーター・アクティビティーの平均並行性の最高値を示すこともできます。

```
  SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
         SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
         SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
         CONCURRENT_ACT_TOP AS ACTTOP,
         CONCURRENT_WLO_TOP AS CONNTOP
  FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),
         CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
  ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB
```

以下はその出力例です。

```
SUPERCLASS_NAME     SUBCLASS_NAME      MEMB ACTTOP    CONNTOP
------------------- ------------------ ---- --------- ---------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0           10         7
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1            0         0
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2            0         0
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3            0         0
```

この表関数の出力内のアクティビティーの平均実行時間および回数を調べることで、特定のデータベースの各メンバーの負荷の概要を、的確に知ることができます。この表関数によって戻される高水準ゲージが大きく変わった場合は、システムの負荷の変化を示していることがあります。

例 3: アクティビティーが REMAP ACTIVITY TO アクションでしきい値を使用する場合、そのアクティビティーは、その存続時間中に複数のサービス・クラスで時間を費やすことがあります。以下の例に示すように、ACT_REMAPPED_IN 列と ACT_REMAPPED_OUT 列を調べると、一連のサービス・クラスを通過したアクティビティーの数を判別できます。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       ACT_REMAPPED_IN AS MAPPED_IN,
       ACT_REMAPPED_OUT AS MAPPED_OUT
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),
       CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME
```

以下はその出力例です。

```
SUPERCLASS_NAME     SUBCLASS_NAME      MAPPED_IN MAPPED_OUT
------------------- ------------------ --------- ----------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS         0          0
SUPERCLASS1         SYSDEFAULTSUBCLASS         0          0
SUPERCLASS1         SUBCLASS1                  0          7
SUPERCLASS1         SUBCLASS2                  7          0
```

使用上の注意

一部の統計は、対応するサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA および COLLECT AGGREGATE REQUEST DATA パラメーターを NONE 以外の値に設定した場合のみ、戻されます。

WLM_GET_SERVICE_SUBCLASS_STATS 表関数は、サービス・サブクラスごとおよびメンバーごとに 1 行のデータを戻します。この関数は、(パーティション上の) サービス・クラス全体または (1 つ以上のサービス・クラスの) パーティション全体のデータ集約は行いません。ただし、SQL 照会を使用すると、データを集約することができます。

これらのパラメーターの作用については、論理積 (AND) が取られます。つまり、例えばスーパークラス SUPA と、SUPA のサブクラスでないサブクラス SUBB などの競合する入力パラメーターを指定した場合、行は戻されません。

戻される情報

表 74. WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name - サービス・スーパークラス名モニター・エレメント
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name - サービス・サブクラス名モニター・エレメント
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
LAST_RESET	TIMESTAMP	last_reset - 最後のリセット・タイム・スタンプ・モニター・エレメント
COORD_ACT_COMPLETED_TOTAL	BIGINT	coord_act_completed_total - コーディネーター・アクティビティー完了総数モニター・エレメント

表 74. *WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報 (続き)*

列名	データ・タイプ	説明
COORD_ACT_ABORTED_TOTAL	BIGINT	coord_act_aborted_total - コーディネーター・アクティビティー打ち切り総数モニター・エレメント
COORD_ACT_REJECTED_TOTAL	BIGINT	coord_act_rejected_total - コーディネーター・アクティビティー・リジェクト総数モニター・エレメント
CONCURRENT_ACT_TOP	INTEGER	concurrent_act_top - 並行アクティビティーの最上位モニター・エレメント
COORD_ACT_LIFETIME_TOP	BIGINT	coord_act_lifetime_top - コーディネーター・アクティビティー存続時間の最上位モニター・エレメント
COORD_ACT_LIFETIME_AVG	DOUBLE	coord_act_lifetime_avg - コーディネーター・アクティビティー平均存続期間モニター・エレメント
COORD_ACT_LIFETIME_STDDEV	DOUBLE	最後のリセット以降、このサービス・サブクラスに関連付けられたネスト・レベル 0 のコーディネーター・アクティビティーの存続期間の標準偏差。サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA パラメーターが NONE に設定される場合、列の値は NULL です。単位はミリ秒です。 この標準偏差はコーディネーター・アクティビティーの存続期間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。 サービス・サブクラスの COORD_ACT_LIFETIME_STDDEV 値は、完了前にサービス・サブクラスを通過したものの別のサブクラスに再マップされるアクティビティーの影響は受けません。
COORD_ACT_EXEC_TIME_AVG	DOUBLE	coord_act_exec_time_avg - コーディネーター・アクティビティー平均実行時間モニター・エレメント

表 74. WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
COORD_ACT_EXEC_TIME_STDDEV	DOUBLE	最後のリセット以降、このサービス・サブクラスに関連付けられたネスト・レベル 0 のコーディネーター・アクティビティーの実行時間の標準偏差。単位はミリ秒です。 この標準偏差はコーディネーター・アクティビティーの実行時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。 サービス・サブクラスの実行時間標準偏差は、完了前にサブクラスを通過したものの別のサブクラスに再マップされるアクティビティーの影響は受けません。
COORD_ACT_QUEUE_TIME_AVG	DOUBLE	coord_act_queue_time_avg - コーディネーター・アクティビティー平均キュー時間モニター・エレメント
COORD_ACT_QUEUE_TIME_STDDEV	DOUBLE	最後のリセット以降、このサービス・サブクラスに関連付けられたネスト・レベル 0 のコーディネーター・アクティビティーのキュー時間の標準偏差。サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA パラメーターが NONE に設定される場合、列の値は NULL です。単位はミリ秒です。 この標準偏差はコーディネーター・アクティビティーのキュー時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。 キュー時間標準偏差のカウント対象は、アクティビティーがキューに入れられたサービス・サブクラスだけです。
NUM_REQUESTS_ACTIVE	BIGINT	この表関数の実行時にサービス・サブクラスで実行している要求の数。

表 74. *WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報 (続き)*

列名	データ・タイプ	説明
NUM_REQUESTS_TOTAL	BIGINT	最後のリセット以降、このサービス・サブクラスで実行を終了する要求の数。この終了状態は、アクティビティー内の要求のメンバーシップに関係なく、任意の要求に適用されます。サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA パラメーターが NONE に設定される場合、列の値は NULL です。 サービス・サブクラスの NUM_REQUESTS_TOTAL 値は、サービス・サブクラスを通過したものの、その中で完了しない要求の影響は受けません。
REQUEST_EXEC_TIME_AVG	DOUBLE	request_exec_time_avg - 要求の平均実行時間モニター・エレメント
REQUEST_EXEC_TIME_STDDEV	DOUBLE	最後のリセット以降、このサービス・サブクラスに関連付けられた要求の実行時間の標準偏差。単位はミリ秒です。サービス・クラスの COLLECT AGGREGATE REQUEST DATA パラメーターが NONE に設定される場合、この列の値は NULL です。 この標準偏差は要求実行時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。 サービス・サブクラスの実行時間標準偏差は、サブクラスを通過したものの、その中で完了しなかった要求の影響は受けません。

表 74. WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
REQUEST_EXEC_TIME_TOTAL	BIGINT	最後のリセット以降、このサービス・サブクラスに関連付けられた要求の実行時間の合計。単位はミリ秒です。サービス・クラスの COLLECT AGGREGATE REQUEST DATA パラメーターが NONE に設定される場合、この列の値は NULL です。 この合計は要求実行時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。 サービス・サブクラスの実行時間合計は、サブクラスを通過したものの、その中で完了しない要求の影響は受けません。
ACT_REMAPPED_IN	BIGINT	最後のリセット以降にしきい値 REMAP ACTIVITY アクションによってこのサービス・サブクラスに再マップされたアクティビティー数。
ACT_REMAPPED_OUT	BIGINT	最後のリセット以降にしきい値 REMAP ACTIVITY アクションによってこのサービス・サブクラスから出て再マップされたアクティビティー数。
CONCURRENT_WLO_TOP	INTEGER	concurrent_wlo_top - 並行ワークロード・オカレンスの最上位モニター・エレメント
UOW_TOTAL_TIME_TOP	BIGINT	uow_total_time_top - 作業単位合計時間トップ・モニター・エレメント
UOW_THROUGHPUT	DOUBLE	uow_throughput - 作業単位スループット・モニター・エレメント 統計を最後にリセットして以降の作業単位のスループット。
UOW_LIFETIME_AVG	DOUBLE	uow_lifetime_avg - 作業単位存続期間平均モニター・エレメント
UOW_COMPLETED_TOTAL	BIGINT	uow_completed_total - 完了した作業単位合計数モニター・エレメント
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間モニター・エレメント
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャー実行キュー時間の合計モニター・エレメント
ACT_THROUGHPUT	DOUBLE	act_throughput - アクティビティー・スループット・モニター・エレメント
CPU_UTILIZATION	DOUBLE	cpu_utilization - CPU 使用率モニター・エレメント

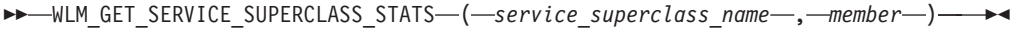
表 74. *WLM_GET_SERVICE_SUBCLASS_STATS* について戻される情報 (続き)

列名	データ・タイプ	説明
APP_ACT_COMPLETED_TOTAL	BIGINT	app_act_completed_total - 成功した外部コーディネーター・アクティビティーの総数モニター・エレメント
APP_ACT_ABORTED_TOTAL	BIGINT	app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの総数モニター・エレメント
APP_ACT_REJECTED_TOTAL	BIGINT	app_act_rejected_total - リジェクトされた外部コーディネーター・アクティビティーの総数モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー・モニター・エレメント

WLM_GET_SERVICE_SUPERCLASS_STATS - サービス・スーパークラスの統計を戻す

WLM_GET_SERVICE_SUPERCLASS_STATS 関数は、1 つ以上のサービス・スーパークラスの基本統計を戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

service_superclass_name
現在接続されているデータベースのサービス・スーパークラスの名前を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、データベース内のすべてのスーパークラスについてデータが取得されます。

member
現在接続されているデータベースと同じインスタンス内の有効なメンバー番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・メンバーには -1、すべてのデータベース・メンバーには -2 を指定します。NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

以下の照会は、システム上のすべてのサービス・スーパークラスの基本統計を、すべてのデータベース・メンバーにわたって表示します。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME, 1, 26) SERVICE_SUPERCLASS_NAME,
       MEMBER,
       LAST_RESET,
       CONCURRENT_CONNECTION_TOP CONCURRENT_CONN_TOP
FROM TABLE(WLM_GET_SERVICE_SUPERCLASS_STATS('', -2)) as SCSTATS
```

出力例を以下に示します。

```
SERVICE_SUPERCLASS_NAME    MEMBER ...
-------------------------- ------------- ...
SYSDEFAULTSYSTEMCLASS                  0 ...
SYSDEFAULTMAINTENANCECLASS             0 ...
SYSDEFAULTUSERCLASS                    0 ...

... LAST_RESET                 CONCURRENT_CONN_TOP
... -------------------------- -------------------
... 2006-09-05-09.38.44.396788                   0
... 2006-09-05-09.38.44.396795                   0
... 2006-09-05-09.38.44.396796                   1
```

使用上の注意

WLM_GET_SERVICE_SUPERCLASS_STATS 表関数は、サービス・スーパークラスおよびメンバーごとに 1 行のデータを戻します。この関数は、(メンバー上の) サービス・スーパークラス全体または (1 つ以上のサービス・スーパークラスの) メンバー全体のデータ集約は行いません。ただし、上の例で示された SQL 照会を使用すると、データを集約することができます。

戻される情報

表 75. WLM_GET_SERVICE_SUPERCLASS_STATS について戻される情報

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name - サービス・スーパークラス名モニター・エレメント
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
LAST_RESET	TIMESTAMP	last_reset - 最後のリセット・タイム・スタンプ・モニター・エレメント
CONCURRENT_CONNECTION_TOP	INTEGER	concurrent_connection_top - 並行接続の最上位モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー・モニター・エレメント

WLM_GET_WORK_ACTION_SET_STATS - 作業アクション・セット統計を戻す

WLM_GET_WORK_ACTION_SET_STATS 関数は、作業アクション・セットの統計を戻します。

構文

スキーマは SYSPROC です。

表関数パラメーター

work_action_set_name
統計を戻すワーク・アクション・セットを指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、すべてのワーク・アクション・セットについて統計が戻されます。

member
現在接続されているデータベースと同じインスタンス内の有効なメンバー番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・メンバーには -1、すべてのデータベース・メンバーには -2 を指定します。NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

3 つの作業クラス、ReadClass、WriteClass、および LoadClass があると想定します。 ReadClass に関連した作業アクションと LoadClass に関連した作業アクションはありますが、WriteClass に関連した作業アクションはありません。メンバー 0 上で現在実行されている、またはキューに入れられているアクティビティーの数は次のとおりです。

- ReadClass クラス: 8
- WriteClass クラス: 4
- LoadClass クラス: 2
- 未割り当て: 3

```
SELECT SUBSTR(WORK_ACTION_SET_NAME,1,18) AS WORK_ACTION_SET_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(WORK_CLASS_NAME,1,15) AS WORK_CLASS_NAME,
       LAST_RESET,
       SUBSTR(CHAR(ACT_TOTAL),1,14) AS ACT_TOTAL
  FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS
       (CAST(NULL AS VARCHAR(128)), -2)) AS WASSTATS
  ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME, MEMB
```

出力例を以下に示します。 WriteClass 作業クラスに関連した作業アクションはないため、その作業クラスが該当する 4 つのアクティビティーは、出力でアスタリスク (*) によって示される人工的なクラスでカウントされます。どの作業クラスにも割り当てられなかった 3 つのアクティビティーも、人工的なクラスに組み込まれています。

```
WORK_ACTION_SET_NAME MEMB WORK_CLASS_NAME LAST_RESET                ACT_TOTAL
-------------------- ---- --------------- ------------------------- --------------
AdminActionSet       0    ReadClass       2005-11-25-18.52.49.343000 8
AdminActionSet       1    ReadClass       2005-11-25-18.52.50.478000 0
AdminActionSet       0    LoadClass       2005-11-25-18.52.49.343000 2
AdminActionSet       1    LoadClass       2005-11-25-18.52.50.478000 0
AdminActionSet       0    *               2005-11-25-18.52.49.343000 7
AdminActionSet       1    *               2005-11-25-18.52.50.478000 0
```

戻される情報

表 76. WLM_GET_WORK_ACTION_SET_STATS について戻される情報

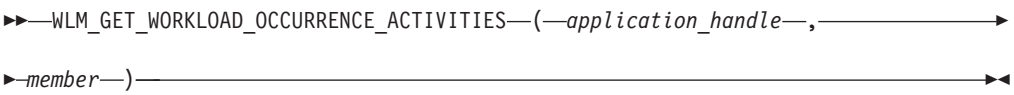
列名	データ・タイプ	説明
WORK_ACTION_SET_NAME	VARCHAR(128)	work_action_set_name - 作業アクション・セット名モニター・エレメント
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
LAST_RESET	TIMESTAMP	last_reset - 最後のリセット・タイム・スタンプ・モニター・エレメント
WORK_CLASS_NAME	VARCHAR(128)	work_class_name - 作業クラス名モニター・エレメント
ACT_TOTAL	BIGINT	act_total - アクティビティーの合計モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー・モニター・エレメント

WLM_GET_WORKLOAD _OCCURRENCE _ACTIVITIES - アクティビティーのリストを戻す

WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 関数は、指定されたメンバー上の指定されたアプリケーションによってサブミットされ、まだ完了していないすべてのアクティビティーのリストを戻します。

戻される可能性のある情報の完全なリストは、 449 ページの表 77を参照してください。

構文



スキーマは SYSPROC です。

表関数パラメーター

application_handle
アクティビティーのリストが戻されるアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL である場合、データベース内のすべてのアプリケーションについてデータが取得されます。

member
現在接続されているデータベースと同じインスタンス内の有効なメンバー番号を指定する、タイプ INTEGER の入力引数。現行メンバーであれば -1、すべてのメンバーであれば -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

認識済みのアプリケーション・ハンドルで現在実行中のアクティビティー

アプリケーション・ハンドルを識別した後、このアプリケーションで現在実行中のすべてのアクティビティーを検索できます。 **LIST APPLICATIONS** コマンドを使用して判別されるアプリケーション・ハンドルが 1 であるアプリケーションのアクティビティーをリストすることを管理者が望んでいるという場面を、例として考えてみましょう。管理者は以下の照会を実行します。

```
SELECT SUBSTR(CHAR(COORD_MEMBER),1,5) AS COORD,
     SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
     SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
     SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
     SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
     SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PARACTID,
            ACTIVITY_TYPE AS ACTTYPE,
     SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(1, -2)) AS WLOACTS
ORDER BY MEMB, UOWID, ACTID
```

照会からの出力例は、次のようになります。

```
COORD MEMB UOWID ACTID PARUOWID PARACTID ACTTYPE  NESTING
----- ---- ----- ----- -------- -------- -------- -------
0     0    2     3     -        -        CALL     0
0     0    2     5     2        3        READ_DML 1
```

システムで現在実行中のアクティビティー

以下の照会では、EXECUTABLE_ID で WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 出力と MON_GET_PKG_CACHE_STMT 出力を結合し、システムで現在実行中のすべての SQL アクティビティーに関するステートメント・テキストを提供します。

```
SELECT t.application_handle,
       t.uow_id,
       t.activity_id,
       varchar(p.stmt_text, 256) as stmt_text
FROM table(wlm_get_workload_occurrence_ACTIVITIES(NULL, -1)) as t,
     table(mon_get_pkg_cache_stmt(NULL, NULL, NULL, -1)) as p
WHERE t.executable_id = p.executable_id
```

出力例を以下に示します。

```
APPLICATION_HANDLE   UOW_ID      ACTIVITY_ID    STMT_TEXT
-------------------- ----------- -------------- ------------------------------
1                    1           1              SELECT * FROM SYSCAT.TABLES
47                   1           36             INSERT INTO T1 VALUES(123)
```

戻される情報

表 77. WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES によって戻される情報

列名	データ・タイプ	説明
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル・モニター・エレメント
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
COORD_PARTITION_NUM	SMALLINT	coord_partition_num - コーディネーター・パーティション番号モニター・エレメント
LOCAL_START_TIME	TIMESTAMP	local_start_time - ローカル開始時刻モニター・エレメント
UOW_ID	INTEGER	uow_id - 作業単位 ID モニター・エレメント
ACTIVITY_ID	INTEGER	activity_id - アクティビティー ID モニター・エレメント
PARENT_UOW_ID	INTEGER	parent_uow_id - 親作業単位 ID モニター・エレメント
PARENT_ACTIVITY_ID	INTEGER	parent_activity_id - 親アクティビティー ID モニター・エレメント
ACTIVITY_STATE	VARCHAR(32)	activity_state - アクティビティーの状態モニター・エレメント

表 77. *WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES* によって戻される情報 (続き)

列名	データ・タイプ	説明
ACTIVITY_STATE (続き)	VARCHAR(32)	アクティビティーの状態。可能な値は、以下のとおりです。 *QUEUED* アクティビティーが、ワークロード管理キューイングしきい値によってキューに入れられています。パーティション・データベース環境では、この状態は、コーディネーター・エージェントがカタログ・メンバーに対する RPC を行ってしきい値チケットを取得したものの、まだ応答を受け取っていないことを示す場合があります。この状態は、アクティビティーがワークロード管理キューイングしきい値によってキューに入れられているか、それほど時間が経過していない場合はアクティビティーがそのチケットを取得する処理中であることを示すことがあります。アクティビティーがキューに入れられているかどうかについての詳細を知るために、どのエージェントがアクティビティーで作業しているかを判別し、カタログ・メンバーにあるオブジェクトの EVENT_OBJECT の値が WLM_QUEUE の値であるかどうかを調べてください。 *TERMINATING* アクティビティーは実行を完了し、システムから除去されています。
ACTIVITY_TYPE	VARCHAR(32)	アクティビティー・タイプ。可能な値は、以下のとおりです。 • CALL • DDL • LOAD • OTHER • READ_DML • WRITE_DML
NESTING_LEVEL	INTEGER	nesting_level - ネスティング・レベル・モニター・エレメント
INVOCATION_ID	INTEGER	invocation_id - 呼び出し ID モニター・エレメント
ROUTINE_ID	INTEGER	routine_id - ルーチン ID モニター・エレメント

表 77. *WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES* によって戻される情報 (続き)

列名	データ・タイプ	説明
UTILITY_ID	INTEGER	utility_id - ユーティリティー ID モニター・エレメント
SERVICE_CLASS_ID	INTEGER	service_class_id - サービス・クラス ID モニター・エレメント
DATABASE_WORK_ACTION_SET_ID	INTEGER	以下の値のいずれか。 • このアクティビティーがデータベース有効範囲の作業クラスに分類されている場合、値はこの作業クラスがメンバーとなっている作業クラス・セットの ID です。 • このアクティビティーがデータベース有効範囲の作業クラスに分類されていない場合、値は NULL です。
DATABASE_WORK_CLASS_ID	INTEGER	以下の値のいずれか。 • このアクティビティーがデータベース有効範囲の作業クラスに分類されている場合、値は作業クラスの ID です。 • このアクティビティーがデータベース有効範囲の作業クラスに分類されていない場合、値は NULL です。
SERVICE_CLASS_WORK_ACTION_SET_ID	INTEGER	以下の値のいずれか。 • このアクティビティーがサービス・クラス有効範囲の作業クラスに分類されている場合、値は作業クラスが属する作業クラス・セットに関連付けられた作業アクション・セットの ID です。 • このアクティビティーがサービス・クラス有効範囲の作業クラスに分類されていない場合、値は NULL です。
SERVICE_CLASS_WORK_CLASS_ID	INTEGER	以下の値のいずれか。 • このアクティビティーがサービス・クラス有効範囲の作業クラスに分類されている場合、この値はこのアクティビティーに割り当てられた作業クラスの ID です。 • このアクティビティーがサービス・クラス有効範囲の作業クラスに分類されていない場合、値は NULL です。
EXECUTABLE_ID	VARCHAR(32) FOR BIT DATA	executable_id - 実行可能 ID モニター・エレメント
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間
ROWS_READ	BIGINT	rows_read - 読み取り行数
ROWS_RETURNED	BIGINT	rows_returned - 戻り行数
QUERY_COST_ESTIMATE	BIGINT	query_cost_estimate - 照会コストの見積もり
DIRECT_READS	BIGINT	direct_reads - データベースからの直接読み取り

表 77. WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES によって戻される情報 (続き)

列名	データ・タイプ	説明
DIRECT_WRITES	BIGINT	direct_writes - データベースへの直接書き込み
ENTRY_TIME	TIMESTAMP	entry_time - エントリー時間
MEMBER	SMALLINT	member - データベース・メンバー・モニター・エレメント
COORD_MEMBER	SMALLINT	coord_member - コーディネーター・メンバー・モニター・エレメント
PACKAGE_SCHEMA	VARCHAR(128)	package_schema - パッケージ・スキーマ。
PACKAGE_NAME	VARCHAR(128)	package_name - パッケージ名。
PACKAGE_VERSION_ID	VARCHAR (64)	package_version_id - パッケージ・バージョン。
SECTION_NUMBER	BIGINT	section_number - セクション番号。
STMTNO	INTEGER	stmtno - ステートメント番号モニター・エレメント

WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す

WLM_GET_WORKLOAD_STATS 関数は、ワークロード名とデータベース・メンバー番号のすべての組み合わせについてワークロード統計の 1 行を返します。

戻される可能性のある情報の完全なリストは、 453 ページの表 78を参照してください。

構文

スキーマは SYSPROC です。

表関数パラメーター

workload_name
統計が戻されるワークロードを指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、すべてのワークロードについて統計が戻されます。

member
現在接続されているデータベースと同じインスタンス内のメンバーの番号を指定する、タイプ INTEGER の入力引数。現行のメンバーには -1、すべてのメンバーには -2 を指定します。NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権

- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

以下の照会は、ワークロードの統計を表示します。

```
SELECT SUBSTR(WORKLOAD_NAME,1,18) AS WL_DEF_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       COORD_ACT_LIFETIME_TOP,
       COORD_ACT_LIFETIME_AVG,
       COORD_ACT_LIFETIME_STDDEV
FROM TABLE(WLM_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2)) AS WLSTATS
ORDER BY WL_DEF_NAME, MEMB
```

照会からの出力例は、次のようになります。

```
WL_DEF_NAME        MEMB COORD_ACT_LIFETIME_TOP ...
------------------ ---- ---------------------- ...
SYSDEFAULTADMWORKL 0                        -1 ...
SYSDEFAULTUSERWORK 0                        -1 ...
WL1                0                         2 ...

... COORD_ACT_LIFETIME_AVG   COORD_ACT_LIFETIME_STDDEV
... ------------------------ ------------------------
... -1.00000000000000E+000    -1.00000000000000E+000
... -1.00000000000000E+000    -1.00000000000000E+000
... +2.56000000000000E+000    +6.00000000000001E-002
```

使用上の注意

この関数は、ワークロード、メンバー、またはサービス・クラスを越えたデータ集約は行いません。ただし、SQL 照会を使用すると、データを集約することができます。

戻される情報

表 78. WLM_GET_WORKLOAD_STATS によって戻される情報

列名	データ・タイプ	説明
WORKLOAD_NAME	VARCHAR(128)	workload_name - ワークロード名 : モニター・エレメント
DBPARTITIONNUM	SMALLINT	dbpartitionnum - データベース・パーティション番号モニター・エレメント
LAST_RESET	TIMESTAMP	last_reset - 最後のリセット・タイム・スタンプ・モニター・エレメント
CONCURRENT_WLO_TOP	INTEGER	concurrent_wlo_top - 並行ワークロード・オカレンスの最上位モニター・エレメント
CONCURRENT_WLO_ACT_TOP	INTEGER	concurrent_wlo_act_top - 並行 WLO アクティビティーの最上位モニター・エレメント

表 78. *WLM_GET_WORKLOAD_STATS* によって戻される情報 (続き)

列名	データ・タイプ	説明
COORD_ACT_COMPLETED_TOTAL	BIGINT	coord_act_completed_total - コーディネーター・アクティビティー完了総数モニター・エレメント
COORD_ACT_ABORTED_TOTAL	BIGINT	coord_act_aborted_total - コーディネーター・アクティビティー打ち切り総数モニター・エレメント
COORD_ACT_REJECTED_TOTAL	BIGINT	coord_act_rejected_total - コーディネーター・アクティビティー・リジェクト総数モニター・エレメント
WLO_COMPLETED_TOTAL	BIGINT	wlo_completed_total - 完了したワークロード・オカレンスの合計モニター・エレメント
COORD_ACT_LIFETIME_TOP	BIGINT	coord_act_lifetime_top - コーディネーター・アクティビティー存続時間の最上位モニター・エレメント
COORD_ACT_LIFETIME_AVG	DOUBLE	coord_act_lifetime_avg - コーディネーター・アクティビティー平均存続期間モニター・エレメント
COORD_ACT_LIFETIME_STDDEV	DOUBLE	このワークロードに関連付けられたネスト・レベル 0 の完了または打ち切られたコーディネーター・アクティビティーの存続期間の標準偏差。単位はミリ秒です。ワークロードの COLLECT AGGREGATE ACTIVITY DATA パラメーターが NONE に設定される場合、列の値は NULL です。この標準偏差はコーディネーター・アクティビティーの存続期間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。
COORD_ACT_EXEC_TIME_AVG	DOUBLE	coord_act_exec_time_avg - コーディネーター・アクティビティー平均実行時間モニター・エレメント
COORD_ACT_EXEC_TIME_STDDEV	DOUBLE	このワークロードに関連付けられたネスト・レベル 0 の完了または打ち切られたコーディネーター・アクティビティーの実行時間の標準偏差。単位はミリ秒です。この標準偏差はコーディネーター・アクティビティーの実行時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。 値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。 ワークロードの COLLECT AGGREGATE ACTIVITY DATA パラメーターが NONE に設定される場合、列の値は NULL です。
COORD_ACT_QUEUE_TIME_AVG	DOUBLE	coord_act_queue_time_avg - コーディネーター・アクティビティー平均キュー時間モニター・エレメント

表 78. WLM_GET_WORKLOAD_STATS によって戻される情報 (続き)

列名	データ・タイプ	説明
COORD_ACT_QUEUE_TIME_STDDEV	DOUBLE	このワークロードに関連付けられたネスト・レベル 0 の完了または打ち切られたコーディネーター・アクティビティーのキュー時間の標準偏差。単位はミリ秒です。ワークロードの COLLECT AGGREGATE ACTIVITY DATA パラメーターが NONE に設定される場合、列の値は NULL です。この標準偏差はコーディネーター・アクティビティーのキュー時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。
UOW_TOTAL_TIME_TOP	BIGINT	uow_total_time_top - 作業単位合計時間トップ・モニター・エレメント
UOW_THROUGHPUT	DOUBLE	uow_throughput - 作業単位スループット・モニター・エレメント
UOW_LIFETIME_AVG	DOUBLE	uow_lifetime_avg - 作業単位存続期間平均モニター・エレメント
UOW_COMPLETED_TOTAL	BIGINT	uow_completed_total - 完了した作業単位合計数モニター・エレメント
TOTAL_CPU_TIME	BIGINT	total_cpu_time - 合計 CPU 時間モニター・エレメント
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - ディスパッチャー実行キュー時間の合計モニター・エレメント
ACT_THROUGHPUT	DOUBLE	act_throughput - アクティビティー・スループット・モニター・エレメント
CPU_UTILIZATION	DOUBLE	cpu_utilization - CPU 使用率モニター・エレメント
APP_ACT_COMPLETED_TOTAL	BIGINT	app_act_completed_total - 成功した外部コーディネーター・アクティビティーの総数モニター・エレメント
APP_ACT_ABORTED_TOTAL	BIGINT	app_act_aborted_total - 失敗した外部コーディネーター・アクティビティーの総数モニター・エレメント
APP_ACT_REJECTED_TOTAL	BIGINT	app_act_rejected_total - リジェクトされた外部コーディネーター・アクティビティーの総数モニター・エレメント
MEMBER	SMALLINT	member - データベース・メンバー・モニター・エレメント

WLM_SET_CLIENT_INFO プロシージャー - クライアント情報設定

WLM_SET_CLIENT_INFO プロシージャーは、DB2 サーバーでの現行接続に関連付けられたクライアント情報を設定します。

このプロシージャーを使用することにより、クライアントのユーザー ID、アプリケーション名、ワークステーション名、アカウンティング情報、またはワークロード

情報を DB2 サーバーで設定できます。このプロシージャーを呼び出すと、この接続に関する、関連するトランザクション・プロセッサー (TP) モニターのクライアント情報フィールドおよび特殊レジスター設定の保管された値が変更されます。

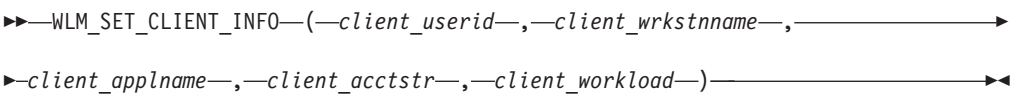
クライアント情報フィールドは、現在接続を使用しているアプリケーションまたはユーザーの ID を判別するために、DB2 サーバーで使用されます。接続用のクライアント情報フィールドは、DB2 のワークロード評価中に考慮され、この接続用に生成される DB2 監査レコードまたはアプリケーション・スナップショットにも表示されます。

sqleseti API とは異なり、このプロシージャーは、クライアントでクライアント情報を設定するのではなく、対応するクライアント属性を DB2 サーバーで設定します。そのため、このプロシージャーを使用して DB2 サーバーに設定されたクライアント情報を照会するために sqleqry API を使用することはできません。アプリケーションが sqleseti API を使用してクライアント情報を変更する場合、新しい値によって DB2 サーバーでの設定値が変更されます。 sqleseti API を使用してアカウンティング情報を変更することなくユーザー ID またはアプリケーション名のいずれかを変更すると、DB2 サーバーのアカウンティング情報もクライアントのアカウンティング情報の値にリセットされます。

このプロシージャーで提供されたデータ値は、関連した TP モニターのフィールドまたは特殊レジスターに保管される前に、適切なデータベースのコード・ページに変換されます。データベースのコード・ページへの変換後に最大サポート・サイズを超えるデータ値は、サーバーに保管される前に切り捨てられます。切り捨てられた値は、それら保管された値が照会されるときに TP モニターのフィールドおよび特殊レジスターの両方によって返されます。

WLM_SET_CLIENT_INFO プロシージャーはトランザクションの制御下にはなく、このプロシージャーにより加えられたクライアント情報変更は、作業単位のコミットまたはロールバックとは無関係です。ただし、各アプリケーションの次の作業単位の開始時にワークロードの再評価が行われるので、クライアント情報変更を有効にするには、COMMIT または ROLLBACK のいずれかのステートメントを発行する必要があります。

構文



スキーマは SYSPROC です。

プロシージャー・パラメーター

client_userid
クライアントのユーザー ID を指定する、タイプ VARCHAR(255) の入力引数。 NULL を指定する場合、値は変わりません。空ストリング (デフォルト値) を指定する場合、クライアントのユーザー ID はデフォルト値であるブランクにリセットされます。

client_wrkstnname
クライアントのワークステーション名を指定する、タイプ VARCHAR(255) の入力引数。 NULL を指定する場合、値は変わりません。空ストリング (デフォルト値) を指定する場合、クライアントのワークステーション名はデフォルト値であるブランクにリセットされます。

client_applname
クライアントのアプリケーション名を指定する、タイプ VARCHAR(255) の入力引数。 NULL を指定する場合、値は変わりません。空ストリング (デフォルト値) を指定する場合、クライアントのアプリケーション名はデフォルト値であるブランクにリセットされます。

client_acctstr
クライアント・アカウンティング・ストリングを指定する、タイプ VARCHAR(200) の入力引数。 NULL を指定する場合、値は変わりません。空ストリング (デフォルト値) を指定する場合、クライアント・アカウンティング・ストリングはデフォルト値であるブランクにリセットされます。

client_workload
クライアントのワークロードの割り当てを指定する、タイプ VARCHAR(255) の入力引数。 NULL を指定する場合、値は変わりません。値は、以下のとおりです。

SYSDEFAULTADMWORKLOAD
データベース接続が SYSDEFAULTADMWORKLOAD に割り当てられることを指定します。これにより、ACCESSCTRL、DATAACCESS、DBADM、SECADM、または WLMADM 権限を持つユーザーは通常のワークロード評価を迂回することができます。

AUTOMATIC
サーバーが自動的に実行するワークロード計算に選ばれているワークロードに、データベース接続を割り当てるよう指定します。

注: *client_workload* 引数は大文字小文字を区別します。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限
- WLMADM 権限

デフォルトの PUBLIC 特権

なし

例

以下のプロシージャー呼び出しは、クライアントのユーザー ID、ワークステーション名、アプリケーション名、アカウンティング・ストリング、およびワークロード割り当てモードを設定します。

```
CALL SYSPROC.WLM_SET_CLIENT_INFO('db2user', 'machine.torolab.ibm.com',
   'auditor', 'Accounting department', 'AUTOMATIC')
```

以下のプロシージャー呼び出しは、クライアントのユーザー ID を db2user2 に設定し、他のクライアント属性は設定しません。

```
CALL SYSPROC.WLM_SET_CLIENT_INFO('db2user2', NULL, NULL, NULL, NULL)
```

以下のプロシージャー呼び出しは、クライアントのユーザー ID をブランクにリセットし、他のクライアント属性の値を変更しません。

```
CALL SYSPROC.WLM_SET_CLIENT_INFO('', NULL, NULL, NULL, NULL)
```

使用上の注意

プロシージャー・パラメーターに指定した入力がその指定フィールド長を超える場合、入力フィールドは切り捨てられ、プロシージャーは切り捨てられた入力を使用して実行されます。

単一引用符を含む入力フィールドはサポートされておらず、エラーになります。

ワークロード管理に関するモニター・エレメント

次のモニター・エレメントにより、アクティビティー、しきい値違反、およびワークロード管理の統計に関する情報が提供されます。

act_cpu_time_top - アクティビティーの CPU 時間の最上位 : モニター・エレメント

サービス・クラス、ワークロード、または作業クラスでの、すべてのネスト・レベルにおけるアクティビティーで使用されるプロセッサー時間の最高水準点。この値はマイクロ秒単位で報告されます。

アクティビティーが実行されるサービス・クラスまたはワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。要求メトリックが使用可能になっている場合にのみ、アクティビティーはこの最高水準点に寄与します。アクティビティー・メトリックの収集が使用可能になっていない場合には、値 0 が返されます。

サービス・クラスでは、REMAP ACTIVITY アクションを使用してサービス・サブクラス間でアクティビティーを再マップすると、新規の最高水準点に達した場合、アクティビティーが完了するサービス・サブクラスの act_cpu_time_top 最高水準点のみが更新されます。アクティビティーがマップされるものの完了していない他のサービス・サブクラスの act_cpu_time_top 最高水準点は、影響を受けません。

表 79. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーでアクティビティーによって使用された最大プロセッサー時間を判別することができます。

act_exec_time アクティビティー実行時間 : モニター・エレメント

act_exec_time エレメントは、このメンバーで実行するために費やされた時間 (マイクロ秒単位) を格納します。

カーソルの場合、実行時間はオープン、フェッチ、およびクローズの時間を組み合わせたものです。カーソルのアイドル時間は実行時間にカウントされません。ルーチンの場合、実行時間はルーチン呼び出しの開始から終了までです。ルーチンの完了後にそのルーチンによって (結果セットを戻すために) オープンされたままになっているカーソルの存続期間は、ルーチンの実行時間にカウントされません。他のすべてのアクティビティーの場合、実行時間は開始時刻から停止時刻までの時間です。どの場合でも、実行時間には、初期化されている時間またはキューに入れられている時間は含まれません。

表 80. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを単独で使用すると、メンバーごとに DB2 によるアクティビティーの実行に費やされた経過時間を知ることができます。このエレメントは、**time_started** および **time_completed** モニター・エレメントと一緒にコーディネーター・メンバーで使用して、カーソル・アクティビティーにおけるアイドル時間を計算することもできます。以下の公式を使用できます。

```
カーソルのアイドル時間 = (time_completed - time_started) - act_exec_time
```

act_remapped_in - 再マッピングするアクティビティー：モニター・エレメント

最後のリセット以降に、このサービス・サブクラスに再マッピングするアクティビティーの数。

表 81. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このカウントを使用して、サービス・サブクラスへのアクティビティーの再マップが期待どおりに行われているかを判別します。

act_remapped_out - 再マッピングの際に除外されるアクティビティー：モニター・エレメント

最後のリセット以降、再マッピングの際にこのサービス・サブクラスから除外されるアクティビティーの数。

表 82. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このカウントを使用して、サービス・サブクラスからのアクティビティーの再マップが期待どおりに行われているかを判別します。

act_rows_read_top - アクティビティーの読み取り行数の最上位：モニター・エレメント

サービス・クラス、ワークロード、または作業クラスでの、すべてのネスト・レベルにおけるアクティビティーによって読み取られる行数の最高水準点。

アクティビティーが実行されるサービス・クラスまたはワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。要求メトリックが使用可能になっている場合にのみ、アクティビティーはこの最高水準点に寄与します。アクティビティー・メトリックの収集が使用可能になっていない場合には、値 0 が返されます。

サービス・クラスでは、REMAP ACTIVITY アクションを使用してサービス・サブクラス間でアクティビティーを再マップすると、新規の最高水準点に達した場合、アクティビティーが完了するサービス・サブクラスの act_rows_read_top 最高水準点のみが更新されます。アクティビティーがマップされるものの完了していないサービス・サブクラスの act_rows_read_top 最高水準点は、影響を受けません。

表 83. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーでアクティビティーによって読み取られる最大行数を判別することができます。

act_throughput - アクティビティー・スループット : モニター・エレメント

任意のネスト・レベルでコーディネーター・アクティビティーが完了するレート。1秒単位のコーディネーター・アクティビティー数で測定されます。

表 84. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 85. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (metrics 文書に報告されます)	常に収集される
統計	event_wlstats (metrics 文書に報告されます)	常に収集される

使用法

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降のアクティビティー・スループットを表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降のアクティビティー・スループットを表します。

act_total アクティビティーの合計 : モニター・エレメント

最後にリセットしてから指定した作業クラスに対応する作業アクションが適用された、任意のネスト・レベルのアクティビティーの合計数。

表 86. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE

表 87. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	-

使用法

作業クラスに関連付けられた 1 つ以上の作業アクションがアクティビティーに適用されるたびに、この作業クラスのカウンターが更新されます。**act_total** モニター・エレメントを使用すると、このカウンターが公開されます。このカウンターを使用して、作業アクション・セットの有効性 (例えば、アクションが適用されているアクティビティーの数) を判断できます。また、システム上のアクティビティーのさまざまなタイプを理解するためにも使用できます。

activate_timestamp タイム・スタンプの活動化 : モニター・エレメント

イベント・モニターがアクティブにされた時刻。

表 88. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
アクティビティー	event_activitystmt	-
アクティビティー	event_activityvals	-
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、上記のイベント・タイプで戻された情報を関連付けることができます。

activity_collected 収集されたアクティビティー : モニター・エレメント

このエレメントは、しきい値の違反が発生した場合にアクティビティー・イベント・モニター・レコードが収集されるかどうかを示します。

表 89. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値を違反したアクティビティーのアクティビティー・イベントがアクティビティー・イベント・モニターに書き込まれるかどうかを判別できます。

アクティビティーが完了またはアボートし、その時点でアクティビティー・イベント・モニターがアクティブである場合、このモニター・エレメントの値が「Y」である場合には、このしきい値に違反したアクティビティーは収集されます。このモニター・エレメントの値が「N」である場合、それは収集されません。

activity_id アクティビティー ID : モニター・エレメント

特定の作業単位内のアプリケーションのアクティビティーを一意的に識別するカウンター。

表 90. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 91. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
アクティビティー	event_activity	常に収集される
アクティビティー	event_activitystmt	常に収集される
アクティビティー	event_activityvals	常に収集される
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
しきい値違反	event_thresholdviolations	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

アクティビティーをその作業単位外から一意的に識別するには、**activity_id** および **uow_id** と、**appl_id** または **agent_id** のいずれかのモニター・エレメントを組み合わせて使用してください。

activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント

このエレメントの値は、同じアクティビティーに関してアクティビティー・レコードが書き込まれるたびに増分されます。

例えば、アクティビティー・レコードが、WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを呼び出した結果として 1 回目に書き込まれ、アクティビティーが終了した時に 2 回目に書き込まれた場合、エレメントの値は、最初のレコードについては 0、2 番目のレコードについては 1 となります。

表 92. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
アクティビティー	event_activitystmt	-
アクティビティー	event_activityvals	-
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE

使用法

このエレメントを **activity_id**、**uow_id**、および **appl_id** モニター・エレメントと一緒に使用すると、同一のアクティビティーに関する情報がアクティビティー・イベント・モニターに複数回書き込まれた場合にアクティビティー・レコードを一意的に識別できます。

例えば、以下の場合には、アクティビティーに関する情報がアクティビティー・イベント・モニターに 2 回送信されます。

- WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを使用して、実行中のアクティビティーに関する情報をキャプチャーした場合
- アクティビティーが関連付けられているサービス・クラス上で COLLECT ACTIVITY DATA 文節を指定したために、そのアクティビティーの完了時にそのアクティビティーに関する情報を収集した場合。

activity_type アクティビティー・タイプ : モニター・エレメント

アクティビティーのタイプ。

表 93. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 94. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

可能な値は以下のとおりです。

- LOAD
- READ_DML
- WRITE_DML
- DDL
- CALL
- OTHER

SQL を実行しない SET ステートメント (SET 特殊レジスターや SET EVENT MONITOR STATE など) および LOCK TABLE ステートメント場合、値 OTHER が戻ります。

agg_temp_tablespace_top - TEMPORARY 表スペースの集約最上位 : モニター・エレメント

agg_temp_tablespace_top モニター・エレメントは、サービス・クラスでの、すべてのネスト・レベルにおける DML アクティビティーの TEMPORARY 表スペースの集約された使用量の最高水準点 (KB 単位) を格納します。

集約は、サービス・サブクラスのすべてのアクティビティーに渡る TEMPORARY 表スペースの使用量を合計して計算されます。この最高水準点は、最後のリセット以降のこの集約の最大値を表しています。 サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合に、このモニター・エレメントは -1 を返します。このレコードのあるサブクラスと同じスーパークラスに、AGGSQLTEMPSPACE しきい値が定義され使用可能になっているサービス・サブクラスが少なくとも 1 つ必要です。そうでない場合、0 の値が返されます。

表 95. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される

使用法

このエレメントを使用して、収集された時間間隔にサービス・サブクラスのメンバーで到達した DML アクティビティーの SYSTEM TEMPORARY 表スペース使用量の最大集約値を判別することができます。

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

アプリケーション応答測定 (ARM) 標準のトランザクションの ID。

表 96. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを使用すると、アクティビティー・イベント・モニターによって収集されるアクティビティーに関連付けられたアプリケーションもアプリケーション応答測定 (ARM) 標準をサポートする場合には、このアクティビティーをそのアプリケーションにリンクさせることができます。

bin_id ヒストグラム・ビン ID : モニター・エレメント

ヒストグラム・ビンの ID。**bin_id** はヒストグラム内で固有です。

表 97. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントを使用すると、同じヒストグラム内でビンを区別できます。

bottom ヒストグラム・ビンの最下位 : モニター・エレメント

ヒストグラム・ビンの範囲外の最終点。このモニター・エレメントの値は、前のヒストグラム・ビンがある場合には、その最終範囲に含まれる最初の値になります。

表 98. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントと対応する **top** エレメントを一緒に使用して、ヒストグラム中のビンの範囲を判別します。

concurrent_act_top 並行アクティビティーの最上位 : モニター・エレメント

最後にリセットされてからのサービス・サブクラスにおける並行アクティビティー (すべてのネスト・レベル) の最高水準点。

注: このエレメントは、すべてのアクティビティーの最大同時実行数をモニターします。これには、CONCURRENTDBCOORDACTIVITIES しきい値とは関係がないア

クティビティーも含まれます。例えば、CALL ステートメントは、CONCURRENTDBCOORDACTIVITIES しきい値で制御される同時実行数には数えられませんが、現行アクティビティーの最高水準点の計測値には含まれます。

表 99. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE

表 100. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・サブクラス用のメンバーで到達したアクティビティー (ネストされたアクティビティーを含む) の並行性の最大数を調べることができます。

concurrent_connection_top 並行接続の最上位 : モニター・エレメント

最後にリセットされてからのこのサービス・クラスにおける並行コーディネーター接続の最高水準点。同じスーパークラスを持つすべてのサブクラスにおいて、このフィールドの値は同じです。

表 101. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE

表 102. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このエレメントは、現在の最高水準点がある場所を示すことにより、接続並行性のどの位置にしきい値を設定するかを判別する上で役立つ場合があります。さらに、そのようなしきい値が正しく構成され、作動しているかを検証する上でも役立ちます。

concurrent_wlo_act_top 並行 WLO アクティビティーの最上位 : モニター・エレメント

最後にリセットされてからの、このワークロードの任意のオカレンスにおける並行アクティビティー (すべてのネスト・レベル) の最高水準点。

表 103. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 104. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にこのワークロードの任意のオカレンス用のメンバーで到達した並行アクティビティーの最大数を調べることができます。

concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント

最後にリセットされてからのワークロードの並行オカレンスの最高水準点。

表 105. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 106. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-
統計	event_scstats	-

使用法

このエレメントを使用して、収集された時間間隔にワークロード用のメンバーで到達したワークロード・オカレンスの並行性の最大数を調べることができます。

concurrentdbcoordactivities_db_ threshold_id - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値 ID モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値の ID。

表 107. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES データベースしきい値がアクティビティーに適用されていた場合、どのデータベースしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_db_threshold _queued 並行データベース・コーディネーター・アクティビティーのデータベースしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES データベースしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 108. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES データベースしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_db_ threshold_value - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値の上限を戻します。

表 109. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES データベースしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_db_ threshold_violated - 並行データベース・コーディネーター・アクティビティーのデータベースしきい値の違反モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES データベースしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 110. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_subclass_ threshold_id - 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 ID モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値の ID を戻します。

表 111. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値がアクティビティーに適用されていた場合、どのサービス・サブクラスしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_subclass_ threshold_queued 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 112. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_subclass_ threshold_value 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 : モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値の上限を戻します。

表 113. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_ACTIVITY_DETAILS 表関数 - 特定のアクティビティーに関する詳細情報を戻す	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_subclass_ threshold_violated 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 114. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_superclass_ threshold_id 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値 ID : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES_SUPERCLASS しきい値の ID。

表 115. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_superclass_ threshold_queued 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値によりキ

ューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 116. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_superclass_ threshold_value 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値 : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値の上限。

表 117. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_superclass_ threshold_violated 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 118. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_wl_was_threshold _id - 並行データベース・コーディネーター・アクティビティーのワークロード作業アクション・セットしきい値 ID : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値の ID。

表 119. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_wl_was_threshold _queued - 並行データベース・コーディネーター・アクティビティーのワークロード作業アクション・セットしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 120. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_wl_was_threshold _value - 並行データベース・コーディネーター・アクティビティーのワークロード作業アクション・セットしきい値 : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値の上限。

表 121. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用された CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値の値を判別します。

concurrentdbcoordactivities_wl_was_threshold _violated - 並行データベース・コーディネーター・アクティビティーのワークロード作業アクション・セットしきい値違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 122. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES ワークロード作業アクション・セットしきい値にアクティビティーが違反したかどうかを判別します。

coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント

最後にリセットしてからの、エラーで完了した任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

サービス・クラスでは、アクティビティーが異常終了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、このアクティビティーのカウントは、異常終了時のサブクラスでの合計にのみ含められます。

表 123. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 124. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、システム上のアクティビティーが正常に完了しているかを知ることができます。アクティビティーは、取り消し、エラー、または反作用しきい値のために打ち切られる場合があります。

coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント

最後にリセットしてからの、正常に完了した任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

サービス・クラスでは、アクティビティーが完了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、このアクティビティーのカウントは、完了時のサブクラスでの合計にのみ含められます。

表 125. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 126. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-
統計	event_scstats	-

使用法

このエレメントを使用すると、システムのアクティビティーのスループットを判別したり、複数のメンバー間の平均アクティビティー存続時間の計算を補助したりすることができます。

coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター DML アクティビティーの見積コストの算術平均。

内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA EXTENDED 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。timeron 単位で測定されます。

サービス・クラスの場合、アクティビティーの見積コストは、アクティビティーがシステムに入るのに使用するサービス・サブクラスでしかカウントされません。REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの coord_act_est_cost_avg 平均は影響されません。

表 127. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される

使用法

この統計を使用すると、最後の統計リセット以降に完了または異常終了したこのサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター DML アクティビティーの見積コストの算術平均を判別できます。

この平均を使用して、アクティビティー見積コストのヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー見積コストのヒストグラムからアクティビティーの平均見積コストを計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー見積コストのヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの実行時間の算術平均。

内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 単位はミリ秒です。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーがマップされたが完了しなかったサービス・サブクラスの coord_act_exec_time_avg 平均は影響されません。

表 128. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	COLLECT AGGREGATE ACTIVITY DATA

表 129. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーの実行時間の算術平均を判別できます。

この平均を使用して、アクティビティー実行時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー実行時間のヒストグラムから平均アクティビティー実行時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー実行時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの到着間隔の時間の算術平均。

内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA EXTENDED 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。ミリ秒単位で測定されます。

サービス・クラスの場合、inter-arrival 時間の平均は、アクティビティーがシステムに入るのに使用するサービス・サブクラスから計算されます。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの coord_act_interarrival_time_avg は影響されません。

表 130. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、このサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの到着間隔の算術平均を判別できます。

到着間隔の時間を使用して、到着レートを判別できます。到着レートは到着間隔の時間の逆になります。この平均を使用して、アクティビティー到着間隔の時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー到着間隔の時間のヒストグラムから平均アクティビティー到着間隔の時間を計算してください。計算した平均をこのモニター・エレ

メントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー到着間隔の時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの存続時間の算術平均。

内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 ミリ秒単位で測定されます。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了した最後のサービス・クラスの coord_act_lifetime_avg 平均のみが影響されます。

表 131. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	COLLECT AGGREGATE ACTIVITY DATA

表 132. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーの存続時間の算術平均を判別できます。

この統計を使用して、アクティビティー存続時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー存続時間のヒストグラムから平均アクティビティー存続時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均

が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー存続時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント

coord_act_lifetime_top エレメントは、すべてのネスト・レベルでカウントされる、コーディネーター・アクティビティー存続期間の最高水準点です。この情報は、ミリ秒間単位の表記で格納されます。

サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップする際に、サービス・クラスでこの統計を効果的に使用するには、与えられたサービス・サブクラスの coord_act_lifetime_top 最高水準点と、同じ再マップのしきい値 (複数可) によって影響される他のサブクラスの最高水準点を集約する必要があります。 これは、アクティビティーが完了するのは、再マップしきい値によって別のサービス・サブクラスへ再マップされた後になるからであり、アクティビティーが再マップされるまで他のサービス・サブクラスにいた時間は、完了時のサブクラスでの合計にのみ含められるからです。

表 133. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	COLLECT AGGREGATE ACTIVITY DATA

表 134. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	-
統計	event_scstats	-
統計	event_wlstats	-

使用法

このエレメントを使用すると、アクティビティー存続時間のしきい値が有効であるかどうかを判別する助けになります。さらに、そのようなしきい値を構成する方法を判別する助けとすることもできます。

coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーのキュー時間の算術平均。

内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 ミリ秒単位で測定されます。

サービス・クラスの場合、キュー時間はアクティビティーが完了するもしくは異常終了したサービス・サブクラスでしかカウントされません。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーがマップされたが完了しなかったサービス・サブクラスの coord_act_queue_time_avg 平均は影響されません。

表 135. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	COLLECT AGGREGATE ACTIVITY DATA

表 136. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	常に収集される
統計	event_wcstats	常に収集される
統計	event_wlstats	常に収集される

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーのキュー時間の算術平均を判別できます。

この統計を使用して、アクティビティー・キュー時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー・キュー時間のヒストグラムから平均アクティビティー・キュー時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビテ

ィー・キュー時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント

coord_act_rejected_total は、最後にリセットしてからの、実行が許可されず、リジェクトされた任意のネスト・レベルのコーディネーター・アクティビティーの合計数を格納します。

このカウンターは、予測しきい値または実行阻止作業アクションのいずれかによりアクティビティーの実行が阻止された場合に更新されます。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

表 137. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE ACTIVITY DATA
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 138. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-

使用法

このエレメントを使用すると、予測しきい値および実行を阻止する作業アクションが有効であるかどうか、および、それらの制限が大きすぎないかどうかを判別する助けになります。

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

作業単位またはアクティビティーのコーディネーター・パーティション。複数パーティションのシステムでは、コーディネーター・パーティションは、アプリケーションがデータベースに接続したパーティションです。

表 139. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 139. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 140. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
しきい値違反	event_thresholdviolations	常に収集される

使用法

このエレメントを使用して、コーディネーター以外のパーティションにレコードがあるアクティビティーまたは作業単位の、コーディネーター・パーティションを識別できます。

cost_estimate_top コスト見積もりの最上位 : モニター・エレメント

cost_estimate_top モニター・エレメントは、サービス・サブクラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの見積コストの最高水準点です。

サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、DML アクティビティーの見積コストは、アクティビティーがシステムに入るのに使用するサービス・サブクラスでしかカウントされません。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの cost_estimate_top は影響されません。

表 141. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーで到達した DML アクティビティー見積コストの最大値を判別することができます。

cpu_limit - WLM ディスパッチャーの CPU リミット : モニター・エレメント

サービス・クラスに関して構成されている WLM ディスパッチャーの CPU リミット。

表 142. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE

cpu_share_type - WLM ディスパッチャー CPU シェア・タイプのモニター・エレメント

サービス・クラスに関して構成されている WLM ディスパッチャーの CPU シェアのタイプ。値は、soft および hard のどちらかです。

表 143. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE

cpu_shares - WLM ディスパッチャーの CPU 共有 : モニター・エレメント

サービス・クラスに関して構成されている WLM ディスパッチャーの CPU 共有数。

表 144. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE

cpu_utilization - CPU 使用率 : モニター・エレメント

特定の論理パーティション上のサービス・クラスまたはワークロードが消費した合計 CPU 時間を、特定の期間にホストまたは LPAR で使用可能だった CPU 時間で除算した数値。

表 145. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 146. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (metrics 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (metrics 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このモニター・エレメントが WLM_GET_WORKLOAD_STATS 関数または WLM_GET_SERVICE_SUBCLASS_STATS 関数によって戻される場合、最後に統計を最後にリセットして以降の CPU 使用率を表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降の CPU 使用率を表します。

cpu_velocity - CPU 速度モニター・エレメント

CPU リソースの競合の度合いの測定。 0 から 1 までのスケールで表され、数値が小さいほど、CPU リソースの競合が大きいことを意味します。

CPU 速度は、サービス・クラス内の処理が CPU にアクセスした時間を、CPU にアクセスしたり、 CPU にアクセスするために待機したりした合計時間で除算して算出します。この指標によって、作業が CPU をまったく待機しないで実行された場合の効率に比べ、どれほど効率的に実行されているかを測定することができます。数式は、次のとおりです。

```
cpu_velocity = total_cpu_time / (total_cpu_time + total_disp_run_queue_time)
```

cpu_velocity を収集するには、 **wlm_dispatcher** データベース・マネージャー構成パラメーターを ON に設定する必要があります。

表 147. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラス・メトリックのサンプリング	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロード・メトリックのサンプリング	REQUEST METRICS BASE

使用法

ディスパッチャーは、サービス・クラスまたはワークロードが、ある瞬間に付与可能量より多くの CPU リソースを要求する場合に、サービス・クラスまたはワークロードを優先順位付けする効果を持っています。このような場合、サービス・クラスまたはワークロード内で実行される処理は、 CPU リソースにアクセスするためのキューイングに時間を費やします。ディスパッチャーが、他のサービス・クラスまたはワークロードに付与する CPU リソースを減らして、そのようなサービス・クラスまたはワークロードに、より多くの CPU リソースを付与することができるのは、このようなときです。 CPU 速度が高い場合、現行のレベルの CPU 要求は既に満たされているため、ディスパッチャーは、このサービス・クラスに、応答時間またはスループットを向上させる影響をほとんど与えていないことを意味します。 CPU 速度が低い場合、ディスパッチャーは、現行レベルの CPU 要求のサービス・クラスまたはワークロードに、応答時間またはスループットを向上させる重大な影響を与えている可能性があることを意味します。

このエレメントを使用すると、サービス・クラスまたはワークロード内で実行される処理が、 CPU リソースを使用するためのキューイングに費やす時間の比率が、高いかどうかを調べることができます。サービス・クラスの CPU 速度が低い場合、高くするには、 CPU シェア数を増やしたり、低い CPU 速度を示すサービス・クラスに割り当てられている CPU リミットを大きくしたりして、CPU リソースの WLM ディスパッチャー制御を調整します。

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

このアクティビティーがデータベース有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスが所属する作業クラス・セットに関連した作業アクション・セットの ID を示します。それ以外の場合、このモニター・エレメントは 0 の値を示します。

表 148. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE

表 149. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントと **db_work_class_id** エレメントを組み合わせて使用すると、アクティビティーのデータベース作業クラスが存在する場合にはそれを一意的に識別できます。

db_work_class_id データベース作業クラス ID : モニター・エレメント

このアクティビティーがデータベース有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 150. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 151. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントと **db_work_action_set_id** エレメントを組み合わせて使用すると、アクティビティーのデータベース作業クラスが存在する場合にはそれを一意的に識別できます。

destination_service_class_id - 宛先サービス・クラス ID : モニター・エレメント

このエレメントのしきい値違反レコードが生成された時に、アクティビティーが再マップされたサービス・サブクラスの ID。 しきい値アクションが REMAP ACTIVITY 以外の場合、このエレメントの値はゼロです。

表 152. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントは、アクティビティーが再マップされたサービス・クラスをたどるのに使用できます。 このエレメントを使用して、特定のサービス・サブクラスに対

してマップされたアクティビティー数の総計を計算することもできます。

estimated_cpu_entitlement - 見積もりの CPU 割り当て率のモニター・エレメント

サービス・サブクラスが CPU 割り当て率に基づいて消費するように構成されている、ホストまたは LPAR 上の合計 CPU の比率。サービス・サブクラスは、消費するように構成された比率を上回ることも下回ることもなく消費するという想定で構成されます。

この計算にどのサービス・クラスを参加させるかは、サンプリング期間にわたって測定された実際の CPU 使用率、対 WLM_DISP_MIN_UTIL データベース・マネージャー構成設定値に基づいて決定されます。サービス・クラス自体、サービスが競合するサービス・クラス、または親のサービス・クラス (ある場合) が CPU リミットによって受ける影響は、計算には考慮されていません。

表 153. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE

histogram_type ヒストグラム・タイプ : モニター・エレメント

ヒストグラムのタイプ (ストリング形式)。

ヒストグラムには、7 つのタイプがあります。

CoordActQueueTime
ネストなしアクティビティーがキュー (しきい値キューなど) に入れられていた時間のヒストグラム (ミリ秒単位)。コーディネーター・メンバー上で測定されます。

CoordActExecTime
ネストなしアクティビティーのコーディネーター・メンバー上での実行時間のヒストグラム (ミリ秒単位)。実行時間には、初期化されている時間またはキューに入れられている時間は含まれません。カーソルの場合、実行時間にはオープン、フェッチ、およびクローズ要求に要する時間のみ含まれます。アクティビティーがサービス・サブクラス間で再マップされると、実行時間ヒストグラムは、アクティビティーが実行を完了するサービス・サブクラスについてのみ更新されます。

CoordActLifetime
ネストなしアクティビティーがデータベース・マネージャーによって識別されてから、そのアクティビティーが実行を完了するまでの経過時間のヒストグラム (ミリ秒単位)。コーディネーター・メンバー上で測定されます。 アクティビティーをサービス・サブクラス間で再マップした場合、存続時間ヒストグラムは、アクティビティーが実行を完了するサービス・サブクラスについてのみ更新されます。

CoordActInterArrivalTime
ネストなしコーディネーター・アクティビティーが到着してから次が到着す

るまでの時間間隔のヒストグラム (ミリ秒単位)。到着間隔時間の平均値は、アクティビティーがシステムに入るときに使用されるサービス・サブクラスを対象に計算されます。アクティビティーをサービス・サブクラス間で再マップした場合、アクティビティーの再マップ先のサービス・サブクラスの到着間隔時間ヒストグラムは影響を受けません。

CoordActEstCost
ネストなし DML アクティビティーの見積コストのヒストグラム (timeron 単位)。アクティビティーの見積コストは、アクティビティーがシステムに入るときのサービス・サブクラスに関してのみカウントされます。

ReqExecTime
要求の実行時間のヒストグラム (ミリ秒単位)。要求には、コーディネーター・メンバーでの要求と、コーディネーター・メンバーと非コーディネーター・メンバーの両方でのサブリクエスト (RPC 要求、SMP サブエージェント要求など) が含まれます。含まれる要求には、アクティビティーに関連付けられるものとそうでないものがあります。例えば、このヒストグラムには PREPARE 要求と OPEN 要求の両方が含まれています。しかし、OPEN 要求は常にカーソル・アクティビティーと関連付けられるのに対し、PREPARE 要求はアクティビティーの一部ではありません。再マップに関係するサービス・サブクラスの実行時間ヒストグラムでは、そのサービス・サブクラス内で部分要求が費やす実行時間部分がカウントされます。

UowLifetime
データベース・マネージャーによって作業単位が識別されてから、その作業単位の実行を完了 (コミットまたはロールバック) するまでの経過時間のヒストグラム (ミリ秒単位)。

表 154. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	常に収集される

使用法

このエレメントを使用すると、ヒストグラムのタイプを識別できます。複数のヒストグラムが同じ統計レコードに所属する場合がありますが、各タイプごとに 1 つずつしか所属しません。

last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント

このエレメントは、このタイプの統計イベント・レコードが最後に作成された時刻をローカル・タイム・スタンプの形式で示します。

表 155. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-
統計	event_wcstats	-
統計	event_qstats	-

使用法

wlm_last_reset および **statistics_timestamp** モニター・エレメントを使用すると、イベント・モニター統計レコード中の統計が収集された期間を判別できます。収集間隔の開始時刻は **wlm_last_reset** で、終了時刻は **statistics_timestamp** です。

num_remaps 再マップ数 : モニター・エレメント

このアクティビティーが再マップされた回数のカウント。 num_remaps がゼロより大きい場合、このアクティビティー・レコードの service_class_id は、アクティビティーが最後に再マップされたサービス・クラスの ID です。

表 156. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

この情報を使用して、予期された回数アクティビティーが再マップされたかどうかを検証します。

num_threshold_violations しきい値違反の回数 : モニター・エレメント

このデータベースが最後にアクティブにされてからそこで発生したしきい値違反の回数。

このモニター・エレメントは、いくつかのモニター (MON_*) 表関数によって戻される 508 ページの『thresh_violations - しきい値違反の回数 : モニター・エレメント』 モニター・エレメントの別名です。

表 157. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 158. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	常に収集される

使用法

このエレメントを使用すると、この特定のアプリケーションにおいてしきい値が有効であるかどうか、またはしきい値違反が多すぎないかを判別する助けになります。

number_in_bin ビン内の数 : モニター・エレメント

このエレメントは、ヒストグラム・ビンの中に入るアクティビティーまたは要求のカウント数を保持します。

表 159. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントを使用すると、ヒストグラムのビンの高さを示すことができます。

parent_activity_id 親アクティビティー ID : モニター・エレメント

アクティビティーの親アクティビティーの作業単位内における、その親アクティビティーのユニーク ID。親アクティビティーがない場合、このモニター・エレメントの値は 0 です。

表 160. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 161. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを **parent_uow_id** エレメントおよび **appl_id** エレメントと組み合わせて使用すると、このアクティビティー・レコードで記述されているアクティビティーの親アクティビティーを一意的に識別できます。

parent_uow_id 親作業単位 ID : モニター・エレメント

アプリケーション・ハンドル内の固有の作業単位 ID。アクティビティーの親アクティビティーが発生する作業単位の ID。親アクティビティーがない場合、値は 0 です。

表 162. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 163. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを **parent_activity_id** エレメントおよび **appl_id** エレメントと組み合わせて使用すると、このアクティビティー・レコードで記述されているアクティビティーの親アクティビティーを一意的に識別できます。

prep_time 準備時間 : モニター・エレメント

SQL ステートメントを準備するために要した時間 (ミリ秒単位) (アクティビティーが SQL ステートメントである場合。それ以外の場合の値は 0)。

表 164. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメントのアクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE

表 165. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される
パッケージ・キャッシュ	-	COLLECT BASE DATA

使用法

prep_time モニター・エレメントは、このアクティビティーが SQL ステートメントだった場合に、SQL ステートメントが DB2 パッケージ・キャッシュに最初に取り込まれたときのステートメントの準備に費やされた時間を示します。この準備時間

は、アクティビティー存続時間の一部ではありません。また、ステートメントが呼び出しの前に既にパッケージ・キャッシュに入れられていた場合のステートメントの特定の呼び出しに費やされた時間を表しているわけでもありません。

queue_assignments_total キュー割り当ての合計 : モニター・エレメント

最後にリセットされてからこのしきい値キューに接続またはアクティビティーが割り当てられた回数。

表 166. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 167. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、統計収集間隔により決定される特定の期間にアクティビティーまたは接続がこの特定のキューに入れられた回数を判別できます。 これは、キューのしきい値の効果性を判別する助けになります。

queue_size_top キュー・サイズの最上位 : モニター・エレメント

最後にリセットしてから到達したキュー・サイズの最大値。

表 168. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 169. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、キューのしきい値の効果性を測定したり、キューイングが大きすぎる時を検出したりすることができます。

queue_time_total キュー時間の合計 : モニター・エレメント

最後にリセットされてからキューに置かれたすべての接続またはアクティビティーについて、このキューで費やされた合計時間。単位はミリ秒です。

表 170. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 171. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	常に収集される

このエレメントは、キューイングのしきい値の有効性を評価する場合や、過度のキューイングを検出する場合に使用します。

使用上の注意

queue_time_total は、統計収集間隔の終わりにリセットされません。**queue_time_total** を複数の間隔で使用した場合は、**wlm_collect_int** と **queue_size_top** の積よりも大きくなることがあります。

request_exec_time_avg 要求の平均実行時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスに関連付けられた要求の実行時間の算術平均。内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスの COLLECT AGGREGATE REQUEST DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。単位はミリ秒です。

REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、request_exec_time_avg 平均は再マップに関係した部分要求をカウントします。

表 172. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	COLLECT AGGREGATE REQUEST DATA

表 173. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

この統計を使用すると、このサービス・サブクラス中のメンバー上での要求ごとの平均処理時間を即時に知ることができます。

この平均を使用して、要求の実行時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。要求の実行時間のヒストグラムから要求の平均実行時間を計算してください。計算した平均をこのモニタ

ー・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、要求の実行時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

routine_id - ルーチン ID : モニター・エレメント

固有なルーチン ID。このモニター・エレメントでは、アクティビティーがルーチンの一部ではない場合、0 が戻されます。

表 174. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_ROUTINE_EXEC_LIST 表関数 - ルーチンによって実行されるステートメントのリストの取得	常に収集される
MON_GET_SECTION_ROUTINE 表関数 - 入力セクションのルーチンのリストの取得	常に収集される
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 175. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	常に収集される
作業単位	uow_package_list	常に収集される
パッケージ・キャッシュ	pkgcache_metrics	ACTIVITY METRICS BASE

使用法

このエレメントの値は、ビュー SYSCAT.ROUTINES の列 ROUTINEID の値と一致します。アクティビティーが別の SQL PL ルーチン内で宣言された SQL PL ルーチンの一部である場合、このエレメントの値は外部ルーチンの ROUTINEID です。

rows_fetched フェッチ行数 : モニター・エレメント

表から読み取られた行の数。

このモニター・エレメントは、**rows_read** モニター・エレメントの別名です。

注: このモニター・エレメントは、この情報を記録する対象としたメンバーにおける値のみを報告します。マルチメンバー・データベース環境の場合、これらの値は、アクティビティー全体での総量を正確に示していない場合があります。

表 176. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	ステートメント

使用法

詳しくは、**rows_read** モニター・エレメントを参照してください。

rows_modified 変更行数 : モニター・エレメント

挿入、更新、または削除された行数。

このモニター・エレメントは、**rows_written** モニター・エレメントの別名です。

表 177. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される

表 177. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 178. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (metrics 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (metrics 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
アクティビティー	event_activity	ステートメント
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

詳しくは、**rows_written** モニター・エレメントを参照してください。

rows_returned 戻り行数 : モニター・エレメント

rows_returned モニター・エレメントは、選択されてアプリケーションに戻された行の数です。

このエレメントは、アクティビティー・レコードが部分的な場合 (例えば、アクティビティーがまだ実行中に収集された場合、またはメモリーの制約のために完全なアクティビティー・レコードをイベント・モニターに書き込むことができなかったとき) に、値が 0 になります。

このモニター・エレメントは、**fetch_count** モニター・エレメントの別名です。

表 179. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 179. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	REQUEST METRICS BASE

表 180. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (metrics 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (metrics 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE
アクティビティー	event_activity	常に収集される
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE

使用法

このエレメントを使用すると、アプリケーションに戻される行数のしきい値を判別する助けになります。または、そのようなしきい値が正しく構成され、作動しているかを検証するために使用できます。

rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント

rows_returned_top モニター・エレメントは、サービス・クラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの実際の戻り行数の最高水準点です。

サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返し

ます。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了したサービス・サブクラスの rows_returned_top 最高水準点のみが更新されます。アクティビティーがマップされたサービス・サブクラスでも、アクティビティーがそこで完了しなかった場合、そのサービス・サブクラスの最高水準点は何も影響を受けません。

表 181. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーで到達した DML アクティビティーの実際の戻り行数の最大数を調べることができます。

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

このアクティビティーがサービス・クラス有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスが所属する作業クラス・セットに関連した作業アクション・セットの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 182. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 183. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントと **sc_work_class_id** エレメントを組み合わせて使用すると、アクティビティーのサービス・クラス作業クラスが存在する場合にはそれを一意的に識別できます。

sc_work_class_id サービス・クラス作業クラス ID : モニター・エレメント

このアクティビティーがサービス・クラス有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、このアクティビティーに割り当てられた作業クラスの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 184. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 185. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントと **sc_work_action_set_id** エレメントを組み合わせて使用すると、アクティビティーのサービス・クラス作業クラスが存在する場合にはそれを一意的に識別できます。

section_env セクション環境 : モニター・エレメント

SQL ステートメントのセクションを含む BLOB。これは、実際のセクション内容で、照会プランの実行可能形式です。

表 186. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	常に収集される
パッケージ・キャッシュ	pkgcache	COLLECT DETAILED DATA

使用法

セクション Explain のプロシージャーと一緒にこのエレメントを使用すると、ステートメントを Explain して、ステートメントのアクセス・プランを表示させることができます。

service_class_id サービス・クラス ID : モニター・エレメント

サービス・サブクラスのユニーク ID。作業単位の場合、この ID は、その作業単位を発行している接続が関連付けられているワークロードのサービス・サブクラス ID を表します。

表 187. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・クラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE _ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 188. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (metrics 文書に報告されます)	REQUEST METRICS BASE
ロッキング	-	常に収集される
作業単位	-	常に収集される
統計	event_histogrambin	常に収集される
統計	event_scstats	常に収集される

使用法

このエレメントの値は、ビュー SYSCAT.SERVICECLASSES の列 SERVICECLASSID の値と一致します。このエレメントを使用して、サービス・サブクラス名、または別のソースのサービス・サブクラスに関するリンク情報を検索します。例えば、サービス・クラス統計をヒストグラム・ビン・レコードと結合させます。

以下の条件が満たされている場合、このエレメントの値は 0 になります。

- このエレメントが、event_histogrambin 論理データ・グループでレポートされる。
- ヒストグラム・データが、サービス・クラスではないオブジェクトに関して収集される。

service_subclass_name サービス・サブクラス名 : モニター・エレメント

サービス・サブクラスの名前。

表 189. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE

表 190. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
ロッキング	-	常に収集される
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
統計	event_scstats	常に収集される
統計	event_qstats	常に収集される

使用法

このエレメントを他のアクティビティー・エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと

一緒に使用すると、サービス・クラスまたはしきい値キューの動作の分析をすることができます。

service_superclass_name サービス・スーパークラス名 : モニター・エレメント

サービス・スーパークラスの名前。

表 191. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_SUPERCLASS_STATS 表関数 - サービス・スーパークラスの統計を戻す	ACTIVITY METRICS BASE

表 192. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
統計	event_scstats	常に収集される
統計	event_qstats	常に収集される

使用法

このエレメントを他のアクティビティー・エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと一緒に使用すると、サービス・クラスまたはしきい値キューの動作の分析をすることができます。

source_service_class_id ソース・サービス・クラス ID : モニター・エレメント

このエレメントのしきい値違反レコードが生成された時に、アクティビティーから再マップしたサービス・サブクラスの ID。 しきい値アクションが REMAP ACTIVITY アクション以外の場合、このエレメントの値はゼロです。

表 193. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントは、アクティビティーが再マップされたサービス・クラスをたどるのに使用できます。 これを使用して、特定のサービス・サブクラスからマップされたアクティビティー数の総計を計算することもできます。

statistics_timestamp 統計タイム・スタンプ : モニター・エレメント

この統計レコードが生成された時刻。

表 194. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-
統計	event_wcstats	-
統計	event_qstats	-
統計	event_histogrambin	-

使用法

このエレメントを使用すると、この統計レコードが生成された時点を判別できます。

このエレメントと **last_wlm_reset** エレメントを組み合わせて使用すると、この統計レコードの統計が生成された時間間隔を識別できます。

このモニター・エレメントを使用すると、同じ収集間隔において生成されたすべての統計レコードをグループ化することもできます。

stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント

ルーチンの呼び出しを、作業単位内の同じネスト・レベルにある他のものと区別する ID。これは、作業単位内の特定のネスト・レベルで固有です。

表 195. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 196. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
作業単位	パッケージ・リストに報告されます。	-

1 このオプションは推奨されなくなりました。このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。ロック・タイムアウト、ロック待機、デッドロックなどのロック関連イベントをモニターするには、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用してください。

使用法

このエレメントを使用して、特定の SQL ステートメントが実行された呼び出しを一意に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント

temp_tablespace_top モニター・エレメントは、サービス・クラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの TEMPORARY 表スペース使用量 (KB 単位) の最高水準点です。

サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了したサービス・サブクラスの temp_tablespace_top 最高水準点のみが変更されます。 アクティビティーがマップされたサービス・サブクラスでも、アクティビティーがそこで完了しなかった場合、そのサービス・サブクラスの最高水準点は何も影響を受けません。

表 197. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のメンバーで到達した DML アクティビティーの SYSTEM TEMPORARY 表スペース使用量の最大値を判別することができます。

このエレメントは、適用される TEMPORARY 表スペースのしきい値があるアクティビティーによってのみ更新されます。アクティビティーに TEMPORARY 表スペースのしきい値が適用されない場合、0 の値が返されます。

thresh_violations - しきい値違反の回数 : モニター・エレメント

しきい値が違反された回数。

このモニター・エレメントは、スナップショット・モニター・ルーチンおよびデータベース・イベント・モニターによって戻される、 491 ページの『num_threshold_violations しきい値違反の回数 : モニター・エレメント』 モニター・エレメントの別名です。

表 198. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 198. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 199. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (metrics 文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

このエレメントを使用して、違反された WLM しきい値があるかどうかを迅速に判別できます。しきい値が違反された場合、しきい値の違反イベント・モニター (すでに作成済みでアクティブな場合) を使用してしきい値違反に関する詳細を入手できます。

一例として、どのしきい値が違反されたのか詳細を入手できます。

threshold_action しきい値アクション : モニター・エレメント

このしきい値違反レコードが適用されるしきい値のアクション。可能な値は「停止」、「続行」および「再マップ」です。

表 200. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値を違反したアクティビティーが、違反が起きた時点で停止したか、実行を継続できたか、それとも他のサービス・サブクラスに再マップされたかを判別できます。 アクティビティーが停止された場合は、そのアクティビティーをサブミットしたアプリケーションは SQL4712N エラーを受け取ることになります。アクティビティーが他のサービス・サブクラスに再マップされた場合、メンバー上でアクティビティー用に作動しているエージェントはしきい値のターゲット・サービス・サブクラスに移動します。

threshold_domain しきい値ドメイン : モニター・エレメント

このキューに関係するしきい値のドメイン。

可能な値は以下のとおりです。

- データベース
- 作業アクションセット
- サービス・スーパークラス
- サービス・サブクラス
- ワークロード

表 201. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 202. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、述部は同じでもドメインが異なるしきい値のキュー統計を区別することができます。

threshold_maxvalue しきい値最大値 : モニター・エレメント

このモニター・エレメントは、キューイング非対象しきい値においては、このしきい値を超えてしまった値を表します。キューのしきい値の場合は、このモニター・エレメントは、キューイングの原因となった並行性のレベルを表します。

キューのしきい値の違反の原因となった並行性のレベルは、**threshold_maxvalue** および **threshold_queuesize** モニター・エレメントの合計です。

表 203. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	常に収集される

使用法

アクティビティーしきい値では、このエレメントは、しきい値の違反が発生した時点でのしきい値の最大値の履歴レコードを提供します。これは、違反の発生以降にしきい値の最大値が変更され、古い値が SYSCAT.THRESHOLDS ビューで表示できなくなった場合に便利です。DATATAGINSC IN および DATATAGINSC NOT IN しきい値の場合、このエレメントには、しきい値に違反したデータ・タグ値が入っています。

threshold_name しきい値名 : モニター・エレメント

このキューに関係するしきい値の固有の名前。

表 204. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 205. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、このレコードが示す統計の元となるキューのしきい値を一意的に識別できます。

threshold_predicate しきい値述部 : モニター・エレメント

違反したしきい値または統計の収集の対象となったしきい値のタイプを識別します。

表 206. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 207. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	常に収集される
統計	event_qstats	常に収集される

使用法

このモニター・エレメントを他の統計またはしきい値違反モニター・エレメントと一緒に使用すると、しきい値違反の分析をすることができます。

event_thresholdviolations 論理グループでレポートされる場合のこのモニター・エレメントの有効な値は、次のとおりです。

AggSQLTempSpace

SQLTempSpace

SQLRowsReturned

ActivityTotalTime

EstimatedSQLCost

TotalMemberConnections

ConnectionIdleTime

ConcurrentWorkloadOccurrences

ConcurrentWorkloadActivities

ConcurrentDBCoordActivities

TotalSCMemberConnections

SQLRowsRead

SQLRowsReadInSC

CPUTime

CPUTimeInSC

UowTotalTime

DataTagInSC

DataTagNotInSC

event_qstats 論理グループでレポートされる場合のこのモニター・エレメントの有効な値は、次のとおりです。

TotalMemberConnections

ConcurrentDBCoordActivities

TotalSCMemberConnections

threshold_queuesize しきい値キュー・サイズ : モニター・エレメント

キューのしきい値におけるキューのサイズ。このサイズを超えようとすると、しきい値違反が発生します。キューのしきい値以外では、この値は 0 です。

表 208. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値の違反が発生した時点でのこのしきい値のキューにおけるアクティビティーまたは接続の数を判別できます。

thresholdid しきい値 ID : モニター・エレメント

しきい値違反レコードを適用するしきい値か、キュー統計の収集対象のしきい値を識別します。

表 209. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE

表 210. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-
統計	event_qstats	-

使用法

このモニター・エレメントを他のアクティビティー履歴モニター・エレメントと一緒に使用すると、しきい値キューの分析またはしきい値に違反したアクティビティーの分析をすることができます。

time_completed 完了時刻 : モニター・エレメント

このアクティビティー・レコードにより記述されているアクティビティーが実行を完了した時刻。このエレメントは、ローカル・タイム・スタンプです。

表 211. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

メモリーの制約のためにアクティビティー・レコード全体を表イベント・モニターに書き込むことができなかった場合、このフィールドの値は「0000-00-00-00.00.00.000000」になります。進行中のアクティビティーがキャプチャーされた場合、このフィールドは、アクティビティーが収集された時間を表します。

time_created 作成時刻 : モニター・エレメント

ユーザーが、このアクティビティー・レコードにより記述されているアクティビティーをサブミットした時刻。このエレメントは、ローカル・タイム・スタンプです。

表 212. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

time_of_violation 違反時刻 : モニター・エレメント

このしきい値違反レコードに記述されているしきい値違反が発生した時刻。このエレメントは、ローカル・タイム・スタンプです。

表 213. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを他のしきい値違反モニター・エレメントと一緒に使用すると、しきい値違反の分析をすることができます。

time_started 開始時刻 : モニター・エレメント

このアクティビティー・レコードにより記述されているアクティビティーが実行を開始した時刻。このエレメントは、ローカル・タイム・スタンプです。

表 214. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

アクティビティーが拒否された場合、**act_exec_time** モニター・エレメントの値は 0 です。この場合、**time_started** モニター・エレメントの値は **time_completed** モニター・エレメントの値に等しくなります。

top ヒストグラム・ビンの最上位 : モニター・エレメント

ヒストグラム・ビンの範囲の包括的最上端。このモニター・エレメントの値は、次のヒストグラム・ビンの範囲の排他的最下端でもあります。

表 215. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントと対応する **bottom** エレメントを一緒に使用して、ヒストグラム中のビンの範囲を判別します。

total_disp_run_queue_time - ディスパッチャーの合計実行キュー時間 : モニター・エレメント

このサービス・クラスで実行された要求が CPU にアクセスするために待機していた合計時間。この値はマイクロ秒単位で示されます。

表 216. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 216. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 217. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
統計	event_scstats (metrics 文書に報告されます)	REQUEST METRICS BASE

表 217. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (metrics 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

使用法

total_disp_run_queue_time モニター・エレメントを、**total_cpu_time** と一緒に使用すると、CPU リソースの競合度合いを計算できます。その際、スケール 0 から 1 の範囲で測定し、数値が小さいほど、CPU リソースの競合が大きいことを意味します。CPU 速度と呼ばれるこの指標は、CPU にアクセスしてサービス・クラスで作業していた時間を、CPU にアクセスしていた、または CPU にアクセスするために待機していた合計時間で除算して算出します。この指標によって、作業が CPU をまったく待機しないで実行された場合の効率に比べ、どれほど効率的に実行されているかを測定することができます。数式は、次のとおりです。

```
CPU velocity = total_cpu_time / (total_cpu_time + total_disp_run_queue_time)
```

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降のディスパッチャー実行キューの合計待機時間を表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降のディスパッチャー実行キューの合計待機時間を表します。

uow_completed_total - 完了済みの合計作業単位 : モニター・エレメント

コミットまたはロールバックされて完了した作業単位の合計数。

表 218. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 219. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (metrics 文書に報告されます)	常に収集される
統計	event_wlstats (metrics 文書に報告されます)	常に収集される

使用法

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降に完了した作業単位の合計数を表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降に完了した作業単位の合計数を表します。

uow_comp_status 作業単位完了状況

作業単位の状況およびそれが停止したときの状況。

エレメント ID

uow_comp_status

エレメント・タイプ

情報

表 220. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位
DCS アプリケーション	dcs_appl	基本

表 221. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	常に収集される

使用法 このエレメントを使用すると、作業単位が終了した原因がデッドロックによるものか、または異常終了によるものかを判別できます。次の原因が考えられます。

- コミット・ステートメントによりコミットされた。
- ロールバック・ステートメントによりロールバックされた。
- デッドロックによりロールバックされた。
- 異常終了によりロールバックされた。
- アプリケーションの正常終了によりコミットされた。
- 進行中であった作業単位に対する FLUSH EVENT MONITOR コマンドの結果が不明。

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル (*sqlmon.h*) を参照してください。

uow_elapsed_time 最新の作業単位の経過時間

最後に完了した作業単位の実行経過時間。

エレメント ID

uow_elapsed_time

エレメント・タイプ

time

表 222. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

使用法

作業単位の完了にかかる時間の標識として、このエレメントを使用します。

このエレメントは、秒およびマイクロ秒 (100 万分の 1 秒) の単位で消費時間を報告する 2 つのサブエレメントで構成されています。このモニター・エレメントの名前に「_s」と「_ms」を追加したものがサブエレメントの名前になります。このモニター・エレメントの消費時間の合計を取得するには、2 つのサブエレメントの値を合計する必要があります。例えば、「_s」サブエレメントの値が 3 で、「_ms」サブエレメントの値が 20 の場合、モニター・エレメントの消費時間の合計は 3.00002 秒です。

uow_id 作業単位 ID : モニター・エレメント

作業単位の ID。作業単位 ID は、アプリケーション・ハンドル内で固有です。

表 223. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数 - アクティビティーのリストを戻す	ACTIVITY METRICS BASE

表 224. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される
アクティビティー	event_activitystmt	常に収集される
アクティビティー	event_activityvals	常に収集される
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
しきい値違反	event_thresholdviolations	常に収集される
変更履歴	ddlstmtexec txncompletion	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

さらにこのエレメントを **activity_id** および **appl_id** モニター・エレメントと一緒に使用すると、アクティビティーを一意的に識別できます。

uow_lifetime_avg - 作業単位の平均存続期間 : モニター・エレメント

作業単位の平均存続期間。ミリ秒で計測されます。

表 225. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 226. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (metrics 文書に報告されます)	常に収集される

表 226. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (metrics 文書に報告されます)	常に収集される

使用法

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降の作業単位の平均存続期間を示します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降の作業単位の平均存続期間を表します。

uow_lock_wait_time - ロック待機中の作業単位の合計時間 : モニター・エレメント

この作業単位がロックの待機に要した合計経過時間。値はミリ秒単位で示されます。

エレメント ID
uow_lock_wait_time

エレメント・タイプ
カウンター

表 227. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位

使用法 このエレメントは、リソース競合問題の重大度を判別するときに利用できます。

uow_log_space_used - 使用されている作業単位ログ・スペース: モニター・エレメント

モニター対象のアプリケーションで現行作業単位に使用されているログ・スペースの量 (バイト単位)。

表 228. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 229. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位

表 230. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	常に収集される
作業単位	-	常に収集される

使用法

このエレメントを使用すると、作業単位レベルでのロギングの所要量を把握することができます。

uow_start_time - 作業単位開始タイム・スタンプ : モニター・エレメント

作業単位が最初にデータベース・リソースを要求した日時。

表 231. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 232. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

表 233. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
トランザクション	event_xact	-

使用法

このリソース要求は、その作業単位で SQL ステートメントを初めて実行したときに発生します。

- 最初の作業単位の場合は、**conn_complete_time** の後の最初のデータベース要求 (SQL ステートメントの実行) の時刻。
- その後の作業単位の場合は、前回の COMMIT または ROLLBACK の後の最初のデータベース要求 (SQL ステートメントの実行) の時刻。

注: 「*SQL リファレンス*」は、作業単位の境界を COMMIT または ROLLBACK のポイントとして定義します。

データベース・システム・モニターでは、 COMMIT/ROLLBACK とその作業単位定義から出される次の SQL ステートメントまでの経過時間を除外します。この測定方式により、データベース・マネージャーがデータベース要求の処理に要する時間を、その作業単位の最初の SQL ステートメント以前にアプリケーション・ロジック内で要する時間とは切り離して反映します。作業単位の経過時間には、作業単位内で SQL ステートメント間のアプリケーション・ロジックを実行する時間が含まれます。

このエレメントと **uow_stop_time** モニター・エレメントを組み合わせると、作業単位の合計経過時間を計算できます。**prev_uow_stop_time** モニター・エレメントと組み合わせると、作業単位間にアプリケーションで要した時間を計算できます。

uow_stop_time と **prev_uow_stop_time** モニター・エレメントを組み合わせると、*SQL リファレンス*の定義による作業単位の経過時間を計算できます。

uow_status 作業単位の状況

作業単位の状況。

エレメント ID

uow_status

エレメント・タイプ

情報

表 234. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	常に収集される

使用法 このエレメントを使用すると、作業単位の状況を判別できます。 API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル sqlmon.h を参照してください。

uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント

最新の作業単位が完了した日時。これが起こるのはデータベースの変更がコミットまたはロールバックされたときです。

表 235. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

表 236. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-

使用法

このエレメントと **prev_uow_stop_time** モニター・エレメントを組み合わせて使用すると、COMMIT/ROLLBACK ポイント間の合計経過時間を計算できます。**uow_start_time** モニター・エレメントと組み合わせると、前回の作業単位の経過時間を計算できます。

タイム・スタンプの内容は、次のように設定されます。

- アプリケーションが 1 つの作業単位を完了し、(**uow_start_time** モニター・エレメントで定義されたように) 新しい作業単位をまだ開始していない場合、このエレメントは有効な非ゼロタイム・スタンプをレポートします。
- アプリケーションが作業単位を実行中の場合は、このエレメントがゼロをレポートします。
- アプリケーションがデータベースに初めて接続すると、このエレメントは **conn_complete_time** モニター・エレメントの値に設定されます。

新しい作業単位が開始すると、このエレメントの内容は、**prev_uow_stop_time** モニター・エレメントに移動します。

uow_throughput - 作業単位スループット : モニター・エレメント

1 秒単位の作業単位数で計測される、作業単位完了数。

表 237. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_SAMPLE_SERVICE_CLASS_METRICS - サービス・サブクラスのメトリックのサンプルの取得	REQUEST METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - ワークロードのメトリックのサンプルの取得	REQUEST METRICS BASE
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 238. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (metrics 文書に報告されます)	常に収集される
統計	event_wlstats (metrics 文書に報告されます)	常に収集される

使用法

このモニター・エレメントが WLM_GET_SERVICE_SUBCLASS_STATS 関数または WLM_GET_WORKLOAD_STATS 関数によって戻される場合、統計を最後にリセットして以降の作業単位スループットを表します。

このモニター・エレメントが MON_SAMPLE_SERVICE_CLASS_METRICS 関数または MON_SAMPLE_WORKLOAD_METRICS 関数によって戻される場合、関数が実行されて以降の作業単位スループットを表します。

uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント

作業単位の存続時間の最高水準点 (ミリ秒)。

表 239. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_SERVICE_SUBCLASS_STATS 表関数 - サービス・サブクラスの統計を戻す	REQUEST METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	REQUEST METRICS BASE

表 240. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	常に収集される
統計	event_scstats	常に収集される

使用法

このエレメントを使用すると、UOWTOTALTIME しきい値が有効であるかどうか、また、そのようなしきい値の構成方法を判別する助けになります。

サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、この最高水準点はワークロードが割り当てたサービス・クラス用に計算されます。アクティビティーのサービス・クラスを変更する作業アクション・セットのどのマッピングもこの最高水準点に影響しません。

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

このアクティビティーがワークロード有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスが所属する作業クラス・セットに関連した作業アクション・セットの ID を示します。それ以外の場合、このモニター・エレメントは 0 の値を示します。

表 241. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 242. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このモニター・エレメントを **wl_work_class_id** モニター・エレメントと組み合わせて使用すると、アクティビティーのワークロード作業クラスが存在する場合にはそれを一意的に識別できます。

wl_work_class_id - ワークロード作業クラス ID : モニター・エレメント

このアクティビティーがワークロード有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 243. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 244. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	常に収集される

使用法

このモニター・エレメントを **wl_work_action_set_id** モニター・エレメントと組み合わせて使用すると、アクティビティーのワークロード作業クラスが存在する場合にはそれを一意的に識別できます。

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

アクティビティーまたは接続が WLM しきい値によってキューに入れられた回数。

表 245. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 246. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

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

WLM キューイングしきい値の待機にかかった時間。この値はミリ秒単位で示されます。

表 247. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_FORMAT_XML_METRICS_BY_ROW - すべてのメトリックに関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_TIMES_BY_ROW - フォーマット設定された行ベースの待機/処理時間の結合された階層を取得する	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_FORMAT_XML_WAIT _TIMES_BY_ROW - 待機時間に関するフォーマット設定された行ベースの出力の取得	適用外: フォーマット関数への入力として与えられた XML 文書内に含まれているエレメントをすべて報告する
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 247. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_GET_PKG_CACHE_STMT_DETAILS 表関数 - パッケージ・キャッシュ項目の詳細メトリックの取得	ACTIVITY METRICS BASE
MON_GET_ROUTINE 表関数 - ルーチンの集約された実行メトリックの取得	常に収集される
MON_GET_ROUTINE_DETAILS 表関数 - ルーチンの集約された実行メトリックの詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラスのメトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位の詳細メトリックの取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 248. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
アクティビティー	event_activitymetrics	ACTIVITY METRICS BASE
統計	event_scstats (メトリック文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (メトリック文書に報告されます)	REQUEST METRICS BASE
パッケージ・キャッシュ	activity_metrics 文書に報告されます。	ACTIVITY METRICS BASE
作業単位	system_metrics 文書に報告されます。	REQUEST METRICS BASE

wlo_completed_total 完了したワークロード・オカレンスの合計 : モニター・エレメント

最後にリセットしてから完了するワークロード・オカレンスの数。

表 249. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 250. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-

使用法

このエレメントを使用すると、処理をシステムに移動させている特定のワークロードのオカレンスの数を判別できます。

work_action_set_id 作業アクション・セット ID : モニター・エレメント

この統計レコードが適用される作業アクション・セットの ID。

表 251. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	常に収集される
統計	event_wcstats	常に収集される

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと一緒に使用すると、作業クラスの動作の分析をすることができます。

以下の条件が満たされている場合、このエレメントの値は 0 になります。

- このエレメントが、event_histogrambin 論理データ・グループでレポートされる。
- ヒストグラム・データが、処理クラスではないオブジェクトに関して収集される。

work_action_set_name 作業アクション・セット名 : モニター・エレメント

このイベントの一部として示された統計が関連付けられた作業アクション・セットの名前。

表 252. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE

表 253. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-
統計	event_wcstats	-

使用法

このエレメントと **work_class_name** エレメントを組み合わせて使用すると、統計がこのレコードに示されている作業クラスを一意的に識別したり、統計がこのレコードに示されているしきい値キューのドメインである作業クラスを一意的に識別できます。

work_class_id 作業クラス ID : モニター・エレメント

この統計レコードが適用される作業クラスの ID。

表 254. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	常に収集される
統計	event_histogrambin	常に収集される

使用法

このエレメントを他の統計エレメントと一緒に使用すると、作業クラスの分析をすることができます。

以下の条件が満たされている場合、このエレメントの値は 0 になります。

- このエレメントが、event_histogrambin 論理データ・グループでレポートされる。
- ヒストグラム・データが、処理クラスではないオブジェクトに関して収集される。

work_class_name 作業クラス名 : モニター・エレメント

このイベントの一部として示された統計が関連付けられた作業クラスの名前。

表 255. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_WORK_ACTION_SET_STATS 表関数 - 作業アクション・セット統計を戻す	ACTIVITY METRICS BASE

表 256. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-
統計	event_wcstats	-

使用法

このエレメントと **work_action_set_name** エレメントを組み合わせて使用すると、統計がこのレコードに示されている作業クラスを一意的に識別したり、統計がこのレコードに示されているしきい値キューのドメインである作業クラスを一意的に識別できます。

workload_id ワークロード ID : モニター・エレメント

ワークロードを一意的に識別する整数。

表 257. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得	ACTIVITY METRICS BASE

表 258. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本

表 259. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (metrics 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (metrics 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	常に収集される
統計	event_wlstats	常に収集される

表 259. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	常に収集される
アクティビティー	event_activity	常に収集される
しきい値違反	event_thresholdviolations	常に収集される

使用法

この ID を使用して、このアクティビティー、アプリケーション、ヒストグラム・ビン、またはワークロード統計レコードが所属するワークロードを一意的に識別します。

以下の条件が満たされている場合、このエレメントの値は 0 になります。

- このエレメントが、event_histogrambin 論理データ・グループでレポートされる。
- ワークロードではないオブジェクトのヒストグラム・データが収集される。

workload_name ワークロード名 : モニター・エレメント

ワークロードの名前。

表 260. 表関数モニター情報

表関数	モニター・エレメントの収集レベル: (モニター・エレメントの収集レベルについて詳しくは、モニター・エレメントの収集レベルを参照してください。)
MON_FORMAT_LOCK_NAME 表関数 - 内部ロック名のフォーマット設定と詳細の出力	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	ACTIVITY METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	ACTIVITY METRICS BASE
MON_SAMPLE_WORKLOAD_METRICS - サンプルの取得	ACTIVITY METRICS BASE
WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE
WLM_GET_WORKLOAD_STATS 表関数 - ワークロード統計を戻す	ACTIVITY METRICS BASE

表 261. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	常に収集される
作業単位	-	常に収集される
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (metrics 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (metrics 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	常に収集される
統計	event_wlstats	常に収集される

使用法

統計イベント・モニターおよびワークロードの表関数において、ワークロード名は統計またはメトリックが収集されたり報告されるワークロードを識別します。 作業単位イベント・モニターおよび作業単位表関数の場合、ワークロード名は作業単位が関連付けられたワークロードを識別します。

ワークロード名を使用して、特定のワークロードに関する作業単位や情報のセットを識別します。

workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント

このアクティビティーが所属するワークロード・オカレンスの ID。

表 262. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_AGENTS 表関数 - サービス・クラスで実行中のエージェントのリスト	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

表 263. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	常に収集される
アクティビティー	event_activity	常に収集される

使用法

これを使用すると、アクティビティーをサブミットしたワークロード・オカレンスを識別できます。

workload_occurrence_state - ワークロード・オカレンスの状態 : モニター・エレメント

ワークロード・オカレンスの状態。

表 264. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	ACTIVITY METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数 - ワークロード・オカレンスのリスト	ACTIVITY METRICS BASE

使用法

可能な値は以下のとおりです。

DECOUPLED
ワークロード・オカレンスにコーディネーター・エージェントが割り当てられていません (コンセントレーターのケース)。

DISCONNECTPEND
ワークロード・オカレンスはデータベースから切断中です。

FORCED
ワークロード・オカレンスは強制されました。

INTERRUPTED
ワークロード・オカレンスは中断されました。

QUEUED
ワークロード・オカレンス・コーディネーター・エージェントが、ワークロード管理キューイングしきい値によってキューに入れられています。パーティション・データベース環境では、この状態は、コーディネーター・エージェントがしきい値チケットを取得するために別のメンバーに対して RPC を行ったものの、まだ応答を受け取っていないことを示している可能性があります。

TRANSIENT
ワークロード・オカレンスは、サービス・スーパークラスにまだマップされていません。

UOWEXEC
ワークロード・オカレンスは要求を処理中です。

UOWWAIT
ワークロード・オカレンスはクライアントからの要求を待機中です。

コマンド

SET WORKLOAD

データベース接続の接続先として割り当てるワークロードを指定します。このコマンドは、データベースに接続する前に発行することができます。あるいは、接続が確立されてから現行接続の再割り当てをするために使用することができます。接続が確立されている場合は、次の作業単位の開始時にワークロードの再割り当てが行われます。

許可

なし。ただし、『使用上の注意』を参照してください。

必要な接続

なし

コマンド構文

コマンド・パラメーター

AUTOMATIC
サーバーが自動的に実行するワークロード計算に選ばれているワークロードに、データベース接続を割り当てるよう指定します。

SYSDEFAULTADMWORKLOAD
データベース接続を SYSDEFAULTADMWORKLOAD に割り当てて、*accessctrl* 権限、*dataaccess* 権限、*wlmadm* 権限、*secadm* 権限、または *dbadm* 権限を持つユーザーが通常のワークロード計算を迂回できるよう指定します。

例

接続を **SYSDEFAULTADMWORKLOAD** に割り当てる方法。

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD
```

ワークロード割り当てをリセットして、サーバーが実行するワークロード計算に選ばれているワークロードを使用するようにする方法。

```
SET WORKLOAD TO AUTOMATIC
```

使用上の注意

データベース接続の SESSION 許可 ID に *accessctrl* 権限、*dataaccess* 権限、*wlmadm* 権限、*secadm* 権限、または *dbadm* 権限がない場合、接続を **SYSDEFAULTADMWORKLOAD** に割り当てることはできず、SQL0552N エラーが戻されます。**SET WORKLOAD TO SYSDEFAULTADMWORKLOAD** コマンドがデータベース接続前に発

行される場合、データベース接続が確立された後の、次の作業単位の開始時に SQL0552N エラーが戻されます。データベース接続が確立されているときにコマンドが発行された場合は、ワークロードの再割り当てが実行される予定の、次の作業単位の開始時に SQL0552N エラーが戻されます。

構成パラメーター

wlm_collect_int - ワークロード管理収集間隔構成パラメーター

このパラメーターは、ワークロード管理 (WLM) 統計の収集およびリセットの間隔を分単位で指定します。

x 分ごと (*x* は **wlm_collect_int** パラメーターの値) に、すべてのワークロード管理統計が収集されて、任意のアクティブな統計イベント・モニターに送信され、その後で統計がリセットされます。アクティブな統計イベント・モニターが存在する場合は、それが作成された方法に応じて、統計はファイル、パイプ、または表のいずれかに書き込まれます。アクティブなイベント・モニターが存在しない場合は、統計はリセットのみされて、収集はされません。

収集は、日曜日の 00:00:00 を基準として測定される、指定の間隔で行われます。カタログ・メンバーがアクティブになると、この定刻を基準として次にスケジュールされた間隔が開始するときに、次の収集が行われます。スケジュールされた間隔は、カタログ・メンバーがアクティブになった時間を基準とはしません。収集の時刻にメンバーがアクティブになっていない場合、そのメンバーの統計は収集されません。例えば、間隔の値が 60 に設定され、カタログ・メンバーが日曜日の午前 9:24 にアクティブ化された場合、収集が毎時正時に行われるようにスケジュールされます。つまり、次の収集は午前 10:00 に行われます。メンバーが午前 10:00 の時点でアクティブになっていない場合、そのメンバーの統計は収集されません。

収集とリセットのプロセスは、カタログ・メンバーから開始されます。カタログ・メンバーでは、**wlm_collect_int** パラメーターを指定する必要があります。これは、他のメンバーでは使用されません。

構成タイプ

データベース

パラメーター・タイプ

- オンラインで構成可能

デフォルト [範囲]

0 [0 (収集は実行されない), 5 - 32 767]

統計イベント・モニターにより収集されるワークロード管理統計は、システムの短期および長期の動作をモニターするために使用できます。短い間隔を使用して、システムの短期および長期の動作をモニターすることができます。これは、その結果をマージすると、長期の動作を取得できるためです。ただし、異なる間隔で得られた結果を手動でマージすると、分析が複雑になります。短い間隔の統計が必要なければ、処理時間が無用に増大するだけです。したがって、長期の動作の分析だけで十分な場合は、短期の動作をキャプチャーする間隔を下げ、処理時間を削減する間隔を上げます。

この間隔は SQL 要求、コマンドの呼び出し、またはアプリケーションごとではなく、データベースごとにカスタマイズする必要があります。他の構成パラメーターを考慮する必要はありません。

注: すべての WLM 統計表関数は、統計が前回リセットされてから累積した統計を戻します。この統計は、この構成パラメーターで指定された間隔で定期的にリセットされます。

wlm_dispatcher - ワークロード管理ディスパッチャー

このパラメーターは、DB2 ワークロード管理ディスパッチャーを使用可能 (YES) または使用不可 (NO) にします。デフォルトでは、ディスパッチャーを使用可能にすると、CPU リミットを設定できます。

構成タイプ
データベース・マネージャー

適用

- ローカルおよびリモート・クライアントを持つデータベース・サーバー
- ローカル・クライアントを持つデータベース・サーバー
- ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
即時

デフォルト [範囲]
NO [NO; YES]

DB2 データベース・マネージャーのアップグレードの際、**wlm_dispatcher** データベース・マネージャー構成パラメーターの値は NO に設定されます。

ワークロード管理ディスパッチャーは、CPU リソースまたは CPU リミット、あるいはその両方の共有ベースの割り振りを使用して、DB2 データベース・マネージャーのサービス・クラス・レベルの CPU スケジューリング機能を提供します。

ワークロード管理ディスパッチャーを使用可能にすると、ユーザー・サービス・クラスおよび保守サービス・クラスで実行されているすべての作業がディスパッチャーの制御下に置かれます。これが使用可能になると、デフォルト・ケースとして CPU リミット設定がディスパッチャーによって適用されます。共有ベースの CPU リソース割り振りを使用するためには、**wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターを使用可能にする必要があります。

wlm_dispatcher 構成パラメーターを YES に設定する場合、以下の条件が適用されます。

- いずれかのサービス・クラスのエージェント優先順位をデフォルト以外の任意の値に設定する場合、データベースを活動化するときに警告メッセージが db2diag ログおよび管理通知ログに書き込まれます。

- サービス・クラスを作成または変更してエージェント優先順位をデフォルト値以外の値に設定しようとすると、サービス・クラスを作成または変更するためにステートメントを発行したアプリケーションに警告が返されます。

wlm_disp_concur - ワークロード・マネージャー・ディスパッチャー・スレッド並行性

このパラメーターは、DB2 ワークロード・マネージャー (WLM) ディスパッチャーがスレッド並行性レベルを設定する方法を指定します。スレッド並行性レベルを手動で固定値に設定することもできます。

構成タイプ
データベース・マネージャー

適用

- ローカルおよびリモート・クライアントを持つデータベース・サーバー
- ローカル・クライアントを持つデータベース・サーバー
- ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
即時

デフォルト [範囲]
COMPUTED [COMPUTED; *manually_set_value*]

DB2 データベース・マネージャーをアップグレードする際の **wlm_disp_concur** データベース・マネージャー構成パラメーターの値は COMPUTED です。

COMPUTED
DB2 データベース・マネージャーは、DB2 データベース・マネージャーで使用可能な論理 CPU の数の 4 倍の値に基づいて固定スレッド並行性レベルを計算します。

manually_set_value
スレッド並行性レベルを手動で固定値 (1 - 32767) に設定することができます。最適値は、使用される特定のハードウェアおよびオペレーティング・システム・レベルによって異なります。一般に、ホストまたは LPAR の論理 CPU の数の 2 倍から 4 倍の範囲です。

単位 並行スレッドの数

このデータベース・マネージャー構成パラメーターの設定は、オペレーティング・システムの実行キューへ並行してディスパッチすることが WLM ディスパッチャーによって許可されるスレッドの数を制御します。値は、DB2 データベース・マネージャーで使用可能な論理 CPU の数に小さい数を乗じた倍数として設定されます。一般的には、スレッドがアクティブ状態に切り替わるとき、およびアクティブ状態が解除されるときに発生する可能性のあるスケジューリング待ち時間を考慮に入れて、使用可能な論理 CPU の数の 4 倍に値を設定することができます。最適値は、ホストまたは LPAR の CPU を DB2 データベース・マネージャーが十分使用する

ために必要な数のスレッドをちょうど確保するに足りるだけの大きさで、それを超えない値です。この最適値により、効率が最大化され、DB2 WLM ディスパッチャーの CPU 割り振りに対する制御も最大化されます。

wlm_disp_cpu_shares - ワークロード・マネージャー・ディスパッチャーの CPU シェア

このパラメーターは、DB2 ワークロード・マネージャー (WLM) ディスパッチャーによる CPU シェアの制御を使用可能 (YES) または使用不可 (NO) にします。デフォルトで、使用可能な WLM ディスパッチャーは CPU リミットのみを制御します。

構成タイプ
データベース・マネージャー

適用

- ローカルおよびリモート・クライアントを持つデータベース・サーバー
- ローカル・クライアントを持つデータベース・サーバー
- ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
即時

デフォルト [範囲]
NO [NO; YES]

DB2 データベース・マネージャーをアップグレードする際の **wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターの値は NO です。

wlm_dispatcher データベース・マネージャー構成パラメーターの値を YES に設定し、**wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターの値を NO に設定する場合、WLM ディスパッチャーは CPU リミットのみをサービス・クラスの管理に適用できます。

wlm_dispatcher データベース・マネージャー構成パラメーターの値を YES に設定し、**wlm_disp_cpu_shares** データベース・マネージャー構成パラメーターの値を YES に設定する場合、WLM ディスパッチャーは CPU リミットと CPU シェアの両方をサービス・クラスの管理に適用できます。デフォルトでは、CPU リソースが均等に分割されるように、すべてのサービス・クラスにハード CPU シェアとして 1000 が割り当てられます。

表 265. DB2 WLM ディスパッチャーによるサービス・クラス管理に必要なデータベース・マネージャー構成パラメーター設定のサマリー

サービス・クラス管理	wlm_dispatcher の設定	wlm_disp_cpu_shares の設定
None	NO	NO
CPU リミット	YES	NO

表 265. DB2 WLM ディスパッチャーによるサービス・クラス管理に必要なデータベース・マネージャー構成パラメーター設定のサマリー (続き)

サービス・クラス管理	wlm_dispatcher の設定	wlm_disp_cpu_shares の設定
CPU リミット + CPU シェア	YES	YES

wlm_disp_min_util - ワークロード・マネージャー・ディスパッチャー最小 CPU 使用率

このパラメーターは、DB2 WLM によって管理される CPU リソースの共有にサービス・クラスを組み込むために必要な最小 CPU 使用率を指定します。

構成タイプ
データベース・マネージャー

適用

- ローカルおよびリモート・クライアントを持つデータベース・サーバー
- ローカル・クライアントを持つデータベース・サーバー
- ローカルとリモート・クライアントを持つマルチメンバー・データベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
即時

デフォルト [範囲]
5 [0 から 100]

DB2 データベース・マネージャーをアップグレードする際の **wlm_disp_min_util** データベース・マネージャー構成パラメーターの値は 5 です。

単位 パーセント

このデータベース・マネージャー構成パラメーターの使用を例で示すために、A、B、C という 3 つのサービス・クラスがあるとします。それぞれに、CPU リソースの共有が 1000 あります。この例では、サービス・クラスの共有がハード CPU シェアかソフト CPU シェアかに関係なく、同じ結果が得られます。サービス・クラス A および B の CPU 使用率の値はそれぞれ、**wlm_disp_min_util** 構成パラメーターに設定される 8% 以上です。サービス・クラス C の CPU 使用率は、**wlm_disp_min_util** 構成パラメーターに設定される値の 8% より少ない 3% です。CPU シェアの計算において、サービス・クラス C は実行中の作業をまったく持っていないと見なされます。そのため、サービス・クラス A と B だけが均等に CPU リソースを共有し、それぞれ 50 % のシェアを受け取ります。サービス・クラス C が作業を開始し、CPU 使用率が **wlm_disp_min_util** 構成パラメーターに設定される 8% という値以上になるまで続けると、 8% に達した時点で、サービス・クラス A、B、および C は均等に CPU リソースを共有すると見なされるようになり、それぞれ 33.3% の共有を受け取ります。

マルチメンバー・データベース環境では、WLM によって管理される CPU リソースの共有にホストまたは LPAR が含まれるかどうかを判別するために、ホストまたは LPAR 上の全メンバーの CPU 使用率の総計が **wlm_disp_min_util** 構成パラメーターに対して比較されます。

カタログ・ビュー

SYSCAT.HISTOGRAMTEMPLATEBINS

各行は、ヒストグラム・テンプレート bin を表します。

表 266. SYSCAT.HISTOGRAMTEMPLATEBINS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATENAME	VARCHAR (128)	Y	ヒストグラム・テンプレートの名前。
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
BINID	INTEGER		ヒストグラム・テンプレート bin の ID。
BINUPPERVALUE	BIGINT		ヒストグラム・テンプレートの 1 つの bin の高い方の値。

SYSCAT.HISTOGRAMTEMPLATES

各行は、ヒストグラム・テンプレートを表します。

表 267. SYSCAT.HISTOGRAMTEMPLATES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
TEMPLATENAME	VARCHAR (128)		ヒストグラム・テンプレートの名前。
CREATE_TIME	TIMESTAMP		ヒストグラム・テンプレートが作成された時刻。
ALTER_TIME	TIMESTAMP		ヒストグラム・テンプレートが最後に変更された時刻。
NUMBINS	INTEGER		ヒストグラム・テンプレートの bin の数。これには上限値のない最後の bin も含まれます。
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.HISTOGRAMTEMPLATEUSE

各行は、ヒストグラム・テンプレートを使用できるワークロード管理オブジェクトとヒストグラム・テンプレートの関係を表します。

表 268. SYSCAT.HISTOGRAMTEMPLATEUSE カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATENAME	VARCHAR (128)	Y	ヒストグラム・テンプレートの名前。

表 268. SYSCAT.HISTOGRAMTEMPLATEUSE カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
HISTOGRAMTYPE	CHAR (1)		このテンプレートに基づくヒストグラムで収集される情報のタイプ。 • C = アクティビティー見積コストのヒストグラム • E = アクティビティー実行時間のヒストグラム • I = 1 つのアクティビティーが到着してから別のアクティビティーが到着するまでの時間のヒストグラム • L = アクティビティー存続時間のヒストグラム • Q = アクティビティー・キュー時間のヒストグラム • R = 要求実行時間のヒストグラム • U = 作業単位存続時間のヒストグラム
OBJECTTYPE	CHAR (1)		WLM オブジェクトのタイプ。 • b = サービス・クラス • k = 作業アクション • w = ワークロード
OBJECTID	INTEGER		WLM オブジェクトの ID。
SERVICECLASSNAME	VARCHAR (128)	Y	サービス・クラスの名前。
PARENTSERVICECLASSNAME	VARCHAR (128)	Y	ヒストグラム・テンプレートを使用するサービス・サブクラスの親サービス・クラスの名前。
WORKACTIONNAME	VARCHAR (128)	Y	ヒストグラム・テンプレートを使用する作業アクションの名前。
WORKACTIONSETNAME	VARCHAR (128)	Y	ヒストグラム・テンプレートを使用する作業アクションが含まれる作業アクション・セットの名前。
WORKLOADNAME	VARCHAR (128)	Y	ヒストグラム・テンプレートを使用するワークロードの名前。

SYSCAT.SERVICECLASSES

各行はサービス・クラスを表します。

表 269. SYSCAT.SERVICECLASSES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
SERVICECLASSNAME	VARCHAR (128)		サービス・クラスの名前。
PARENTSERVICECLASSNAME	VARCHAR (128)	Y	親サービス・スーパークラスのサービス・クラス名。

表 269. SYSCAT.SERVICECLASSES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
SERVICECLASSID	SMALLINT		サービス・クラスの ID。
PARENTID	SMALLINT		このサービス・クラスの親サービス・クラスの ID。このサービス・クラスがスーパー・サービス・クラスの場合は 0 です。
CREATE_TIME	TIMESTAMP		サービス・クラスが作成された時刻。
ALTER_TIME	TIMESTAMP		サービス・クラスが最後に変更された時刻。
ENABLED	CHAR (1)		サービス・クラスの状態。 • N = 使用不可 • Y = 使用可能
AGENTPRIORITY	SMALLINT		DB2 スレッドの通常優先順位に対して相対的なサービス・クラス内のエージェントのスレッド優先順位。 • -20 から 20 (Linux および UNIX) • -6 から 6 (Windows) • -32768 = 設定しない
PREFETCHPRIORITY	CHAR (1)		サービス・クラス内のエージェントのプリフェッチ優先順位。 • H = 高 • L = 低 • M = 中 • ブランク = 設定しない
MAXDEGREE	SMALLINT	Y	将来の利用のために予約済み。
BUFFERPOOLPRIORITY	CHAR (1)		サービス・クラス内のエージェントのバッファー・プール優先順位。 • H = 高 • L = 低 • M = 中 • ブランク = 設定しない
INBOUNDCORRELATOR	VARCHAR (128)	Y	将来の利用。
OUTBOUNDCORRELATOR	VARCHAR (128)	Y	サービス・クラスとオペレーティング・システムのワークロード・マネージャー・サービス・クラスを関連付けるために使用されるストリング。
COLLECTAGGACTDATA	CHAR (1)		該当するイベント・モニターにサービス・クラスでキャプチャーさせる集約アクティビティー・データを指定します。 • B = 基礎集約アクティビティー・データを収集する • E = 拡張集約アクティビティー・データを収集する • N = なし

表 269. SYSCAT.SERVICECLASSES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLECTAGGREQDATA	CHAR (1)		このサービス・クラスについて、該当するイベント・モニターでキャプチャーする集約要求データを指定します。 • B = 基礎集約要求データを収集する • N = なし
COLLECTACTDATA	CHAR (1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • D = 詳細ありのアクティビティー・データ • N = なし • S = 詳細およびセクション環境のあるアクティビティー・データ • V = 詳細および値ありのアクティビティー・データ • W = 詳細なしのアクティビティー・データ • X = 詳細、セクション環境、および値のあるアクティビティー・データ
COLLECTACTPARTITION	CHAR (1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーター・メンバー • D = すべてのメンバー
COLLECTREQMETRICS	CHAR (1)		サービス・スーパークラスに関連付けられている接続によって実行依頼された要求のモニター・レベルを指定します。 • B = 基礎要求メトリックを収集する • E = 拡張要求メトリックを収集する • N = なし
CPUSHARES	INTEGER		このサービス・クラスに割り振る CPU シェアの数。
CPUSHARETYPE	CHAR (1)		CPU シェアのタイプを指定します。 • S = ソフト・シェア • H = ハード・シェア
CPULIMIT	SMALLINT		サービス・クラスに割り振り可能な CPU リソースの最大パーセンテージ。CPU リミットがない場合は -1
SORTMEMORYPRIORITY	CHAR (1)		将来の利用のために予約済み。

表 269. SYSCAT.SERVICECLASSES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
SECTIONACTUALSOPTIONS	VARCHAR (32)		セクションの実行中に収集するセクション実行時統計を指定します。 ストリング内の最初の位置は、セクション実行時統計の収集が有効かどうかを表します。 • B = 有効。セクションで参照されるオブジェクトごとに基本演算子カーディナリティーのカウントおよび統計を収集します (DML ステートメントのみ)。 • N = 有効ではありません。 2 番目の位置は常に「N」で、将来の利用のために予約してあります。
COLLECTAGGUOWDATA	CHAR (1)		このサービス・クラスについて、該当するイベント・モニターでキャプチャーする集約作業単位データを指定します。 • B = 基礎集約作業単位データを収集する • N = なし
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.THRESHOLDS

各行はしきい値を表します。

表 270. SYSCAT.THRESHOLDS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
THRESHOLDNAME	VARCHAR (128)		しきい値の名前。
THRESHOLDID	INTEGER		しきい値の ID。
ORIGIN	CHAR (1)		しきい値の作成元。 • U = しきい値はユーザーによって作成された • W = しきい値は作業アクション・セットから作成された
THRESHOLDCLASS	CHAR (1)		しきい値の分類。 • A = 集約のしきい値 • C = アクティビティーのしきい値

表 270. SYSCAT.THRESHOLDS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
THRESHOLDPREDICATE	VARCHAR (15)		しきい値のタイプ。可能な値は以下のとおりです。 • AGGTEMPSPACE • CONCDBC • CONCWCN • CONCWOC • CONNIDLETIME • CPUTIME • CPUTIMEINSC • DATATAGINSC • DATATAGNOTINSC • DBCONN • ESTSQLCOST • ROWSREAD • ROWSREADINSC • ROWSRET • SCCONN • TEMPSPACE • TOTALTIME • UOWTOTALTIME
THRESHOLDPREDICATEID	SMALLINT		しきい値の述部の ID。
DOMAIN	CHAR (2)		しきい値のドメイン。 • DB = データベース • SB = サービス・サブクラス • SP = サービス・スーパークラス • WA = 作業アクション・セット • WD = ワークロード定義 • SQ = SQL ステートメント
DOMAINID	INTEGER		しきい値が関連付けられているオブジェクトの ID。これは、サービス・クラス、作業アクション、ワークロードの固有 ID、SQL ステートメントのいずれかになります。これがデータベースのしきい値である場合は、値は 0 です。
ENFORCEMENT	CHAR (1)		しきい値の強制の有効範囲。 • D = データベース • P = メンバー • W = ワークロード・オカレンス

表 270. SYSCAT.THRESHOLDS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
QUEUING	CHAR (1)		• N = しきい値はキューイングを行っていない • Y = しきい値はキューイングを行っている
MAXVALUE	BIGINT		しきい値によって指定された上限。THRESHOLDPREDICATE が 'DATATAGINSC' または 'DATATAGNOTINSC' である場合、この値は 1 つ以上のデータ・タグをエンコードします。
DATATAGLIST	VARCHAR (256)	Y	THRESHOLDPREDICATE が 'DATATAGINSC' または 'DATATAGNOTINSC' である場合、この値は 1 つ以上のデータ・タグをコンマ区切りリストで示します。指定されていない場合は NULL 値です。
QUEUESIZE	INTEGER		QUEUEING が 'Y' の場合は、キューのサイズ。それ以外の場合は -1。
OVERFLOWPERCENT	SMALLINT		将来の利用のために予約済み。
COLLECTACTDATA	CHAR (1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • D = 詳細ありのアクティビティー・データ • N = なし • S = 詳細およびセクション環境のあるアクティビティー・データ • V = 詳細および値ありのアクティビティー・データ • W = 詳細なしのアクティビティー・データ • X = 詳細、セクション環境、および値のあるアクティビティー・データ
COLLECTACTPARTITION	CHAR (1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーター・メンバー • D = すべてのメンバー
EXECUTION	CHAR (1)		しきい値を超えた後の実行アクションを示します。 • C = 実行を続行する • F = アプリケーションは強制的にシステムから切断される • R = 別のサービス・サブクラスに実行を再マップする • S = 実行を停止する

表 270. SYSCAT.THRESHOLDS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REMAPSCID	SMALLINT		REMAP ACTIVITY アクションのターゲット・サービス・サブクラス ID。
VIOLATIONRECORDLOGGED	CHAR (1)		しきい値違反の場合に、レコードをイベント・モニターに書き込むかどうかを示します。 • N = いいえ • Y = はい
CHECKINTERVAL	INTEGER		THRESHOLDPREDICATE が以下の場合に、しきい値条件を検査する間隔 (秒単位)。 • 'CPUTIME' • 'CPUTIMEINSC' • 'ROWSREAD' • 'ROWSREADINSC' その他の場合には -1。
ENABLED	CHAR (1)		• N = このしきい値は使用不可。 • Y = このしきい値は使用可能。
CREATE_TIME	TIMESTAMP		しきい値が作成された時刻。
ALTER_TIME	TIMESTAMP		しきい値が最後に変更された時刻。
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WORKACTIONS

各行は、作業アクション・セットで定義されている作業アクションを表します。

表 271. SYSCAT.WORKACTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ACTIONNAME	VARCHAR (128)		作業アクションの名前。
ACTIONID	INTEGER		作業アクションの ID。
ACTIONSETNAME	VARCHAR (128)	Y	作業アクション・セットの名前。
ACTIONSETID	INTEGER		この作業アクションが属する作業アクション・セットの ID。この列は、SYSCAT.WORKACTIONSETS ビューの ACTIONSETID 列を参照します。
WORKCLASSNAME	VARCHAR (128)	Y	作業クラスの名前。
WORKCLASSID	INTEGER		作業クラスの ID。この列は、SYSCAT.WORKCLASSES ビューの WORKCLASSID 列を参照します。
CREATE_TIME	TIMESTAMP		作業アクションが作成された時刻。
ALTER_TIME	TIMESTAMP		作業アクションが最後に変更された時刻。
ENABLED	CHAR (1)		• N = この作業アクションは使用不可です。 • Y = この作業アクションは使用可能です。

表 271. SYSCAT.WORKACTIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ACTIONTYPE	CHAR (1)		有効範囲内にある作業クラスの属性と一致する各 DB2 アクティビティーで実行されるアクション・タイプ。 • B = 基礎集約アクティビティー・データを収集します。サービス・クラスまたはワークロードに適用される作業アクション・セットにのみ指定可能です。 • C = 関連作業クラスの下にある任意の DB2 アクティビティーが、作業クラス・カウンターを実行して増分することを許可します。 • D = アクティビティーのコーディネーター・メンバーで詳細を含むアクティビティー・データを収集します。 • E = 拡張集約アクティビティー・データを収集します。サービス・クラスまたはワークロードに適用される作業アクション・セットにのみ指定可能です。 • F = アクティビティーのコーディネーター・メンバーで詳細、セクション、および値を含むアクティビティー・データを収集します。 • G = アクティビティーのコーディネーター・メンバーでアクティビティーの詳細およびセクションを収集し、すべてのメンバーでアクティビティー・データを収集します。 • H = アクティビティーのコーディネーター・メンバーでアクティビティーの詳細、セクション、および値を収集し、すべてのメンバーでアクティビティー・データを収集します。 • M = サービス・サブクラスにマップします。サービス・クラスに適用される作業アクション・セットにのみ指定可能です。 • P = この作業アクションが関連付けられている作業クラスにある DB2 アクティビティーが実行されないようにします。 • S = アクティビティーのコーディネーター・メンバーで詳細およびセクションを含むアクティビティー・データを収集します。 • T = このアクションはしきい値を表します。データベースまたはワークロードに関連付けられている作業アクション・セットにのみ指定可能です。 • U = ネスト・レベルがゼロのすべてのアクティビティーと、それらのアクティビティーの下にネストされているすべてのアクティビティーを、サービス・サブクラスにマップします。サービス・クラスに適用される作業アクション・セットにのみ指定可能です。 • V = コーディネーター・メンバーで詳細および値を含むアクティビティー・データを収集します。 • W = コーディネーター・メンバーで詳細を含まないアクティビティー・データを収集します。 • X = コーディネーター・メンバーで詳細を含むアクティビティー・データを収集し、すべてのメンバーでアクティビティー・データを収集します。 • Y = コーディネーター・メンバーで詳細および値を含むアクティビティー・データを収集し、すべてのメンバーでアクティビティー・データを収集します。 • Z = すべてのメンバーで、詳細を含まないアクティビティー・データを収集します。
REFOBJECTID	INTEGER	Y	ACTIONTYPE が 'M' (マップ) または 'N' (ネストされたマップ) の場合、この値は DB2 アクティビティーのマップ先のサービス・サブクラスの ID に設定されます。 ACTIONTYPE が 'T' (しきい値) の場合、この値は、使用されるしきい値の ID に設定されます。それ以外のすべてのアクションの場合、この値は NULL になります。

表 271. SYSCAT.WORKACTIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REFOBJECTTYPE	VARCHAR (30)		ACTIONTYPE が 'M' または 'N' の場合、この値は 'SERVICE CLASS' に設定されます。ACTIONTYPE が 'T' の場合、この値は 'THRESHOLD' になり、それ以外の場合は NULL 値になります。
SECTIONACTUALSOPTIONS	VARCHAR (32)		セクションの実行中に収集するセクション実行時統計を指定します。 ストリング内の最初の位置は、セクション実行時統計の収集が有効かどうかを表します。 • B = 有効。セクションで参照されるオブジェクトごとに基本演算子カーディナリティーのカウントおよび統計を収集します (DML ステートメントのみ)。 • N = 有効ではありません。 2 番目の位置は常に「N」で、将来の利用のために予約してあります。

SYSCAT.WORKACTIONSETS

各行は、作業アクション・セットを表します。

表 272. SYSCAT.WORKACTIONSETS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ACTIONSETNAME	VARCHAR (128)		作業アクション・セットの名前。
ACTIONSETID	INTEGER		作業アクション・セットの ID。
WORKCLASSSETNAME	VARCHAR (128)	Y	作業クラス・セットの名前。
WORKCLASSSETID	INTEGER		OBJECTID で指定されたオブジェクトにマップされる作業クラス・セットの ID。この列は SYSCAT.WORKCLASSSETS ビューの WORKCLASSSETID を参照します。
CREATE_TIME	TIMESTAMP		作業アクション・セットが作成された時刻。
ALTER_TIME	TIMESTAMP		作業アクション・セットが最後に変更された時刻。
ENABLED	CHAR (1)		• N = この作業アクション・セットは使用不可です。 • Y = この作業アクション・セットは使用可能です。
OBJECTTYPE	CHAR (1)		• b = サービス・スーパークラス • w = ワークロード • ブランク = データベース
OBJECTNAME	VARCHAR (128)	Y	サービス・クラスまたはワークロードの名前。
OBJECTID	INTEGER		作業クラス・セット (WORKCLASSSETID で指定される) のマップ先のオブジェクトの ID。OBJECTTYPE が 'b' の場合、OBJECTID はサービス・スーパークラスの ID です。OBJECTTYPE が 'w' の場合、OBJECTID はワークロードの ID です。 OBJECTTYPE がブランクの場合、OBJECTID は -1 です。

表 272. SYSCAT.WORKACTIONSETS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WORKCLASSES

各行は、作業クラス・セットで定義されている作業クラスを表します。

表 273. SYSCAT.WORKCLASSES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKCLASSNAME	VARCHAR (128)		作業クラスの名前。
WORKCLASSSETNAME	VARCHAR (128)	Y	作業クラス・セットの名前。
WORKCLASSID	INTEGER		作業クラスの ID。
WORKCLASSSETID	INTEGER		この作業クラスが属する作業クラス・セットの ID。この列は、SYSCAT.WORKCLASSSETS ビューの WORKCLASSSETID 列を参照します。
CREATE_TIME	TIMESTAMP		作業クラスが作成された時刻。
ALTER_TIME	TIMESTAMP		作業クラスが最後に変更された時刻。
EVALUATIONORDER	SMALLINT		作業クラス・セット内の作業クラスを選択するときに使用される評価順序を一意的に識別します。

SYSCAT.WORKCLASSSETS

各行は、作業クラス・セットを表します。

表 274. SYSCAT.WORKCLASSSETS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKCLASSSETNAME	VARCHAR (128)		作業クラス・セットの名前。
WORKCLASSSETID	INTEGER		作業クラス・セットの ID。
CREATE_TIME	TIMESTAMP		作業クラス・セットが作成された時刻。
ALTER_TIME	TIMESTAMP		作業クラス・セットが最後に変更された時刻。
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WORKLOADAUTH

各行は、ワークロードに対する USAGE 特権が付与されているユーザー、グループ、またはロールを表します。

表 275. SYSCAT.WORKLOADAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR (128)		ワークロードの名前。
GRANTOR	VARCHAR (128)		特権の認可者。
GRANTORTYPE	CHAR (1)		• U = GRANTEE は個々のユーザー。
GRANTEE	VARCHAR (128)		特権の保有者。
GRANTEETYPE	CHAR (1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
USAGEAUTH	CHAR (1)		GRANTEE がワークロードに対する USAGE 特権を保有するかどうかを示します。 • N = 保有しない • Y = 保有する

SYSCAT.WORKLOADCONNATTR

各行は、ワークロードの定義における接続属性を表します。

表 276. SYSCAT.WORKLOADCONNATTR カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR (128)		ワークロードの名前。
CONNATTRTYPE	VARCHAR (30)		接続属性のタイプ。 • 1 = APPLNAME • 2 = SYSTEM_USER • 3 = SESSION_USER • 4 = SESSION_USER GROUP • 5 = SESSION_USER ROLE • 6 = CURRENT CLIENT_USERID • 7 = CURRENT CLIENT_APPLNAME • 8 = CURRENT CLIENT_WRKSTNNAME • 9 = CURRENT CLIENT_ACCTNG • 10 = ADDRESS
CONNATTRVALUE	VARCHAR (1000)		接続属性の値。

SYSCAT.WORKLOADS

各行はワークロードを表します。

表 277. SYSCAT.WORKLOADS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR (128)		ワークロードの名前。
EVALUATIONORDER	SMALLINT		ワークロードの選択に使用される評価順序。
CREATE_TIME	TIMESTAMP		ワークロードが作成された時刻。
ALTER_TIME	TIMESTAMP		ワークロードが最後に変更された時刻。
ENABLED	CHAR (1)		• N = このワークロードは使用不可。 • Y = このワークロードは使用可能。
ALLOWACCESS	CHAR (1)		• N = このワークロードに関連付けられている UOW は拒否される。 • Y = このワークロードに関連付けられている作業単位 (UOW) はデータベースにアクセスできる。
MAXDEGREE	SMALLINT		ワークロードにおける最大の並列処理の度合い。有効な値は、1 から 32767 までと、-1 です。MAXIMUM DEGREE が DEFAULT の場合、値は -1 です。
SERVICECLASSNAME	VARCHAR (128)		(このワークロードに関連付けられている) 作業単位が割り当てられるサービス・サブクラスの名前。
PARENTSERVICECLASSNAME	VARCHAR (128)	Y	(このワークロードに関連付けられている) 作業単位が割り当てられるサービス・スーパークラスの名前。
COLLECTAGGACTDATA	CHAR (1)		このワークロードについて、該当するイベント・モニターでキャプチャーする集約アクティビティー・データを指定します。 • B = 基礎集約アクティビティー・データを収集する • E = 拡張集約アクティビティー・データを収集する • N = なし

表 277. SYSCAT.WORKLOADS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLECTACTDATA	CHAR (1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • D = 詳細ありのアクティビティー・データ • N = なし • S = 詳細およびセクション環境のあるアクティビティー・データ • V = 詳細および値ありのアクティビティー・データ (COLLECT 列が 'C' に設定されている場合に適用される) • W = 詳細なしのアクティビティー・データ • X = 詳細、セクション環境、および値のあるアクティビティー・データ
COLLECTACTPARTITION	CHAR (1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーター・メンバー • D = すべてのメンバー
COLLECTDEADLOCK	CHAR (1)		該当するイベント・モニターがデッドロック・イベントを収集するように指定します。 • H = 過去のアクティビティーのデッドロック・データのみを収集する • V = 過去のアクティビティーと値のデッドロック・データを収集する • W = 過去のアクティビティーと値以外のデッドロック・データを収集する
COLLECTLOCKTIMEOUT	CHAR (1)		該当するイベント・モニターがロック・タイムアウト・イベントを収集するように指定します。 • H = 過去のアクティビティーのロック・タイムアウト・データのみを収集する • N = ロック・タイムアウト・データを収集しない • V = 過去のアクティビティーと値のロック・タイムアウト・データを収集する • W = 過去のアクティビティーと値以外のロック・タイムアウト・データを収集する

表 277. SYSCAT.WORKLOADS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLECTLOCKWAIT	CHAR (1)		該当するイベント・モニターがロック待機イベントを収集するように指定します。 • H = 過去のアクティビティーのロック待機データのみを収集する • N = ロック待機データを収集しない • V = 過去のアクティビティーと値のロック待機データを収集する • W = 過去のアクティビティーと値以外のロック待機データを収集する
LOCKWAITVALUE	INTEGER		該当するイベント・モニターがロック・イベントを収集する前に、ロックが待機すべき時間 (ミリ秒単位) を指定します。 COLLECTLOCKWAIT = 'N' の場合は 0。
COLLECTACTMETRICS	CHAR (1)		ワークロードのオカレンスによってサブミットされるアクティビティーのモニター・レベルを指定します。 • B = 基礎アクティビティー・メトリックを収集する • E = 拡張アクティビティー・メトリックを収集する • N = なし
COLLECTUOWDATAOPTIONS	VARCHAR (32)		該当するイベント・モニターによって収集する作業単位データを指定します。ストリング内の最初の位置は、作業単位データの収集が有効かどうかを表します。 • B = 有効。基礎作業単位データを収集する • N = 使用可能でない 2 番目の位置以降では、ストリング内の各位置は以下のように特定の拡張オプションを表します。 • 2 = パッケージ参照リスト • 3 = 実行可能 ID リスト 拡張オプションを表す各位置は、以下のいずれかの値に設定されます。 • Y = 拡張オプションが含まれる • N = 拡張オプションが含まれない

表 277. SYSCAT.WORKLOADS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLECTUOWDATA	CHAR (1)		該当するイベント・モニターによって収集する作業単位データを指定します。 • B = 基礎作業単位データを収集する • N = なし • P = 基礎作業単位データとパッケージ・リストを収集する この列は推奨されません。この列の情報は、COLLECTUOWDATAOPTIONS で得られます。
EXTERNALNAME	VARCHAR (128)	Y	将来の利用のために予約済み。
SECTIONACTUALSOPTIONS	VARCHAR (32)		セクションの実行中に収集するセクション実行時統計を指定します。 ストリング内の最初の位置は、セクション実行時統計の収集が有効かどうかを表します。 • B = 有効。セクションで参照されるオブジェクトごとに基本演算子カーディナリティーのカウントおよび統計を収集します (DML ステートメントのみ)。 • N = 有効ではありません。 2 番目の位置は常に「N」で、将来の利用のために予約してあります。
COLLECTAGGUOWDATA	CHAR (1)		このワークロードについて、該当するイベント・モニターでキャプチャーする集約作業単位データを指定します。 • B = 基礎集約作業単位データを収集する • N = なし
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

付録 A. 一般的な命名規則

すべてのデータベース・オブジェクト、ユーザー名、パスワード、グループ、ファイル、パスには、命名規則があります。これらの規則には、作業しているプラットフォームに特有のものもあります。

例えば、ファイル・システムで表示されるオブジェクト (データベースやインスタンスなど) の名前における大文字と小文字の使用に関して、次のような規則があります。

- UNIX プラットフォームでは、名前は大文字と小文字の区別があります。例えば、`/data1` は、`/DATA1` または `/Data1` と同じディレクトリーではありません。
- Windows プラットフォームでは、名前には大文字と小文字の区別がありません。例えば、`¥data1` は `¥DATA1` および `¥Data1` と同じです。

特に指定がない限り、名前には以下の文字を含めることができます。

- 基本 (7 ビット) ASCII 文字セットで定義される文字 A から Z、および a から z。 SQL ステートメントを使用して作成されたオブジェクトの ID で使用する場合、小文字の「a」から「z」までは引用符 (") で区切らない限りは、大文字に変換されます。
- 0 から 9 の数字。
- ! % ( ) { } . - ^ ~ _ (アンダースコアー) @、#、$、およびスペース。
- ¥ (円記号)。

制約事項

- 数値または下線文字で名前を開始しないでください。
- 表、ビュー、列、索引、または許可 ID の名前には、SQL 予約語を使用しないでください。
- ディレクトリーおよびファイルの名前には、基本 ASCII 文字セットで定義されている文字だけを使用してください。ご使用のコンピューターのオペレーティング・システムでさまざまなコード・ページがサポートされているかもしれませんが、非 ASCII 文字は確実に処理されない可能性があります。非 ASCII 文字を分散環境で使用すると特定の問題が生じることがあります。分散環境においては、コンピューターごとに使用するコード・ページが異なっている場合があります。
- ご使用のオペレーティング・システム、および DB2 データベースを操作している場所によって、異なる働きをする特殊文字が他にもあります。それらの文字は正常に機能する可能性がありますが、必ず機能するという保証はありません。データベース内のオブジェクトを命名する際には、これら他の特殊文字を使用することはお奨めしません。
- ユーザー名およびグループ名も、そのオペレーティング・システムによって定められている規則に従う必要があります。例えば、Linux および UNIX プラットフォームでは、ユーザー名およびグループ名の文字は、a から z までの小文字、0 から 9 までの数字、および _ (下線) である必要があります (名前の最初の文字は 0 から 9 までの数字以外にします)。

- 各長さは、「*SQL リファレンス*」の『SQL および XML の制限値』にリストされた長さ以下である必要があります。
- **AUTHID ID に関する制約事項:** DB2 バージョン 9.5 以降では、128 バイトの許可 ID を付けることが可能です。ただし、許可 ID がオペレーティング・システムのユーザー ID またはグループ名として解釈される場合、オペレーティング・システムの命名上の制約事項が適用されます。例えば、Linux および UNIX オペレーティング・システムではユーザー ID とグループ名に関して最大 8 文字という制限があり、Windows オペレーティング・システムの場合は最大 30 文字という制限があります。このため、128 バイトの許可 ID を付与できますが、この許可 ID を持つユーザーとして接続することはできません。独自のセキュリティー・プラグインを作成する場合は、拡張されたサイズの許可 ID を使用できます。例えば、セキュリティー・プラグインに 30 バイトのユーザー ID を与えて、接続可能な認証中に、セキュリティー・プラグインが 128 バイトの許可 ID を返すようにすることができます。

他にも、オブジェクト命名規則、多文化サポート環境での命名規則、および Unicode 環境での命名規則も考慮する必要があります。

付録 B. ロール

ロールは、特権の管理を簡素化します。つまり、グループと同等の機能は提供されますが、同じ制約事項は設けられません。

ロールは、1 つ以上の特権をまとめたデータベース・オブジェクトですが、これを、GRANT ステートメントを使用してユーザー、グループ、PUBLIC、またはその他のロールに割り当てるか、あるいは、CREATE TRUSTED CONTEXT または ALTER TRUSTED CONTEXT ステートメントを使用して、トラステッド・コンテキストに割り当てることができます。ワークロード定義内で、SESSION_USER ROLE 接続属性用のロールを指定することができます。

ロールは、次のように、データベース・システム内での特権の管理がより簡単になるという利点を備えています。

- セキュリティー管理者は、組織の構造を映し出すような方法で、そのデータベースへのアクセスを制御することができます (組織における役職や担当業務に対応したロールをデータベース内に作成できます)。
- ユーザーには、その役職や担当業務に応じたロールに対するメンバーシップが付与されます。ユーザーの役職や担当業務の変更に応じて、ロールに対するメンバーシップを簡単に付与または取り消すことができます。
- 特権の割り当てが簡素化されます。管理者は、特定の役職や担当業務に該当する個々のユーザーに一連の同じ特権を付与するのではなく、その役職や担当業務に応じたロールに対してこの一連の特権を付与してから、その役職や担当業務に該当する各ユーザーにそのロールを付与することができます。
- そのロールを付与されたすべてのユーザーに対し、更新が適用されます。つまり管理者は、個人ごとに各ユーザーの特権を更新する必要はありません。
- ビュー、トリガー、マテリアライズ照会表 (MQT)、静的 SQL、および SQL ルーチンの作成時には、ロールに対して付与された特権および権限が常に使用されます。この場合、グループに付与された特権および権限は (直接でも間接にでも) 使用されません。

  その理由は、グループはサード・パーティー・ソフトウェア (例えば、オペレーティング・システムまたは LDAP ディレクトリー) によって管理されるので、DB2 データベース・システムは、グループ内のメンバーシップがいつ変更になったかを判別できないからです。ロールはデータベース内部で管理されるので、DB2 データベース・システムは、許可がいつ変更されたかを判別して、それに応じたアクションをとることができます。グループが検討の対象にならないのと同じ理由で、グループに付与されたロールも検討の対象にはなりません。
- ユーザーに割り当てられたすべてのロールは、ユーザーが接続を確立したときに有効になるので、ロールに付与されたすべての特権と許可も、ユーザーが接続するときに有効となります。ロールを明示的に有効または無効にすることはできません。
- セキュリティー管理者は、ロールの管理を他人に委任することができます。

データベース内で付与できるどの DB2 特権および権限でも、ロールに付与することができます。例えば、以下のどの権限および特権でも、ロールに付与することができます。

- DBADM、SECADM、DATAACCESS、ACCESSCTRL、SQLADM、WLMADM、LOAD、および IMPLICIT_SCHEMA データベース権限
- CONNECT、CREATETAB、CREATE_NOT_FENCED、BINDADD、CREATE_EXTERNAL_ROUTINE、または QUIESCE_CONNECT データベース権限
- 任意のデータベース・オブジェクト特権 (CONTROL を含む)

ユーザーがデータベースに接続したとき、そのユーザーのロールは自動的に有効になり、許可の検討の対象になります。つまり、SET ROLE ステートメントを使用してロールを活動化する必要はありません。例えば、ビュー、マテリアライズ照会表 (MQT)、トリガー、パッケージ、または SQL ルーチンを作成すると、ロールを通して取得した特権が適用されます。ただし、自分がメンバーとして所属するグループに付与されたロールを通して取得した特権は適用されません。

ロールには所有者はいません。セキュリティー管理者は、GRANT ステートメントの WITH ADMIN OPTION 節を使用して、ロールの管理を別のユーザーに委任することができます。それによって、他のユーザーがロールのメンバーシップを制御できるようになります。

制約事項

ロールの使用に関しては、次のようないくつかの制約事項があります。

- ロールはデータベース・オブジェクトを所有できません。
- 以下のデータベース・オブジェクトの作成時には、グループに付与された許可およびロールは検討の対象にはなりません。
  - 静的 SQL を格納するパッケージ。
  - ビュー
  - マテリアライズ照会表 (MQT)
  - トリガー
  - SQL ルーチン

  オブジェクトを作成するユーザーに対してか、または PUBLIC に対して直接または間接的に (例えばロール階層を介して) 付与されたロールだけが、上記のオブジェクトの作成時に検討の対象になります。

付録 C. トラステッド・コンテキストおよびトラステッド接続

トラステッド・コンテキストとは、データベースと外部エンティティー (アプリケーション・サーバーなど) の間の接続における信頼関係を定義するデータベース・オブジェクトのことをいいます。

信頼関係は、以下の属性のセットに基づいています。

- システム許可 ID: データベース接続を確立するユーザーを表します
- IP アドレス (またはドメイン・ネーム): データベース接続を確立するホストを表します
- データ・ストリーム暗号化: データベース・サーバーとデータベース・クライアントの間のデータ通信のための暗号化設定がある場合にはそれを表します

ユーザーがデータベース接続を確立するときに、DB2 データベース・システムは、接続がデータベース内のトラステッド・コンテキスト・オブジェクトの定義と一致するかどうかを検査します。一致していた場合、データベース接続は信頼できると見なされます。

トラステッド接続を使用すると、このトラステッド接続の起動側では、トラステッド接続の有効範囲外では使用できない追加機能を取得することができます。追加機能は、トラステッド接続が明示的であるか暗黙的であるかによって異なります。

明示的トラステッド接続の起動側には以下の機能があります。

- その接続の現行ユーザー ID を、認証のあるなしに関係なく別のユーザー ID に切り替える
- トラステッド・コンテキストのロール継承フィーチャーにより追加の特権を取得する

暗黙的トラステッド接続は、明示的に要求されていないトラステッド接続で、明示的トラステッド接続要求ではなく通常の接続要求により確立されます。暗黙接続を取得するためにアプリケーション・コードを変更する必要はありません。また、暗黙的トラステッド接続を取得するかどうかは、接続戻りコードには影響はありません (明示的トラステッド接続を要求する場合は、接続戻りコードは要求が成功したかどうかを示します)。暗黙的トラステッド接続の起動側は、トラステッド・コンテキストのロール継承フィーチャーにより追加の特権を取得できるだけで、ユーザー ID を切り替えることはできません。

トラステッド・コンテキストを使用するとどのようにセキュリティーが向上するか

3 層アプリケーション・モデルは、クライアント・アプリケーションとデータベース・サーバーの間に中間層を置くことにより、標準的な 2 層クライアントおよびサーバー・モデルを拡張します。このモデルは、Web ベースのテクノロジーや Java™ 2 Enterprise Edition (J2EE) プラットフォームの登場により、近年特に大きな人気を得ています。3 層アプリケーション・モデルをサポートするソフトウェア・プロダクトの一例として、IBM WebSphere Application Server (WAS) があります。

3 層アプリケーション・モデルでは、クライアント・アプリケーションを実行するユーザーの認証、およびデータベース・サーバーとの相互作用の管理は、中間層が処理します。従来の方法では、データベース・サーバーとのすべての相互作用は、データベース・サーバーに対して中間層を識別するユーザー ID と資格情報の組み合わせを使用して、その中間層により確立されたデータベース接続を介して行われます。つまり、データベース・サーバーは、中間層のユーザー ID に関連付けられたデータベース特権を使用して、すべてのデータベース・アクセスで行う必要がある許可検査および監査を行います。これには、ユーザーの代わりに中間層により実行されるアクセスも含まれます。

3 層アプリケーション・モデルには多くの利点がありますが、データベース・サーバーとのすべての相互作用 (例えば、ユーザー要求) を中間層の許可 ID で行うようにすると、いくつかのセキュリティー上の問題が生じます。これらを要約すると以下のようになります。

- ユーザー ID の消失

  企業によっては、アクセス制御の目的で、データベースにアクセスしている実際のユーザーの ID を知りたい場合があります。

- ユーザーの説明責任の減少

  監査による説明責任は、データベース・セキュリティーにおける基本原則です。ユーザーの ID が不明であると、中間層の固有の目的のために中間層により実行されるトランザクションと、ユーザーのために中間層により実行されるトランザクションを区別することが難しくなります。

- 中間層の許可 ID に特権を付与しすぎる

  中間層の許可 ID には、すべてのユーザーからのすべての要求を実行するために必要なすべての特権が含まれていなければなりません。これには、特定の情報にアクセスする必要がないユーザーがアクセス権限を取得できてしまうというセキュリティー問題があります。

- 弱いセキュリティー

  前述の特権の問題に加えて、現行方式では、接続するために中間層により使用される許可 ID には、ユーザー要求によりアクセスされる可能性があるすべてのリソースへの特権を付与する必要があります。中間層の許可 ID の暗号漏えいが発生すると、それらすべてのリソースは公開されてしまいます。

- 同じ接続を使用するユーザー間で相互に影響を及ぼす

  直前のユーザーによる変更が現行ユーザーに影響を与える場合があります。

明らかに、実際のユーザーの ID およびデータベース特権が、そのユーザーに代わって中間層により実行されるデータベース要求で使用されるようなメカニズムが必要です。この目標を達成するための最も簡単な方法は、中間層がユーザーの ID とパスワードを使用して新規接続を確立した後、ユーザーの要求をその接続を介して送信するというものです。この方法は単純ですが、以下に挙げるようないくつかの欠点があります。

- 特定の中間層では不適当。多くの中間層サーバーには、接続を確立するために必要なユーザー認証資格情報がありません。

- パフォーマンス上のオーバーヘッド。新しい物理接続を作成し、データベース・サーバーでユーザーを再認証することに関連した、パフォーマンス上の明らかなオーバーヘッドがあります。
- 保守上のオーバーヘッド。一元的なセキュリティー・セットアップ、またはシングル・サインオンを使用していない状況では、2 つのユーザー定義 (1 つは中間層上、もう 1 つはサーバー上) を持つことによる保守上のオーバーヘッドがあります。この状況では、異なる場所にあるパスワードを変更することが必要です。

トラステッド・コンテキスト機能は、この問題を解決します。セキュリティー管理者は、データベースと中間層の間の信頼関係を定義するトラステッド・コンテキスト・オブジェクトをデータベースに作成できます。その後、中間層ではデータベースへの明示的トラステッド接続を確立できますが、この接続では、接続の現行ユーザー ID を、認証のあるなしに関係なく別のユーザー ID に切り替える機能が中間層に付与されます。トラステッド・コンテキストは、エンド・ユーザーの ID アサーション問題を解決するだけでなく、別の利点もあります。それは、データベース・ユーザーが特権を使用できるようになる時期を制御する機能です。ユーザーが特権を使用できる時期を制御できないと、全体的なセキュリティーの低下につながります。例えば、特権が、最初に意図した目的以外で使用される場合があります。セキュリティー管理者は、1 つ以上の特権を 1 つのロールに割り当て、そのロールをトラステッド・コンテキスト・オブジェクトに割り当てることができます。そのトラステッド・コンテキストの定義と一致するトラステッド・データベース接続 (明示的または暗黙的) のみがそのロールに関連付けられた特権を利用できます。

パフォーマンスの向上

トラステッド接続を使用すると以下の利点があるため、パフォーマンスを最大限に発揮します。

- 接続の現行ユーザー ID が切り替わる時に新規接続は確立されません。
- トラステッド・コンテキスト定義が切り替え先のユーザー ID の認証を必要としない場合には、データベース・サーバーで新規ユーザーを認証することに関連したオーバーヘッドは発生しません。

トラステッド・コンテキストの作成例

セキュリティー管理者が以下のトラステッド・コンテキスト・オブジェクトを作成すると想定します。

```
CREATE TRUSTED CONTEXT CTX1
  BASED UPON CONNECTION USING SYSTEM AUTHID USER2
  ATTRIBUTES (ADDRESS   '192.0.2.1')
  DEFAULT ROLE managerRole
   ENABLE
```

ユーザー *user1* が IP アドレス 192.0.2.1 からトラステッド接続を要求した場合、DB2 データベース・システムは、トラステッド接続を確立できなかったためユーザー *user1* が非トラステッド接続を取得したことを示す警告 (SQLSTATE 01679、SQLCODE +20360) を戻します。しかし、ユーザー *user2* が IP アドレス 192.0.2.1 からトラステッド接続を要求した場合には、接続属性はトラステッド・コンテキスト CTX1 により条件が満たされるため、要求は受け入れられます。ユーザー *user2* はトラステッド接続を確立したため、そのユーザーはトラステッド・コンテキストのロール managerRole に関連付けられたすべての特権および権限を取得で

きます。このトラステッド接続の有効範囲外では、ユーザー *user2* はこれらの特権および権限を使用できません。

付録 D. DB2 技術情報の概説

DB2 技術情報は、さまざまな方法でアクセスすることが可能な、各種形式で入手できます。

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、IBM Publications Center (www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) から利用できる DB2 ライブラリーについて説明しています。英語および翻訳された DB2 バージョン 10.1 のマニュアル (PDF 形式) は、www.ibm.com/support/docview.wss?rs=71&uid=swg27009474 からダウンロードできます。

この表には印刷資料が入手可能かどうかが示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: *DB2 インフォメーション・センター*は、PDF やハードコピー資料よりも頻繁に更新されます。

表 278. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SA88-4671-00	入手可能	2012 年 4 月
管理ルーチンおよびビュー	SA88-4672-01	入手不可	2013 年 1 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻	SA88-4676-01	入手可能	2013 年 1 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻	SA88-4677-01	入手可能	2013 年 1 月
コマンド・リファレンス	SA88-4673-01	入手可能	2013 年 1 月
データベース: 管理の概念および構成リファレンス	SA88-4662-01	入手可能	2013 年 1 月
データ移動ユーティリティー: ガイドおよびリファレンス	SA88-4693-01	入手可能	2013 年 1 月
データベースのモニタリング ガイドおよびリファレンス	SA88-4663-01	入手可能	2013 年 1 月
データ・リカバリーと高可用性 ガイドおよびリファレンス	SA88-4694-01	入手可能	2013 年 1 月
データベース・セキュリティー・ガイド	SA88-4695-01	入手可能	2013 年 1 月

表 278. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 ワークロード管理ガイドおよびリファレンス	SA88-4685-01	入手可能	2013 年 1 月
ADO.NET および OLE DB アプリケーションの開発	SA88-4665-01	入手可能	2013 年 1 月
組み込み SQL アプリケーションの開発	SA88-4666-01	入手可能	2013 年 1 月
Java アプリケーションの開発	SA88-4669-01	入手可能	2013 年 1 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SA88-4670-00	入手不可	2012 年 4 月
IBM データ・サーバー用の RDF アプリケーション開発	SA88-5083-00	入手可能	2013 年 1 月
SQL および外部ルーチンの開発	SA88-4667-01	入手可能	2013 年 1 月
データベース・アプリケーション開発の基礎	GI88-4279-01	入手可能	2013 年 1 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4280-00	入手可能	2012 年 4 月
グローバリゼーション・ガイド	SA88-4696-00	入手可能	2012 年 4 月
DB2 サーバー機能 インストール	GA88-4679-01	入手可能	2013 年 1 月
IBM データ・サーバー・クライアント機能 インストール	GA88-4680-00	入手不可	2012 年 4 月
メッセージ・リファレンス 第 1 巻	SA88-4688-01	入手不可	2013 年 1 月
メッセージ・リファレンス 第 2 巻	SA88-4689-01	入手不可	2013 年 1 月
Net Search Extender 管理およびユーザーズ・ガイド	SA88-4691-01	入手不可	2013 年 1 月
パーティションおよびクラスタリングのガイド	SA88-4697-01	入手可能	2013 年 1 月
Preparation Guide for DB2 10.1 Fundamentals Exam 610	SC27-4540-00	入手不可	2013 年 1 月

表 278. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611	SC27-4541-00	入手不可	2013 年 1 月
pureXML ガイド	SA88-4686-01	入手可能	2013 年 1 月
Spatial Extender ユーザーズ・ガイドおよびリファレンス	SA88-4690-00	入手不可	2012 年 4 月
SQL プロシージャー言語: アプリケーションのイネーブルメントおよびサポート	SA88-4668-01	入手可能	2013 年 1 月
SQL リファレンス 第 1 巻	SA88-4674-01	入手可能	2013 年 1 月
SQL リファレンス 第 2 巻	SA88-4675-01	入手可能	2013 年 1 月
Text Search ガイド	SA88-4692-01	入手可能	2013 年 1 月
問題判別およびデータベース・パフォーマンスのチューニング	SA88-4664-01	入手可能	2013 年 1 月
DB2 バージョン 10.1 へのアップグレード	SA88-4678-01	入手可能	2013 年 1 月
DB2 バージョン 10.1 の新機能	SA88-4684-01	入手可能	2013 年 1 月
XQuery リファレンス	SA88-4687-01	入手不可	2013 年 1 月

表 279. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Connect Personal Edition インストールおよび構成	SA88-4681-00	入手可能	2012 年 4 月
DB2 Connect サーバー機能 インストールおよび構成	SA88-4682-01	入手可能	2013 年 1 月
DB2 Connect ユーザーズ・ガイド	SA88-4683-01	入手可能	2013 年 1 月

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果として生じる可能性がある状態に対応した SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順

SQL 状態ヘルプを開始するには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。
例えば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

他のバージョンの DB2 製品の資料は、ibm.com® のそれぞれのインフォメーション・センターにあります。

このタスクについて

DB2 バージョン 10.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v10r1 です。

DB2 バージョン 9.8 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/ です。

DB2 バージョン 9.7 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/ です。

DB2 バージョン 9.5 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5 です。

DB2 バージョン 9.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9/ です。

DB2 バージョン 8 のトピックについては、*DB2 インフォメーション・センター* の URL (http://publib.boulder.ibm.com/infocenter/db2luw/v8/) を参照してください。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

ローカルにインストールした DB2 インフォメーション・センターは、定期的に更新する必要があります。

始める前に

DB2 バージョン 10.1 インフォメーション・センターが既にインストール済みである必要があります。詳しくは、「*DB2 サーバー機能 インストール*」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

このタスクについて

既存の DB2 インフォメーション・センターは、自動で更新することも手動で更新することもできます。

- 自動更新は、既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、手動更新と比べて、更新中にインフォメーション・センターが使用できなくなる時間が短くなるというメリットがあります。さらに、自動更新は、定期的に実行する他のバッチ・ジョブの一部として実行されるように設定することができます。
- 手動更新は、既存のインフォメーション・センターのフィーチャーと言語の更新に使用できます。自動更新は更新処理中のダウン時間を減らすことができますが、フィーチャーまたは言語を追加する場合は手動処理を使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。自動更新処理では、インフォメーション・センターは、更新を行った後に、インフォメーション・センターを再始動するための停止が発生するだけで済みます。

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

手順

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動更新する手順を以下に示します。

1. Linux オペレーティング・システムの場合、次のようにします。
   a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`/opt/ibm/db2ic/V10.1` ディレクトリーにインストールされています。
   b. インストール・ディレクトリーから `doc/bin` ディレクトリーにナビゲートします。
   c. 次のように `update-ic` スクリプトを実行します。

      ```
      update-ic
      ```
2. Windows オペレーティング・システムの場合、次のようにします。
   a. コマンド・ウィンドウを開きます。
   b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`<Program Files>¥IBM¥DB2 Information Center¥バージョン 10.1` ディレクトリーにインストールされています (`<Program Files>` は「Program Files」ディレクトリーのロケーション)。

c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。

d. 次のように update-ic.bat ファイルを実行します。

```
update-ic.bat
```

タスクの結果

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、doc¥eclipse¥configuration ディレクトリーにあります。ログ・ファイル名はランダムに生成された名前です。例えば、1239053440785.log のようになります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

このタスクについて

ローカルにインストールされた *DB2 インフォメーション・センター* を手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の *DB2 インフォメーション・センター* を停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。DB2 インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

   注: ご使用の環境において、インターネットに接続されていないマシンに *DB2 インフォメーション・センター* の更新をインストールする必要がある場合、インターネットに接続されていて *DB2 インフォメーション・センター* がインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシーを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。
   更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。
3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の *DB2 インフォメーション・センター* を再開します。

注: Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、「**管理者として実行**」を選択します。

手順

コンピューターまたはイントラネット・サーバーにインストール済みの *DB2* インフォメーション・センター を更新するには、以下のようにします。

1. *DB2* インフォメーション・センター を停止します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**停止**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv10 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、*Program_Files*¥IBM¥DB2 Information Center¥バージョン 10.1 ディレクトリーにインストールされています (*Program_Files* は Program Files ディレクトリーのロケーション)。
     c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
     d. 次のように help_start.bat ファイルを実行します。

        ```
        help_start.bat
        ```
   - Linux の場合:
     a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、/opt/ibm/db2ic/V10.1 ディレクトリーにインストールされています。
     b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
     c. 次のように help_start スクリプトを実行します。

        ```
        help_start
        ```

   システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「**更新**」ボタン () をクリックします。(ブラウザーで JavaScript が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「**更新の検索**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。

6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
   - Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

     ```
     help_end.bat
     ```

     注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。
   - Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

     ```
     help_end
     ```

     注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを停止しないでください。
7. *DB2 インフォメーション・センター* を再開します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv10 start
     ```

タスクの結果

更新された *DB2 インフォメーション・センター* に、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 データベース製品のさまざまな機能について学習するための支援となります。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML ガイド* 」の『**pureXML®**』
: XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 の資料

トラブルシューティング情報は、「*問題判別およびデータベース・パフォーマンスのチューニング*」または *DB2 インフォメーション・センター* の『データベースの基本』セクションにあります。ここには、以下の情報が記載されています。

- DB2 診断ツールおよびユーティリティーを使用した、問題の切り分け方法および識別方法に関する情報。
- 最も一般的な問題のうち、いくつかの解決方法。
- DB2 データベース製品で発生する可能性のある、その他の問題の解決に役立つアドバイス。

IBM サポート・ポータル

現在問題が発生していて、考えられる原因とソリューションを見つけるには、IBM サポート・ポータルを参照してください。 Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

IBM サポート・ポータル (http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows) にアクセスしてください。

ご利用条件

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

IBM の商標: IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

付録 E. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

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

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、

利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは、現存するままの状態で提供されるものであり、いかなる種類の保証も提供されません。 IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*お客様の会社名*) (*西暦年*). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _*年を入れる*_. All rights reserved.

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- UNIX は The Open Group の米国およびその他の国における登録商標です。
- インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Celeron、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、 Windows、 Windows NT、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ラ行]

[ワ行]

A

C

D

E

I

L

R

S

T

U

W

Printed in Japan

SA88-4685-01

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

DB2 ワークロード管理 ガイドおよびリファレンス