IBM® DB2 Universal Database™

管理ガイド: パフォーマンス

バージョン 8.2

SC88-9134-01
(英文原典：SC09-4821-01)

IBM® DB2 Universal Database™

管理ガイド: パフォーマンス

バージョン 8.2

SC88-9134-01
(英文原典：SC09-4821-01)

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、『*特記事項*』に記載されている情報をお読みください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

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： SC09-4821-01
IBM® DB2 Universal Database™
Administration Guide: Performance
Version 8.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.8

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財）日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

本書について ix
本書の対象読者 x
本書の構成 x
他の管理ガイドの巻の概要 xi
管理ガイド：プランニング xi
管理ガイド：インプリメンテーション xii

第 1 部 パフォーマンスの紹介 1

第 1 章 パフォーマンスの紹介 3
パフォーマンスのエレメント 3
パフォーマンス・チューニングのガイドライン . . . 4
パフォーマンス・チューニング処理. 5
パフォーマンスの向上プロセスの開発 5
ユーザーが提供できるパフォーマンス情報 . . . 6
パフォーマンス・チューニングの限界 7
パフォーマンス・チューニングのためのクイック・スタート・ヒント 7

第 2 章 アーキテクチャーとプロセス. . . 9
DB2 アーキテクチャーおよびプロセスの概要 . . . 9
アプリケーション間のデッドロック 11
ディスク装置の概要 12
ディスク装置のパフォーマンス要因 12
データベース・ディレクトリーとファイル . . . 13
表スペースの概説 16
SMS 表スペース. 16
DMS 表スペース 17
DMS 表スペース・アドレス・マップの図 . . . 18
表と索引 19
標準の表における表および索引の管理 20
MDC 表のための表および索引管理 23
索引の構造 26
プロセス 27
ログ処理 27
挿入処理 28
更新処理 30
クライアント/サーバー処理モデル 31
メモリー管理 37

第 2 部 アプリケーション・パフォーマンスのチューニング. 41

第 3 章 アプリケーションについての考慮事項 43
並行性制御と分離レベル 43
並行性の問題 43
分離レベルのパフォーマンスへの影響 44
分離レベルの指定 48
並行性制御とロッキング 51
ロックおよび並行性の制御 51
ロック属性 53
ロックとパフォーマンス 55
ロッキングのガイドライン 60
ロック・エスカレーション問題の修正 62
ロック据え置きによる非コミット・データの評価 64
ロック・タイプの互換性 66
標準の表用のロック・モードおよびアクセス・パス 68
MDC 表の表および RID 索引スキャンのロック・モード 70
MDC 表のブロック索引スキャンのロック . . . 74
ロッキングに影響を与える要因. 76
ロッキングに影響を与える要因. 77
アプリケーション・プロセスのロックとタイプ . 77
ロックとデータ・アクセス方式. 78
索引タイプとネクスト・キー・ロッキング . . . 79
最適化の要因. 81
最適化クラスのガイドライン 81
最適化クラス. 83
最適化クラスの設定 85
アプリケーションの調整 87
SELECT ステートメントの制限のガイドライン 87
オーバーヘッド削減のための行ブロッキングの指定 90
照会チューニングのガイドライン 92
SQL 照会でのデータ・サンプリング 92
効率的な SELECT ステートメント 94
コンパウンド SQL のガイドライン 96
文字変換のガイドライン 97
ストアード・プロシージャーのガイドライン . . 98
アプリケーションの並列処理 99
| REOPT を指定してバインディングすることによ
| りパフォーマンスを向上させる 101

第 4 章 環境についての考慮事項 . . . 103
照会の最適化に影響を与えるデータベース・パーティション・グループ 103
表スペースが照会の最適化に与える影響 103
フェデレーテッド・データベースに影響を与えるサーバー・オプション 106

第 5 章 システム・カタログ統計 . . . 109
カタログ統計 109
カタログ統計の収集と分析 111
統計の収集および更新のガイドライン 111
カタログ統計の収集 112
特定の列に関する分散統計の収集. 113
詳細索引統計の収集 115
| 表データのサンプルでの統計の収集 116
| 統計プロファイルを使用した統計の収集 . . . 117

自動統計収集 119
自動統計収集の使用 120
収集される統計. 121
カタログ統計の表 121
収集される統計情報 126
分散統計 127
分散統計のオプティマイザーの使用 130
分散統計の使用の拡張例. 131
詳細索引統計 135
サブエレメント統計 136
ユーザーが更新可能なカタログ統計 138
ユーザー定義関数の統計. 138
モデル化および what-if の計画のカタログ統計 139
実動データベースのモデル化の統計 141
手動でのカタログ統計更新の一般規則 143
手動での列統計の更新の規則 144
手動での分散統計の更新の規則 145
手動での表およびニックネーム統計の更新の規則 146
手動での索引統計の更新の規則 147

第 6 章 SQL コンパイラーに関する解説 149
SQL コンパイラーの処理 149
照会の最適化に影響を与える構成パラメーター . . 153
照会の再書き込み 156
照会書き直しのメソッドとその例. 156
コンパイラー書き直しの例: ビューのマージ . . 158
コンパイラー書き直しの例: DISTINCT の除去 160
コンパイラー書き直しの例: 暗黙の述部 . . . 162
複数述部の列相関 163
REOPT BIND オプションを使用した照会最適化 165
データ・アクセス方式 165
データ・アクセス方式 166
索引スキャンによるデータ・アクセス 166
索引アクセスのタイプ 169
索引アクセスとクラスター率 172
述部の用語 173
結合方式およびストラテジー 175
結合 175
結合方式 176
最適の結合を選択する方法 179
パーティション・データベースの複製マテリアライズ照会表 181
パーティション・データベースでの結合ストラテジー 183
パーティション・データベースでの結合方式 . . 185
ソートとグループ化の影響 191
最適化ストラテジー 193
パーティション内並列処理の最適化ストラテジー 193
MDC 表の最適化ストラテジー 195
マテリアライズ照会表 196
フェデレーテッド・データベースの照会コンパイラー・フェーズ 199
フェデレーテッド・データベースのプッシュダウン分析. 199
フェデレーテッド照会を評価する場合の分析のガイドライン 204
フェデレーテッド・データベースにおけるリモート SQL 生成とグローバル最適化. 206
フェデレーテッド・データベース照会のグローバル分析. 209

第 7 章 SQL Explain 機能. 213
SQL Explain 機能 213
Explain 情報の収集および分析用のツール 214
Explain ツール 214
Explain 情報の使用のガイドライン 216
収集された Explain 情報 217
Explain 表および Explain 情報の編成 217
データ・オブジェクトの Explain 情報 219
データ・オペレーターの Explain 情報 220
インスタンスの Explain 情報 221
Explain 情報のキャプチャーのガイドライン . . 223
Explain 情報の分析のガイドライン 226
設計アドバイザー 227
設計アドバイザーの出力. 228
設計アドバイザーのワークロードの定義 230
設計アドバイザーを使用した、単一パーティション・データベースから複数パーティション・データベースへの移行. 231
設計アドバイザーの制限と制約事項 232

第 3 部 システムのチューニングと構成 235

第 8 章 操作上のパフォーマンス . . . 237
メモリー使用 237
メモリー使用の編成 237
データベース・マネージャー共用メモリー. . . 240
FCM バッファー・プールとメモリーの要件 . . 242
グローバル・メモリーとその制御パラメーター 243
メモリー使用に影響を与えるパラメーターのチューニングのガイドライン. 245
バッファー・プール 248
バッファー・プール管理. 248
32 ビット・プラットフォームにおける拡張メモリー内の 2 次バッファー・プール 250
データ・ページのバッファー・プール管理. . . 251
先行ページ・クリーニング 253
バッファー・プールのデータ・ページ管理の図 254
複数のデータベース・バッファー・プールの管理 255
プリフェッチの概念 258
バッファー・プールへのデータのプリフェッチ 258
順次プリフェッチ 259
改善された順次プリフェッチ用のブロック・ベースのバッファー・プール. 261
リスト・プリフェッチ 263
入出力管理 263
プリフェッチと並列処理のための入出力サーバー構成 263

並列入出力を使用したプリフェッチの図 . . . 264
並列入出力の管理 266
ソート・パフォーマンスのガイドライン . . . 268
表管理. 270
表の再編成 270
表の再編成時の決定 272
表を再編成する方式の選択 275
索引管理 277
索引の利点と欠点 277
索引の計画のヒント 279
索引のパフォーマンスのヒント 282
索引のクリーン・アップおよび保守 285
索引の再編成 287
オンライン索引のデフラグ 289
DMS 装置に関する考慮事項 290
エージェント管理 291
データベース・エージェント 291
データベース・エージェント管理. 294
エージェントの数に影響を与える構成パラメーター 294
クライアント接続用の接続コンセントレーターの改善 295
パーティション・データベースにおけるエージェント 298
データベース・システム・モニター情報 299

第 9 章 ガバナー・プログラムの使用 303
ガバナー・ユーティリティー 303
ガバナー・の始動およびシャットダウン 304
ガバナーの開始と停止 304
ガバナー・デーモン 305
ガバナーの構成. 306
ガバナーの構成. 306
ガバナー構成ファイル 307
ガバナーの規則のエレメント 310
ガバナー構成ファイルの例 315
ガバナー・ログ・ファイルの使用. 316
ガバナー・ログ・ファイル 316
ガバナー・ログ・ファイルの照会. 320

第 10 章 構成のスケーリング 321
データベース・サーバー容量の管理 321
パーティション・データベースのパーティション 322
実行中のデータベース・システムへのパーティションの追加 324
Windows NT 上で停止中のデータベース・システムへのパーティションの追加 325
UNIX 上で停止中のデータベース・システムへのパーティションの追加 327
ノード追加エラーのリカバリー 329
データベース・パーティションのドロップ. . . . 330

第 11 章 データベース・パーティション間でのデータの再分散 333
データの再分散. 333
データ再分散の決定 335
パーティション間でのデータの再分散 336
データ再分散のログ・スペース所要量 338
再分散エラー・リカバリー 339
再分散ストアード・プロシージャーおよび関数 . . 340
get_swrd_settings ストアード・プロシージャー 340
set_swrd_settings ストアード・プロシージャー 341
analyze_log_space ストアード・プロシージャー 342
generate_Distfile ストアード・プロシージャー 343
stepwise_redistribute_dbpg ストアード・プロシージャー. 344
db_partitions UDF 345
使用例. 345

第 12 章 ベンチマーク・テスト 347
ベンチマーク・テスト 347
ベンチマークの準備 348
ベンチマーク・テストの作成 350
db2batch テストの例 352
ベンチマーク・テストの実行 356
ベンチマーク・テストの分析例 358

第 13 章 DB2 の構成 361
構成パラメーター 361
構成パラメーターの調整. 363
構成パラメーターによる DB2 の構成 364
パラメーターの動的構成. 367
構成パラメーターのサマリー 370
データベース・マネージャー構成パラメーター・サマリー 370
データベース構成パラメーターのサマリー. . . 376
DB2 Administration Server (DAS) 構成パラメーターのサマリー. 384
機能別のパラメーター詳細 384
キャパシティー管理 385
データベース共用メモリー 386
アプリケーション共用メモリー 398
エージェント専用メモリー 401
エージェント/アプリケーション通信メモリー 412
データベース・マネージャー・インスタンス・メモリー. 417
ロック. 424
入出力およびストレージ. 428
エージェント 436
ストアード・プロシージャーおよびユーザー定義関数 448
ロギングおよびリカバリー 451
データベース・ログ・ファイル 452
データベース・ログ・アクティビティー . . . 463
リカバリー 474
分散作業単位リカバリー. 487
データベース管理 492
Query Enabler 492
属性 493
DB2 Data Links Manager 496
状況 500
コンパイラーの設定 503

| 自動保守 510
通信 512
通信プロトコルの設定 512
DB2 ディスカバリー機能 515
パーティション・データベース環境 517
通信 517
並列処理 525
インスタンス管理 527
診断 527
データベース・システム・モニター・パラメーター 531
システム管理 532
インスタンス管理 542
DB2 Administration Server 558
authentication - 認証タイプ DAS 558
contact_host - 連絡先リストのロケーション . . 559
das_codepage - DAS コード・ページ 560
das_territory - DAS テリトリー 560
dasadm_group - DAS 管理者権限グループ名 . . 561
db2system - DB2 サーバー・システムの名前 . . 561
discover - DAS ディスカバリー・モード . . . 562
exec_exp_task - 有効期限切れタスクの実行 . . 563
jdk_64_path - 64 ビット Software Developer's Kit for Java インストール・パス DAS 563
jdk_path - Software Developer's Kit for Java インストール・パス DAS 564
sched_enable - スケジューラー・モード. . . . 565
sched_userid - スケジューラー・ユーザー ID 565
smtp_server - SMTP サーバー 566
toolscat_db - ツール・カタログ・データベース 567
toolscat_inst - ツール・カタログ・データベース・インスタンス 567
toolscat_schema - ツール・カタログ・データベース・スキーマ 568

第 4 部 付録 569

付録 A. DB2 レジストリー変数と環境変数 571
DB2 レジストリー変数と環境変数 571
カテゴリー別のレジストリー変数および環境変数 572
汎用レジストリー変数 572
システム環境変数 575
通信変数 579
コマンド行変数. 583
MPP 構成変数 584
SQL コンパイラー変数 586
パフォーマンス変数 591
データ・リンク変数 605
その他の変数 607

付録 B. Explain 表 615
Explain 表 615
EXPLAIN_ARGUMENT 表 616
EXPLAIN_INSTANCE 表 620
EXPLAIN_OBJECT 表 622
EXPLAIN_OPERATOR 表 625
EXPLAIN_PREDICATE 表 627
EXPLAIN_STATEMENT 表. 629
EXPLAIN_STREAM 表 632
ADVISE_INDEX 表 634
| ADVISE_INSTANCE 表 637
| ADVISE_MQT 表 638
| ADVISE_PARTITION 表. 640
| ADVISE_TABLE 表 641
ADVISE_WORKLOAD 表 642

付録 C. SQL Explain ツール 643
SQL Explain ツール 643
db2expln 644
db2expln - SQL Explain 644
db2expln の使用上の注意 649
dynexpln 651
Explain 出力情報 651
db2expln および dynexpln 出力の説明 651
表アクセス情報. 653
一時表の情報 658
結合の情報 660
データ・ストリーム情報. 662
挿入、更新、および削除の情報 663
ブロックと行 ID の準備の情報 664
集約の情報 665
並列処理の情報. 665
フェデレーテッド照会の情報 668
その他の情報 669
db2expln および dynexpln 出力の例 671
db2expln および dynexpln 出力の例 671
例 1: 非並列 672
例 2: パーティション内並列処理による単一パーティションのプラン 674
例 3: パーティション間並列処理による複数パーティションのプラン 675
例 4: パーティション間並列処理とパーティション内並列処理による複数パーティションのプラン 678
例 5 : フェデレーテッド・データベースのプラン 680

付録 D. db2exfmt - Explain 表形式 683

| 付録 E. db2adutl コマンドと | logarchopt1 および vendoropt データ | ベース構成パラメーターを使ったノード | 間リカバリー 685

付録 F. DB2 Universal Database の技術情報 691
DB2 資料とヘルプ. 691
| DB2 資料の更新 691
DB2 インフォメーション・センター. 692

| DB2 インフォメーション・センターのインストール・シナリオ 694
| DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX) 696
| DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows) 699
DB2 インフォメーション・センターの呼び出し 702
コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール 703
| DB2 インフォメーション・センターにおける特定の言語でのトピックの表示 704
DB2 PDF 資料および印刷された資料 705
| DB2 の基本情報 705
管理情報 706
アプリケーション開発情報 707
ビジネス・インテリジェンス情報. 708
DB2 Connect 情報. 708
入門情報 708
チュートリアル情報 709
オプショナル・コンポーネント情報 709
リリース・ノート 710
PDF ファイルからの DB2 資料の印刷方法 . . . 711
DB2 の印刷資料の注文方法. 712
DB2 ツールからコンテキスト・ヘルプを呼び出す 713
| コマンド行プロセッサーからメッセージ・ヘルプを呼び出す 714
| コマンド行プロセッサーからコマンド・ヘルプを呼び出す. 714
| コマンド行プロセッサーから SQL 状態ヘルプを呼び出す. 715
DB2 チュートリアル 715
DB2 トラブルシューティング情報 716
アクセス支援 717
キーボードによる入力およびナビゲーション . . 717
アクセスしやすい表示 718
支援テクノロジーとの互換性 718
アクセスしやすい資料 718
| ドット 10 進シンタックス・ダイアグラム. . . . 719
| DB2 Universal Database 製品の共通基準認証 . . . 721

付録 G. 特記事項 723
商標 725

索引 727

IBM と連絡をとる. 741
製品情報 741

本書について

3 巻で構成される本書には、 DB2 リレーショナル・データベース管理システム (RDBMS) 製品を使用し管理するために必要な、以下の情報が収められています。

- データベース設計についての情報 (「*管理ガイド: プランニング*」)
- データベースの使用および管理についての情報 (「*管理ガイド: インプリメンテーション*」)
- パフォーマンスを向上させるための、データベース環境の構成およびチューニングについての情報 (「*管理ガイド: パフォーマンス*」)

本書に記載されているタスクの多くは、以下に示すさまざまなインターフェースを使用して実行することができます。

- **コマンド行プロセッサー**。グラフィカル・インターフェースから、データベースをアクセスし操作することができます。このインターフェースから、SQL ステートメントおよび DB2 ユーティリティー関数も実行することができます。本書にある例のほとんどが、このインターフェースを使った場合を例として取り上げています。コマンド行プロセッサーの使用に関する詳細は、「コマンド・リファレンス」を参照してください。
- **アプリケーション・プログラミング・インターフェース**。アプリケーション・プログラム内で DB2 ユーティリティー関数を実行することができます。アプリケーション・プログラミング・インターフェースの使用についての詳細は、「*管理 API リファレンス*」を参照してください。
- **コントロール・センター**。グラフィカル・ユーザー・インターフェースを使用して、システムの構成、ディレクトリーの管理、システムのバックアップとリカバリー、ジョブのスケジューリング、およびメディアの管理などの管理タスクを実行することができます。またコントロール・センターには、システム間のデータの複製をセットアップするための複製管理が含まれています。さらに、コントロール・センターでは、グラフィカル・ユーザー・インターフェースを介して DB2 ユーティリティー機能を実行できます。ご使用のプラットフォームによっては、コントロール・センターを呼び出すさまざまな方法があります。たとえば、Windows プラットフォームでは、コマンド行で `db2cc` コマンドを使用するか、DB2 フォルダーから「コントロール・センター」アイコンを選択したり、あるいは「スタート」メニューを使用します。紹介のヘルプを表示するには、「コントロール・センター (Control Center)」ウィンドウの**「ヘルプ (Help)」**プルダウンから**「入門 (Getting started)」**を選択してください。 **Visual Explain** ツールは、コントロール・センターから呼び出します。

  コントロール・センターでは、3 つのビューが使用可能です。

  – 基本。このビューには、データベース、表、およびストアード・プロシージャーなどの、中核をなす DB2 UDB 機能が表示されます。
  – 詳細。このビューには、使用可能なすべてのオブジェクトおよびアクションが表示されます。エンタープライズ環境で作業しており、DB2 for z/OS または IMS に接続する場合には、このビューを使用してください。

– カスタム。このビューでは、オブジェクト・ツリーやオブジェクト・アクションを調整できるようになっています。

他にも、管理タスクを行うために使用できるツールがあります。それらのツールには、以下のものがあります。

- コマンド・エディター。これはコマンド・センターに替わるもので、 SQL ステートメントや IMS および DB2 コマンドの生成、編集、実行、および操作に使用できます。さらに、出力結果の処理、EXPLAIN された SQL ステートメントのアクセス・プランのグラフィカル表現を表示するために使用します。
- デベロップメント・センター。ネイティブ SQL Persistent Storage Module (PSM) ストアード・プロシージャー、 iSeries バージョン 5 リリース 3 以降の Java ストアード・プロシージャー、ユーザー定義関数 (UDF)、および構造化タイプをサポートします。
- ヘルス・センターは、 DBA がパフォーマンスおよびリソース割り振りの問題を解決する上で役立つツールを提供します。
- ツール設定は、コントロール・センター、ヘルス・センター、およびレプリケーション・センターの設定値を変更します。
- ジャーナルは、自動で実行するジョブをスケジュールします。
- データウェアハウス・センターは、ウェアハウス・オブジェクトを管理します。

本書の対象読者

本書は、ローカルまたはリモート・クライアントがアクセスするデータベースを設計、使用、および維持する必要のあるデータベース管理担当者、システム管理者、セキュリティー管理者、およびシステム・オペレーターを対象としています。 DB2 Universal Database™ (DB2 UDB) リレーショナル・データベース管理システムの管理および操作について理解しておくことが必要なプログラマーや、その他のユーザーも本書をご使用になれます。

本書の構成

本書には、以下の主な項目に関する情報が記載されています。

パフォーマンスの紹介

- 『第 1 章 パフォーマンスの紹介』では、 DB2 UDB パフォーマンスを管理と改善に関する概念と考慮事項について紹介します。
- 『第 2 章 アーキテクチャーとプロセス』では、基礎となる DB2 Universal Database の構造およびプロセスを紹介します。

アプリケーション・パフォーマンスのチューニング

- 『第 3 章 アプリケーションについての考慮事項』では、アプリケーションの設計時点で、データベースのパフォーマンスを向上させる手法をいくつか説明します。
- 『第 4 章 環境についての考慮事項』では、データベース環境の設定時点で、データベースのパフォーマンスを向上させる手法をいくつか説明します。
- 『第 5 章 システム・カタログ統計』では、最適なパフォーマンスを達成するために、データについての統計を収集し使用する方法について説明します。

- 『第 6 章 SQL コンパイラーに関する解説』では、 SQL コンパイラーを使用してコンパイルしたときに、 SQL ステートメントがどうなるかについて説明します。
- 『第 7 章 SQL Explain 機能』では、Explain 機能について説明します。この機能により、データにアクセスするために SQL コンパイラーが行った選択を調べることができます。

システムのチューニングと構成

- 『第 8 章 操作上のパフォーマンス』では、データベース・マネージャーがメモリーを使用する方法の概要、およびランタイムのパフォーマンスに影響を与えるその他の考慮事項について説明します。
- 『第 9 章 ガバナー・プログラムの使用』では、データベース管理のある局面を制御するためのガバナーの使用について紹介します。
- 『第 10 章 構成のスケーリング』では、データベース・システムのサイズの増加に関連する考慮事項と作業について説明します。
- 『第 11 章 データベース・パーティション間でのデータの再分散』では、パーティション間でデータを再分散するために、パーティション・データベース環境において必要な作業について説明します。
- 『第 12 章 ベンチマーク・テスト』では、ベンチマーク・テストの概要とベンチマーク・テストの実行方法について説明します。
- 『第 13 章 DB2 の構成』では、データベース・マネージャー、データベース構成ファイルと、データベース・マネージャー、データベース、および DAS 構成パラメーターの値について説明します。

付録

- 『付録 A. DB2 レジストリー変数と環境変数』では、プロファイル・レジストリーの値と環境変数を示します。
- 『付録 B. Explain 表』は、 DB2 Explain 機能が使用する表についての情報を示し、それらの表の作成方法について説明する、 Explain 表に関するセクションです。
- 『付録 C. SQL Explain ツール』では、 DB2 Explain ツールである db2expln および dynexpln の使用法について説明します。
- 『付録 D. db2exfmt - Explain 表形式』では、 DB2 Explain 表の内容をフォーマットします。

他の管理ガイドの巻の概要

管理ガイド : プランニング

「*管理ガイド: プランニング*」は、データベース設計を扱っています。論理設計および物理設計、分散トランザクションについて説明しています。この巻のそれぞれの章と付録は、以下のように構成されています。

データベースの概念

- 『リレーショナル・データベースの基本的な概念』では、リカバリー・オブジェクト、ストレージ・オブジェクト、およびシステム・オブジェクトを含むデータベース・オブジェクトの概要を説明します。
- 『並列データベース・システム』では、 DB2 で実現される並列処理のタイプを紹介します。
- 『データウェアハウジングについて』では、データウェアハウジングおよびウェアハウジング・タスクについて概説します。

データベースの設計

- 『論理データベースの設計』では、論理データベースの設計に関する概念と指針について説明します。
- 『物理データベースの設計』では、物理データベースの設計 (データ・ストレージに関する考慮事項を含む) の指針について説明します。
- 『分散データベースの設計』では、単一トランザクションで複数のデータベースをアクセスする方法を説明します。
- 『トランザクション・マネージャーの設計』では、分散トランザクション処理環境でデータベースを使用する方法について説明します。

付録

- 『リリース間の非互換性』では、バージョン 7 とバージョン 8 での非互換性と、意識していなければならない将来の非互換性について示します。
- 『各国語サポート (NLS)』では、テリトリー、言語、およびコード・ページの情報を含む、 DB2 各国語サポート (NLS) について説明します。
- 『64 ビット環境におけるラージ・ページのサポートの使用可能化 (AIX)』では、16 MB ページ・サイズのサポート、およびこのサポートを使用可能にする方法を取り上げます。

管理ガイド : インプリメンテーション

「*管理ガイド: インプリメンテーション*」では、データベース設計の実装について扱います。この巻のそれぞれの章と付録は、以下のように構成されています。

設計の実現

- 『データベースを作成する前に』では、データベースとデータベース内のオブジェクトを作成する前に求められる前提条件について説明します。
- 『DB2 Administration Server (DAS) の作成と使用』では、 DAS およびその作成と使用方法を取り上げます。
- 『データベースの作成』では、データベースおよび関係するデータベース・オブジェクトの作成に関連したタスクを説明します。
- 『表および関連する表オブジェクトの作成』では、データベース設計をインプリメントする際の特定の特性を持つ表の作成方法について説明します。
- 『データベースの変更』では、データベースを変更する前に実行しなければならないタスクや、データベースまたは関係するデータベース・オブジェクトの変更またはドロップに関するタスクについて説明します。

- 『表および関連する表オブジェクトの変更』では、表をドロップする方法、あるいはそれらの表に関連付けられた特定の特性を変更する方法について取り上げます。関連する表オブジェクトのドロップと変更についても説明します。

データベースのセキュリティー

- 『データベース・アクセスの制御』では、データベース・リソースへのアクセスを制御する方法について説明します。
- 『DB2 アクティビティーの監査』では、データに対する不要なアクセスまたは予期せぬアクセスを検出してモニターする方法について説明します。

付録

- 『命名規則への準拠』では、データベースおよびオブジェクトを命名する際に従わなければならない規則を示します。
- 『クライアントの自動転送の使用』では、クライアント・アプリケーションの自動転送およびこのサポートを使用可能にする方法を取り上げます。
- 『Lightweight Directory Access Protocol (LDAP) ディレクトリー・サービスの使用』では、 LDAP ディレクトリー・サービスを使用する方法について説明します。
- 『複数のデータベース・パーティションに対するコマンドの発行』では、コマンドをパーティション・データベース環境内のすべてのパーティションに送るための *db2_all* および *rah* シェル・スクリプトの使用法について説明します。
- 『Windows Management Instrumentation (WMI) サポート』では、さまざまなハードウェアおよびソフトウェア管理システムを統合するために、 DB2 がどのようにこの管理インフラストラクチャー規格をサポートしているかについて説明します。また、 DB2 がどのように WMI と統合されているかについても説明します。
- 『Windows NT セキュリティーの使用』では、 DB2 Universal Database が Windows NT セキュリティーと連動する方法を取り上げます。
- 『Windows パフォーマンス・モニターの使用』では、 Windows NT パフォーマンス・モニターへの DB2 の登録についての情報と、パフォーマンス情報を使用する方法についての情報を示します。
- 『Windows データベース・パーティション・サーバーの使用』では、 Windows NT または Windows 2000 上でデータベース・パーティション・サーバーを動作させるために使用できるユーティリティーに関する情報が示されています。
- 『複数の論理ノードの構成』では、パーティション・データベース環境で複数論理ノードを構成する方法について説明します。
- 『コントロール・センターの拡張』では、新しいアクションを割り当てた新しいツールバー・ボタンの追加、新しいオブジェクト定義の追加、および新しいアクション定義の追加により、コントロール・センターを拡張する方法について説明します。

注: この本からは 2 つの章が除去されています。

「*コマンド・リファレンス*」および「*管理 API リファレンス*」に含まれていた、データの移動に使用する DB2 ユーティリティーに関するすべての情報、およびそれに類似したトピックが、「*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*」に統合されました。

「*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*」が、これらのトピックに関する主要かつ唯一の情報源です。

データの複製に関する詳細については、「*IBM DB2 Information Integrator SQL レプリケーション・ガイドおよびリファレンス*」を参照してください。

「*コマンド・リファレンス*」および「*管理 API リファレンス*」に含まれていた、データのバックアップおよびリカバリーの方法およびツールに関するすべての情報が、「*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*」に統合されました。

「*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*」が、これらのトピックに関する主要かつ唯一の情報源です。

第 1 部 パフォーマンスの紹介

第 1 章 パフォーマンスの紹介

この章のセクションでは、パフォーマンスのチューニングについて説明し、以下についての提案を示します。

- パフォーマンスのモニターおよびチューニング・プランの作成
- パフォーマンス上の問題に関するユーザー情報の使用
- 初期パフォーマンス・チューニングの即時開始

パフォーマンスのエレメント

パフォーマンス とは、特定のワークロードがかかっているコンピューター・システムの動作の仕方のことです。パフォーマンスは、システムの応答時間、スループット、および可用性によって測定されます。また、次の事柄の影響も受けます。

- システムで使用可能なリソース
- リソースの使用頻度と共用の程度

一般に、システムを調整することにより、その費用便益比率を向上させます。これには、次のような目標があります。

- 処理費用を増やすことなく、より大きいワークロードまたは要求がより多いワークロードを処理する

  たとえば、新しいハードウェアを購入したりプロセッサー時間を長くしたりすることなくワークロードを増やすことができます。
- 処理費用を増やすことなく、システムの応答時間を速くしたり、スループットを高くしたりする
- ユーザーに対するサービスを低下させることなく、処理費用を削減する

パフォーマンスを技術的な用語から経済学的な用語に翻訳することは簡単ではありません。パフォーマンスのチューニングには、確かに、人件費やプロセッサー時間などによる費用がかかるので、チューニングを行う前に、費用と見込まれる効果とを比較考察してください。これらの効果には、次のようにはっきり分かるものがあります。

- リソースをより効率的に使用できるようになる
- システムにより多くのユーザーを追加できるようになる

応答時間が速くなったことによるユーザーの満足度の向上などのその他の効果は、はっきりとは分かりません。これらすべての効果について考慮する必要があります。

関連概念:

- 4 ページの『パフォーマンス・チューニングのガイドライン』
- 7 ページの『パフォーマンス・チューニングのためのクイック・スタート・ヒント』

関連タスク:

- 5 ページの『パフォーマンスの向上プロセスの開発』

パフォーマンス・チューニングのガイドライン

次の指針は、パフォーマンス・チューニングの全体的なアプローチを決定するのに役立ちます。

収益逓減の法則を覚えておく: 通常、パフォーマンスの効果を最大にするには、最初の努力が重要です。一般にその後の変更では、得られる効果は少なくなり、必要な努力は多くなります。

チューニングのためのチューニングを行わない: チューニングは、識別されている制約を軽減するために行ってください。パフォーマンス上の問題の主要な原因ではないリソースのチューニングを行っても、応答時間に対する効果は主要な制約を軽減するまでほとんどまたはまったくなく、それ以降のチューニング作業が行いにくくなります。 著しく改善できる可能性があるとすれば、それは応答時間の主要な要因となっているリソースのパフォーマンスを改善することにあります。

システム全体を考慮する: 他に影響を与えることなく 1 つのパラメーターまたはシステムのチューニングを行うことはできません。調整を行う前に、その調整によってシステム全体がどのような影響を受けるかを考慮してください。

一度に 1 つのパラメーターを変更する: 一度に複数のパフォーマンス・チューニング・パラメーターを変更しないでください。すべての変更が有益であることを確信している場合でも 、それぞれの変更の効果を評価することができなくなります。一度に複数のパラメーターを変更すると、行ったトレードオフを効果的に判断することもできません。 1 つのパラメーターを調整して 1 つの分野を改善すると、ほとんどの場合考慮に入れていなかった少なくとも 1 つの別の分野が影響を受けます。一度に 1 つずつ変更を行うことにより、希望どおりの改善が行われたかどうかをベンチマークを使用して評価することができます。

レベルごとに測定と再構成を行う: 一度に変更するパラメーターを 1 つにするのと同じ理由で、一度にチューニングするシステムのレベルは 1 つにしてください。次のリストは、システム内の各レベルを示しています。

- ハードウェア
- オペレーティング・システム
- アプリケーション・サーバーとリクエスター
- データベース・マネージャー
- SQL ステートメント
- アプリケーション・プログラム

ハードウェアおよびソフトウェアの問題を検査する: 一部のパフォーマンス問題は、ハードウェアかソフトウェアのどちらか (あるいはその両方) にサービスを適用することによって修正することができます。単にサービスを適用するだけで済む場合は、システムのモニターとチューニングに余分な時間がかかりません。

ハードウェアをアップグレードする前に問題を理解する: ストレージを増やしたりプロセッサーの能力を高めたりすることでパフォーマンスを即座に改善できるように思えても、時間を取って障害がどこに存在しているのかを理解してください。費用をかけてディスク装置を追加しても、それを活用するだけの処理能力やチャネルがないことに気付く場合があります。

チューニングを始める前にフォールバック手順を実施する: 前述のとおり、チューニングを行うと予想外のパフォーマンス結果が生じることがあります。パフォーマンスが低下した場合は、そのチューニングを元に戻して別のチューニングを行う必要があります。簡単に元に戻せるように元の設定を保管しておけば、誤った情報を取り消すことはずっと簡単になります。

関連概念:

- 3 ページの『パフォーマンスのエレメント』
- 7 ページの『パフォーマンス・チューニングのためのクイック・スタート・ヒント』

関連タスク:

- 5 ページの『パフォーマンスの向上プロセスの開発』

パフォーマンス・チューニング処理

パフォーマンスのモニターおよびチューニング・プランは、ユーザー入力を考慮に入れ、ご使用のシステムでのチューニングの限界も理解して決定します。

パフォーマンスの向上プロセスの開発

パフォーマンスの向上プロセスは、パフォーマンスのモニターおよびチューニングの局面への、対話式かつ長期的なアプローチです。モニターの結果に基づいて、ユーザーおよびユーザーのパフォーマンス・チームはデータベース・サーバーの構成を調整し、データベース・サーバーを使用するアプリケーションに変更を加えます。

パフォーマンスのモニターおよびチューニングは、データ、およびデータ・アクセスのパターンを使用するアプリケーションの種類に関するユーザーの知識に基づいて決定します。パフォーマンス要件は、アプリケーションの種類によって異なります。

以下のパフォーマンスの向上プロセスの概要を指針として検討してください。

手順:

パフォーマンスの向上プロセスを作成するには、以下を行います。

1. パフォーマンスの目標を定義します。
2. システムで主要な制約のパフォーマンス指針を設定します。
3. パフォーマンスのモニター計画を立て、それを実行します。
4. 継続的にモニター結果を分析し、チューニングを必要とするリソースを判別します。
5. 一度に 1 つの調整を行う。

複数のリソースがチューニングを必要としていると判断する場合、あるいは調整したいリソースでいくつかのチューニング・オプションを使用できる場合でも、変更は一度に 1 つのみにしてください。そうすることで、そのチューニング努力によって得られる結果は必ず望みどおりのものとなります。ある時点で、データベース・サーバーおよびアプリケーションのチューニングをしても、パフォーマンスが向上しなくなる場合があります。その場合は、ハードウェアをアップグレードする必要があります。

実際のパフォーマンス・チューニングでは、システム・リソース間のトレードオフが必要になります。たとえば、入出力パフォーマンスを向上させるために、バッファー・プール・サイズを増やすことができます。しかし、バッファー・プールを増やすにはより多くのメモリーを必要とするため、パフォーマンスの他の局面は低下する可能性があります。

関連概念:

- 3 ページの『パフォーマンスのエレメント』
- 4 ページの『パフォーマンス・チューニングのガイドライン』
- 7 ページの『パフォーマンス・チューニングのためのクイック・スタート・ヒント』
- 7 ページの『パフォーマンス・チューニングの限界』
- 6 ページの『ユーザーが提供できるパフォーマンス情報』

ユーザーが提供できるパフォーマンス情報

システム・チューニングが必要なことを示唆する最初の兆候は、ユーザーからの問題報告かもしれません。パフォーマンス目標を設定する時間や、包括的にモニターとチューニングを行う時間が十分にない場合は、ユーザーの意見を聞いてパフォーマンスの問題と取り組むことができます。通常は 2、3 の簡単な質問をするだけで、問題を突き止めるためにどこを調べればよいか、見当がつきます。たとえば、ユーザーに次のような質問をしてみることができます。

- 「応答が遅い」とは、どのような意味でしょうか? 期待している速さより 10 % 遅いのでしょうか、それとも何十倍も遅いのでしょうか?
- 問題に気付いたのはいつですか? 最近ですか、それともこれまでずっとですか?
- 他のユーザーも同じような問題に直面していますか? 問題を訴えているユーザーは 1 人か 2 人ですか、それともグループ全体ですか?
- ユーザーのグループ全体が同じ問題に直面している場合、それらのユーザーは同じローカル・エリア・ネットワークに接続していますか?
- 特定のトランザクションまたはアプリケーション・プログラムに関連した問題と思われますか?
- 問題の発生の仕方に何らかのパターンがありますか? たとえば、一日の特定の時間 (昼休み中など) に問題が発生しますか、それともある程度連続的に発生しますか?

関連概念:

- 4 ページの『パフォーマンス・チューニングのガイドライン』

関連タスク:

- 5 ページの『パフォーマンスの向上プロセスの開発』

パフォーマンス・チューニングの限界

チューニングによってシステム効率を改善することには、限界があります。システム・パフォーマンスの改善にかかる時間と費用、また時間と費用をさらにかけることでシステムのユーザーにもたらされる利益を考慮してください。

たとえば、システムにボトルネックが発生する場合には、チューニングによってパフォーマンスが改善されます。システム・パフォーマンスの限界に近づいているとき、ユーザーの数を約 10 % 増やすと、応答時間は 10 % よりずっと長くなる可能性があります。この場合、どのようにシステムをチューニングして、このパフォーマンス低下を相殺すべきかを判別する必要があります。

しかし、チューニングを行ってもこれ以上は効果がないという限界点が存在します。それに達したときは、目標と期待値を環境における限界の範囲内に修正する必要があります。パフォーマンスを大幅に改善するには、ディスク装置の増加、 CPU の高速化、CPU の追加、メイン・メモリーの増加、通信リンクの高速化、またはこれらを組み合わせる必要があるかもしれません。

関連概念:

- 321 ページの『データベース・サーバー容量の管理』

関連タスク:

- 5 ページの『パフォーマンスの向上プロセスの開発』

パフォーマンス・チューニングのためのクイック・スタート・ヒント

DB2® の新しいインスタンスを開始するとき、基本構成に関する以下の提案を検討してください。

- コントロール・センターの構成アドバイザーを使用して、システムに適切な開始デフォルトについての提案を入手します。 DB2 の出荷時のデフォルトは、ユニークなハードウェア環境に合わせて調整する必要があります。

  サイトで使用しているハードウェアについての情報を収集して、ウィザードの質問に応答できるようにします。提案された構成パラメーターの設定値を即時に適用するか、またはウィザードが応答に基づいてスクリプトを作成するようにして、そのスクリプトを後に実行することができます。

  このスクリプトは、後に参照するための最も一般的に調整されるパラメーターのリストも提供します。
- コントロール・センターおよび Client 構成アシスタントの他のウィザードを使用して、パフォーマンスに関連した管理タスクを行います。これらのタスクは通常、少しの時間と努力を費やすことによって多大のパフォーマンス向上を達成できるものです。

  その他のウィザードは、個々の表と一般のデータ・アクセスのパフォーマンスを改善する点で役立ちます。これらのウィザードには、「データベース作成 (Create Database)」「表作成 (Create Table)」、「索引 (Index)」、および「マルチサイト

更新の構成 (Configure Multisite Update)」ウィザードがあります。ヘルス・センターは、モニター・ツールおよびチューニング・ツールのセットを提供します。

- コントロール・センターから設計アドバイザー・ツールを使用するか、または `db2advis` コマンドを使用して、どの索引、マテリアライズ照会表、マルチディメンション・クラスタリング (MDC) 表、およびデータベース・パーティションが照会パフォーマンスを向上させるかを判別します。
- ACTIVATE DATABASE コマンドを使用してデータベースを開始します。パーティション・データベースでは、このコマンドはすべてのパーティション上のデータベースを活動化して、最初のアプリケーションが接続するときにデータベースを初期化するための起動時間を不要にします。

  注: ACTIVATE DATABASE コマンドを使用する場合、 DEACTIVATE DATABASE コマンドを使用してデータベースをシャットダウンしなければなりません。データベースから最後に切断されるアプリケーションは、データベースをシャットダウンしません。
- データベース・マネージャーおよび各データベースに使用可能な各構成パラメーターをリストして短く説明しているサマリー表を参照してください。

  これらのサマリー表には、パラメーターを調整するとパフォーマンスが大きく、中程度に、または少なく向上、または低下するか、あるいは変化しないかを示す列が含まれています。この表を使用して、パフォーマンスの改善を最大にするために調整できるパラメーターを見つけてください。

関連概念:

- 299 ページの『データベース・システム・モニター情報』

関連資料:

- 370 ページの『構成パラメーターのサマリー』

第 2 章 アーキテクチャーとプロセス

この章では、DB2 アーキテクチャーおよびプロセス・スキーマについての一般情報を示します。

DB2 アーキテクチャーおよびプロセスの概要

DB2® アーキテクチャーおよびプロセスに関する一般情報により、特定のトピックに提供される詳細情報を理解しやすくなります。

次の図に、DB2 UDB のアーキテクチャーとプロセスの概要を示します。

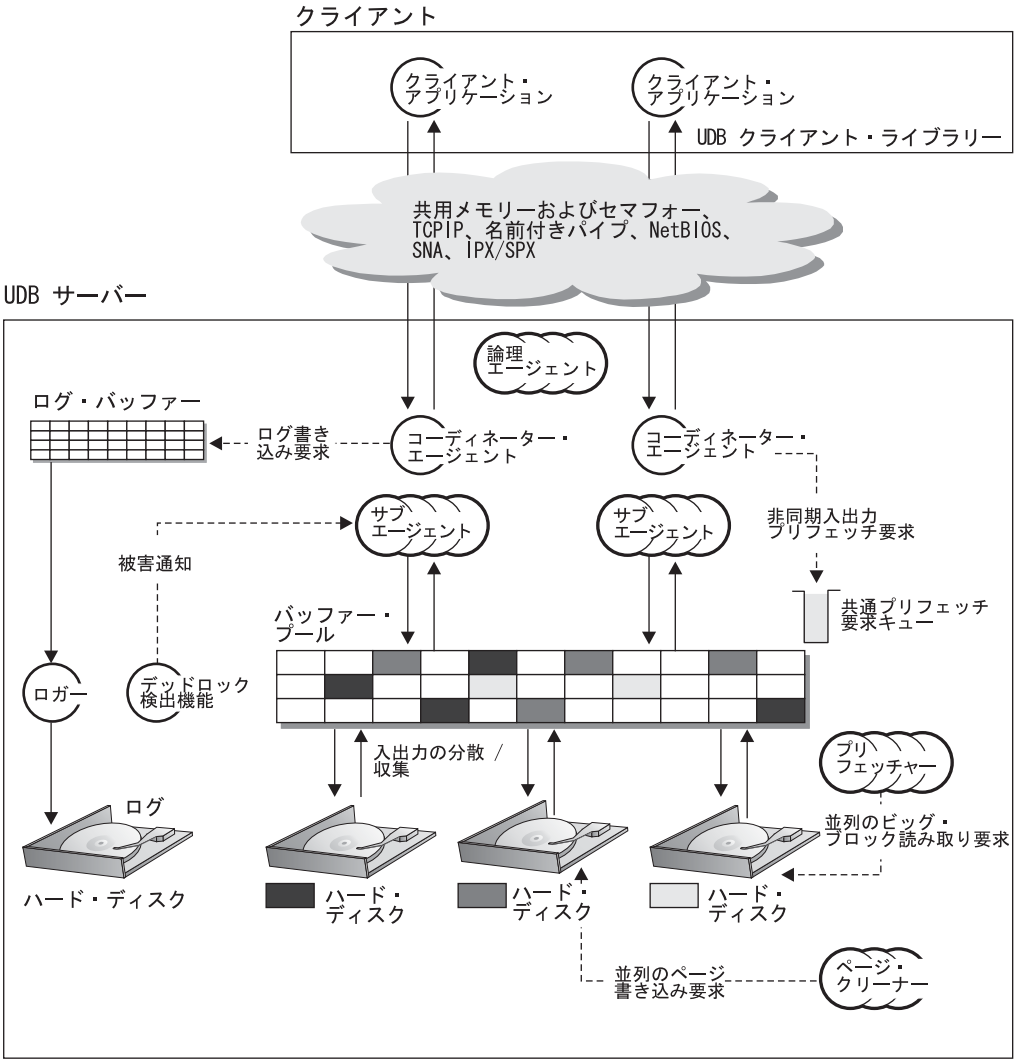


図 1. アーキテクチャーとプロセスの概要

クライアント・サイドでは、ローカルまたはリモート・アプリケーションのどちらかまたは両方が、 DB2 Universal Database™ クライアント・ライブラリーとリンク

しています。ローカル・クライアントは、共用メモリーおよびセマフォーを使用して通信します。リモート・クライアントは、名前付きパイプ (NPIPE)、TCP/IP、NetBIOS、または SNA などのプロトコルを使用します。

サーバー側では、アクティビティーはエンジン・ディスパッチ可能単位 (EDU) により制御されます。このセクションのすべての図で、 EDU は、円または円グループとして示されています。 EDU は、Windows® ベースのプラットフォームではスレッドとして、 UNIX® ではプロセスとしてインプリメントされています。 DB2 エージェントは、最も一般的なタイプの EDU です。これらのエージェントは、アプリケーションに代わって SQL 処理のほとんどを実行します。プリフェッチャーおよびページ・クリーナーは他の共通 EDU です。

サブエージェントのセットは、クライアント･アプリケーション要求を処理するために割り当てられることがあります。サーバーが存在するマシンに複数のプロセッサーがある場合、またはそのマシンがパーティション・データベースの一部である場合、複数のサブエージェントを割り当てることができます。たとえば、対称マルチプロセッシング (SMP) 環境では、複数の SMP サブエージェントが、多くのプロセッサーを活用することができます。

すべてのエージェントおよびサブエージェントは、プール・アルゴリズムを使用して管理されます。これにより、EDU の作成および破棄の数を最小限にとどめることができます。

バッファー・プールは、データベース・サーバー・メモリーのエリアであり、ここで、ユーザー表データ、索引データ、およびカタログ・データが一時的に移動されたり、あるいは変更されたりします。データはディスクからよりもメモリーからの方がずっと速くアクセスできるため、バッファー・プールは、データベースのパフォーマンスのレベルを決定する主要なものとなります。アプリケーションが必要な分より多くのデータがバッファー・プールに存在する場合、ディスクで検索するよりも早くそのデータにアクセスすることができます。

バッファー・プールの構成は、プリフェッチャーおよびページ・クリーナー EDU と共に、データにアクセスする時間、およびアプリケーションに対するデータの準備状態を制御します。

- **プリフェッチャー**は、データをディスクから取り出し、アプリケーションがそのデータを必要とする前にこれをバッファー・プールに移動します。たとえば、データ・プリフェッチャーが存在しなければ、大量のデータ全体をスキャンする必要のあるアプリケーションは、データがディスクからバッファー・プールに移動するのを待機しなければなりません。アプリケーションのエージェントは、非同期読み取り先行要求を共通プリフェッチ・キューに送信します。使用可能になると、プリフェッチャーは大きなブロックを使用してこれらの要求をインプリメントするか、または読み取り入力操作を分散させてディスクからバッファー・プールに要求されたページを移動します。データベース・データのストレージに複数のディスクがあれば、データを複数ディスク間でストライピングすることができます。データ・ストライピングにより、プリフェッチャーは同時に複数のディスクを使用してデータを取り出すことができます。
- **ページ・クリーナー**は、データをバッファー・プールからディスクに戻します。ページ・クリーナーはアプリケーション・エージェントから独立したバックグラウンド EDU です。必要なくなったバッファー・プールからページを検索し、そ

のページをディスクに書き込みます。ページ・クリーナーにより、プリフェッチャーが取り出すページのスペースがバッファー・プール内に確保されます。

独立したプリフェッチャーやページ・クリーナー EDU がない場合には、バッファー・プールとディスク装置との間のデータの読み書きすべてをアプリケーション・エージェントが実行しなければなりません。

関連概念:

- 258 ページの『バッファー・プールへのデータのプリフェッチ』
- 11 ページの『アプリケーション間のデッドロック』
- 13 ページの『データベース・ディレクトリーとファイル』
- 27 ページの『ログ処理』
- 30 ページの『更新処理』
- 31 ページの『クライアント/サーバー処理モデル』
- 37 ページの『メモリー管理』
- 295 ページの『クライアント接続用の接続コンセントレーターの改善』

関連資料:

- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 438 ページの『max_connections - クライアント接続の最大数』

アプリケーション間のデッドロック

複数のアプリケーションがデータベースからのデータを処理している場合には、 2 つ以上のアプリケーションの間でデッドロックが発生する可能性があります。

デッドロックは、1 つのアプリケーションが、別のアプリケーションがデータのロックを解放するのを待機している場合に実行されます。待機しているアプリケーションのそれぞれが、他のアプリケーションで必要なデータをロッキングしています。他のアプリケーションが保持しているデータを解放するのを互いに待機している状態がデッドロックです。 アプリケーションは、 1 つのアプリケーションが保持データのロックを解放するのを永久に待機する可能性があります。

デッドロックについて、次の図に示します。

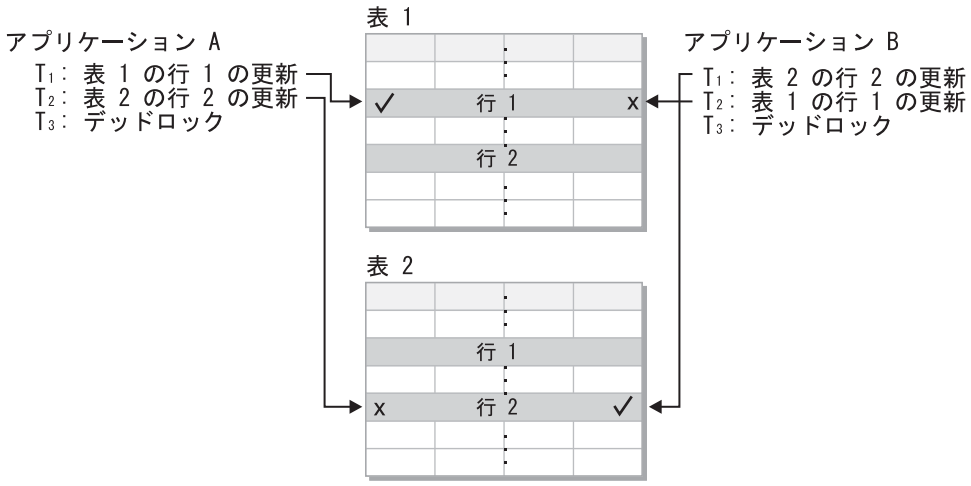


図 2. デッドロック検出機能

他のアプリケーションは、自分の必要なデータを自発的には解放しないので、デッドロックを解除し、アプリケーション処理を続行するためのデッドロック検出機能が必要です。この名前で示すとおり、デッドロック検出機能は、ロックを待機しているエージェントについての情報をモニターします。デッドロック検出機能は、デッドロック状態のアプリケーションの 1 つを無作為に選択し、この「志願した」アプリケーションが保持しているロックを解放します。アプリケーションのロックを解放すると、他の待機しているアプリケーションで必要なデータが使用可能になります。これで、待機しているアプリケーションは、トランザクションを完了するために必要なデータにアクセスできます。

関連概念:

- 55 ページの『ロックとパフォーマンス』
- 9 ページの『DB2 アーキテクチャーおよびプロセスの概要』

ディスク装置の概要

データがどのようにディスク上に保管されているかを理解すると、入出力を調整するのに役立ちます。

ディスク装置のパフォーマンス要因

システムのパフォーマンスには、システムを構成するハードウェアが影響を与える場合があります。ハードウェアがパフォーマンスに与える影響の例として、ディスク装置について考慮しましょう。

ディスク装置の 4 つの性質がパフォーマンスに影響します。

- **ストレージの分割方法**

  限られた量のストレージを、索引とデータ間で、および表スペース間で分割する方法によって、さまざまな状況でそれぞれのストレージがどのように機能するかが大部分決まります。

- **ストレージのむだ**

  ストレージのむだは、それ自体が、そのストレージを使用しているシステムのパフォーマンスに影響を与えることはありませんが、他の場所でパフォーマンスを改善するのに使用できるリソースになります。
- **ディスク入出力の分散**

  複数のディスク記憶装置とコントローラーにディスク入出力要求をバランスよく分散するかどうかは、データベース・マネージャーがディスクから情報を取り出す速度に影響を与えます。
- **使用可能なストレージの不足**

  使用可能なストレージの限界に達すると、全体のパフォーマンスが低下する可能性があります。

関連概念:

- 290 ページの『DMS 装置に関する考慮事項』
- 13 ページの『データベース・ディレクトリーとファイル』
- 16 ページの『SMS 表スペース』
- 17 ページの『DMS 表スペース』
- 20 ページの『標準の表における表および索引の管理』
- 23 ページの『MDC 表のための表および索引管理』

データベース・ディレクトリーとファイル

データベースを作成する際、デフォルト情報を含むデータベースについての情報がディレクトリー階層に保管されます。この階層ディレクトリー構造は、データベース作成 (CREATE DATABASE) コマンドで提供した情報が決定する位置に作成されます。データベースを作成する際にディレクトリー・パスまたはドライブの位置を指定しない場合には、デフォルト位置が使用されます。

データベースを作成する位置を明示的に指定するようにお勧めします。

CREATE DATABASE コマンドで指定するディレクトリーでは、インスタンスの名前を使用したサブディレクトリーが作成されます。このサブディレクトリーにより、同じディレクトリーの異なるインスタンスで作成されたデータベースが同じパスを使用しないことが保証されます。インスタンス名のサブディレクトリーの下に、 NODE0000 と名づけられたサブディレクトリーが作成されます。このサブディレクトリーは、論理パーティション・データベース環境のパーティションを区別します。ノード名ディレクトリーの下に、SQL00001 と名づけられたサブディレクトリーが作成されます。サブディレクトリーの名前は、データベース・トークンを使用し、作成中のデータベースを表します。 SQL00001 には、作成された最初のデータベースに関連したオブジェクトが入りますが、その後のデータベースには、SQL00002 などの後続番号が付けられます。これらのサブディレクトリーは、CREATE DATABASE コマンドで指定したディレクトリーで、このインスタンス内で作成されたデータベースと区別します。

ディレクトリー構造は、次のとおりです。

```
<your_directory>/<your_instance>/NODE0000/SQL00001/
```

データベース・ディレクトリーには、データベース作成 (CREATE DATABASE) コマンドの一部として作成される、次のファイルが組み込まれます。

- ファイル SQLBP.1 および SQLBP.2 には、バッファー・プール情報が入っています。各ファイルには、バックアップを提供するための複製コピーがあります。
- ファイル SQLSPCS.1 および SQLSPCS.2 には、表スペース情報が入っています。各ファイルには、バックアップを提供するための複製コピーがあります。
- SQLDBCON ファイルには、データベース構成情報が入っています。このファイルを編集しないでください。構成パラメーターを変更するには、コントロール・センターまたはコマンド行ステートメント UPDATE DATABASE CONFIGURATION および RESET DATABASE CONFIGURATION のどちらかを使用します。
- DB2RHIST.ASC 履歴ファイルおよびそのバックアップ DB2RHIST.BAK には、バックアップ、リストア、表のロード、表の再編成、表スペースの変更、およびデータベースへの他の変更についての履歴情報が含まれています。

  DB2TSCHNG.HIS ファイルには、ログ・ファイル・レベルでの表スペース変更の履歴が入っています。各ログ・ファイルごとに、DB2TSCHG.HIS には、ログ・ファイルに影響される表スペースを識別するのに役立つ情報が入っています。表スペース・リカバリーは、このファイルからの情報を使用して、表スペース・リカバリー中に処理するログ・ファイルを判別します。テキスト・エディターで、両方の履歴ファイルの内容を調べることができます。
- ログ制御ファイル SQLOGCTL.LFH および SQLOGMIR.LFH にはアクティブ・ログについての情報が入ります。

  リカバリー処理では、このファイルの情報を使用して、リカバリーのために戻るログ内の位置を判別します。 SQLOGDIR サブディレクトリーには、実際のログ・ファイルが入ります。

  注: このログ・サブディレクトリーがご使用のデータに使用されているディスクとは異なるディスクにマップされていることを確認してください。こうすると、ディスクの問題が生じたときに、データとログの両方ではなく、データまたはログのいずれかに範囲を狭めることができます。ログ・ファイルおよびデータベース・コンテナーは、同じディスク・ヘッドの移動で競合することはないため、パフォーマンスにも大きな利点となります。ログ・サブディレクトリーのロケーションを変更するには、 *newlogpath* データベース構成パラメーターを変更します。
- SQLINSLK ファイルは、データベースが必ず 1 つのデータベース・マネージャー・インスタンスでしか使われないようにします。

データベースが作成されると同時に、詳細デッドロック・イベント・モニターも作成されます。詳細デッドロック・イベント・モニター・ファイルは、カタログ・ノードのデータベース・ディレクトリーに保管されています。イベント・モニターが、出力するファイルの最大数に達した場合、イベント・モニターは非活動化され、メッセージが通知ログに書き込まれます。これは、イベント・モニターがディスク・スペースを使用し過ぎるのを防ぎます。不要になった出力ファイルを除去すると、イベント・モニターは次のデータベースの活動化時にアクティブになります。

SMS データベース・ディレクトリーについての追加情報

SQLT* サブディレクトリーには、作動データベースに必要なデフォルトのシステム管理スペース (SMS) 表スペースが含まれます。デフォルトの表スペースは 3 つ作成されます。

- SQLT0000.0 サブディレクトリーには、システム・カタログ表のカタログ表スペースが含まれます。
- SQLT0001.0 サブディレクトリーには、デフォルト TEMPORARY 表スペースが含まれます。
- SQLT0002.0 サブディレクトリーには、デフォルト・ユーザー・データ表スペースが含まれます。

各サブディレクトリーまたはコンテナーには、 SQLTAG.NAM というファイルが作成されています。このファイルは、サブディレクトリーに使用中のマークを付け、後続の表スペース作成で、これらのサブディレクトリーが使用されないようにします。

さらに、SQL*.DAT というファイルが、サブディレクトリーまたはコンテナーに含まれる、各表についての情報を保管します。アスタリスク (*) は、各表を示すユニークな数字の集合で置き換えられます。各 SQL*.DAT ファイルごとに、表タイプ、表の再編成状況、または索引、LOB、または LONG フィールドがその表に存在するかどうかによって、以下のファイルが 1 つ以上存在することがあります。

- SQL*.BKM (MDC 表である場合、ブロック割り振り情報を含む)
- SQL*.LF (LONG VARCHAR または LONG VARGRAPHIC データを含む)
- SQL*.LB (BLOB、CLOB、または DBCLOB データを含む)
- SQL*.LBA (SQL*.LB ファイルについての割り当ておよびフリー・スペース情報を含む)
- SQL*.INX (索引表データを含む)
- SQL*.IN1 (索引表データを含む)
- SQL*.DTR (SQL*.DAT ファイルの再編成についての一時データを含む)
- SQL*.LFR (SQL*.LF ファイルの再編成についての一時データを含む)
- SQL*.RLB (SQL*.LB ファイルの再編成についての一時データを含む)
- SQL*.RBA (SQL*.LBA ファイルの再編成についての一時データを含む)

関連概念:

- 「*管理ガイド: プランニング*」の『SMS 表スペースと DMS 表スペースの比較』
- 290 ページの『DMS 装置に関する考慮事項』
- 16 ページの『SMS 表スペース』
- 17 ページの『DMS 表スペース』
- 18 ページの『DMS 表スペース・アドレス・マップの図』
- 「*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*」の『リカバリー履歴ファイルについて』

関連資料:

- 「コマンド・リファレンス」の『CREATE DATABASE コマンド』

表スペースの概説

次のセクションでは、表スペースについて説明し、 DB2 で使用可能な 2 つのタイプの表スペースについて、その利点と欠点を比較しながら説明します。

SMS 表スペース

システム管理スペース (SMS) 表スペースは、オペレーティング・システム・ファイルのデータを保管します。表スペースのデータは、システム内のすべてのコンテナーにエクステント単位でストライピングされます。エクステント は、データベースに定義される連続ページのグループです。ファイル拡張子は、ファイルに保管されているデータのタイプを示します。データを表スペースのすべてのコンテナーに均一に配布するために、表の開始エクステントは、すべてのコンテナーでラウンドロビン方式で配置されています。そのようなエクステントの分散は、データベースに小さな表がたくさん入っている場合に特に重要です。

SMS 表スペースでは、表のスペースは要求時に割り振られます。割り振られるスペースの量は、 multipage_alloc データベース構成パラメーターの設定によって左右されます。この構成パラメーターが YES に設定されている場合、スペースが必要なときにはエクステント全体が割り振られます。それ以外の場合は、一度に 1 ページのスペースが割り振られます。バージョン 8.2 以前は、構成パラメーターのデフォルト設定が NO であったため、一度に 1 ページしか割り振ることができませんでした。このデフォルトは、db2empfa ツールを使用して変更できました。 db2empfa を実行すると、multipage_alloc データベース構成パラメーターは Yes に設定されます。バージョン 8.2 では、構成パラメーターのデフォルト設定が YES に設定されているので、デフォルトで一度にエクステント全体が割り振られます。

複数ページ・ファイル割り振りは、表のデータおよび索引部分にのみ影響します。したがって、.LF、.LB、および .LBA ファイルが一度に 1 つのエクステントに拡張されることはありません。

1 つのコンテナーのすべてのスペースが表に割り当てられると、他のコンテナーでスペースが残っていても、表スペースはいっぱいであると見なされます。まだコンテナーが 1 つもないパーティションの SMS 表スペースにのみ、コンテナーを追加できます。

注: SMS 表スペースは、ファイル・システムのプリフェッチとキャッシングを利用できます。

関連概念:

- 「*管理ガイド: プランニング*」の『表スペースの設計』
- 「*管理ガイド: プランニング*」の『SMS 表スペースと DMS 表スペースの比較』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『区分内の SMS 表スペースへのコンテナーの追加』

関連資料:

- 501 ページの『multipage_alloc - マルチページ・ファイル割り振り使用可能』

- 「コマンド・リファレンス」の『db2empfa - 複数ページ・ファイル割り振りの使用可能化コマンド』

DMS 表スペース

データベース・マネージャーは、データベース管理スペース (DMS) 表スペースを使用して記憶スペースを制御します。 DMS 表スペースが定義されるときには、表スペースに属するように装置またはファイルのリストが選択されます。これらの装置またはファイルのスペースは、 DB2® データベース・マネージャーにより管理されます。 SMS 表スペースおよびコンテナーと同様、 DMS 表スペースおよびデータベース・マネージャーもエクステント単位のストライピングを使用して、すべてのコンテナー間でデータが均等分散されるようにします。

DMS 表スペースと SMS 表スペースの相違点は、DMS 表スペースでは、必要なときではなく、表スペースが作成される際にスペースが割り当てられることです。

また、この 2 種類の表スペースでは、データの配置が異なることがあります。たとえば、効果的な表スキャンの必要性について考慮しましょう。エクステントのページが物理的に連続していることは重要です。 SMS の場合、オペレーティング・システムのファイル・システムが、各論理ページを物理的に配置する位置を決定します。ファイル・システムの他のアクティビティーのレベル、および配置を判別するのに使用されるアルゴリズムによっては、ページが連続して割り当てられることも、割り当てられないこともあります。ところが、DMS の場合、データベース・マネージャーがディスクとのインターフェースを直接とるため、確実にページが物理的に連続するようにできます。

注: SMS 表スペースと同様、 DMS ファイル・コンテナーもファイル・システムのプリフェッチとキャッシングを利用できます。しかし、DMS 表スペースは利用できません。

このストレージでのページの連続配置については、この一般論が適用できない例外が 1 つあります。 DMS 表スペースを処理する際には、2 つのコンテナー・オプションがあります。ロー・デバイスとファイルです。ファイル・コンテナーを処理する際、データベース・マネージャーは、表スペースの作成時にコンテナー全体を割り当てます。表スペース全体のこの初期割り当ての結果として、物理的な割り当ては通常は連続していますが、ファイル・システムがこの割り当てを実行しているとしても、連続するとは保証されません。ロー・デバイス・コンテナーを処理する場合には、データベース・マネージャーがデバイス全体を制御し、常にエクステントのページを確実に連続配置できます。

SMS 表スペースとは異なり、DMS 表スペースを構成するコンテナーの場合は、容量を互いに均等に近づける必要はありません。ただし、コンテナーの容量は均等、あるいは均等に近いようにすることが勧められています。また、コンテナーで満杯のものがあれば、他のコンテナーで使用可能なフリー・スペースを DMS 表スペースで使用することができます。

DMS 表スペースを処理する場合には、コンテナーを異なるディスクに関連付けることを考慮しなければなりません。これにより、表スペースの容量は大きくなり、並列入出力操作を利用する機能も改善されます。

CREATE TABLESPACE ステートメントは、データベースで新しい表スペースを作成し、この表スペースにコンテナーを割り当て、カタログに表スペース定義と属性を記録します。表スペースの作成時、エクステント・サイズは連続するページの数として定義されます。エクステントは、表スペース内のスペース割り当ての単位です。 1 つの表、または索引などのその他のオブジェクトでは、単一エクステント内のページしか使用できません。表スペースで作成されるすべてのオブジェクトには、論理スペース・アドレス・マップでエクステントが割り当てられます。エクステントの割り当ては、スペース・マップ・ページ (SMP) により管理されます。

論理表スペース・アドレス・マップの先頭のエクステントは、表スペースのヘッダーで、これには内部制御情報が含まれます。 2 番目のエクステントは、表スペースのスペース・マップ・ページ (SMP) の最初のエクステントです。 SMP エクステントは、表スペースで等インターバルに分散されます。それぞれの SMP エクステントは、現行の SMP エクステントから次の SMP エクステントへのエクステントのビットマップに過ぎません。このビットマップは、どの中間エクステントが使用中かをトラッキングするのに使用されます。

SMP に続くエクステントは、表スペースのオブジェクト表です。オブジェクト表は、表スペースにどのユーザー・オブジェクトが存在するか、またどこに最初のエクステント・マップ・ページ (EMP) エクステントが配置されているかをトラッキングする内部表です。各オブジェクトには、それぞれ EMP があり、これは、論理表スペース・アドレス・マップに保管されているオブジェクトの各ページへのマップを提供します。

関連概念:

- 「*管理ガイド: プランニング*」の『表スペースの設計』
- 「*管理ガイド: プランニング*」の『SMS 表スペースと DMS 表スペースの比較』
- 290 ページの『DMS 装置に関する考慮事項』
- 13 ページの『データベース・ディレクトリーとファイル』
- 16 ページの『SMS 表スペース』
- 18 ページの『DMS 表スペース・アドレス・マップの図』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『DMS 表スペースへのコンテナーの追加』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』

DMS 表スペース・アドレス・マップの図

次の図に、DMS 表スペースの論理アドレス・マップを示します。

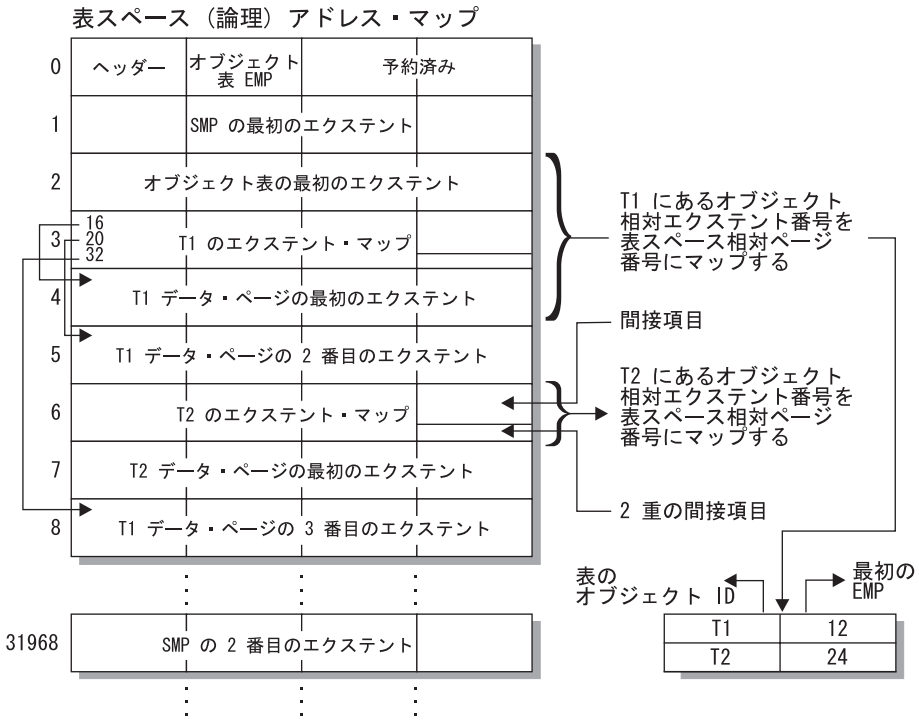


図 3. DMS 表スペース

オブジェクト表は内部リレーショナル表で、オブジェクト ID を表の最初の EMP エクステントのロケーションにマップします。この EMP エクステントは、直接的にも間接的にも、オブジェクトにあるすべてのエクステントをマップします。各 EMP には、項目の配列が含まれています。各項目は、オブジェクト相対エクステント番号を、オブジェクト・エクステントが配置されている表スペース相対ページ番号にマップします。直接 EMP 項目は、オブジェクト相対アドレスをスペース相対アドレスに直接マップします。最初の EMP エクステントにある最後の EMP ページには、間接項目が含まれます。間接 EMP 項目は EMP ページにマップし、続いて EMP ページがオブジェクト・ページにマップします。最初の EMP エクステントにある最後の EMP ページの末尾の 16 個の項目には、 2 重の間接項目が含まれます。

論理表スペース・アドレスからのエクステントは、表スペースに関連付けられているコンテナー全体にラウンドロビン順序でストライピングされます。

関連概念:

- 290 ページの『DMS 装置に関する考慮事項』
- 12 ページの『ディスク装置のパフォーマンス要因』
- 17 ページの『DMS 表スペース』

表と索引

次のセクションでは、標準およびマルチディメンション・クラスタリング (MDC) 表の管理について、またこれらの表での索引について説明します。

標準の表における表および索引の管理

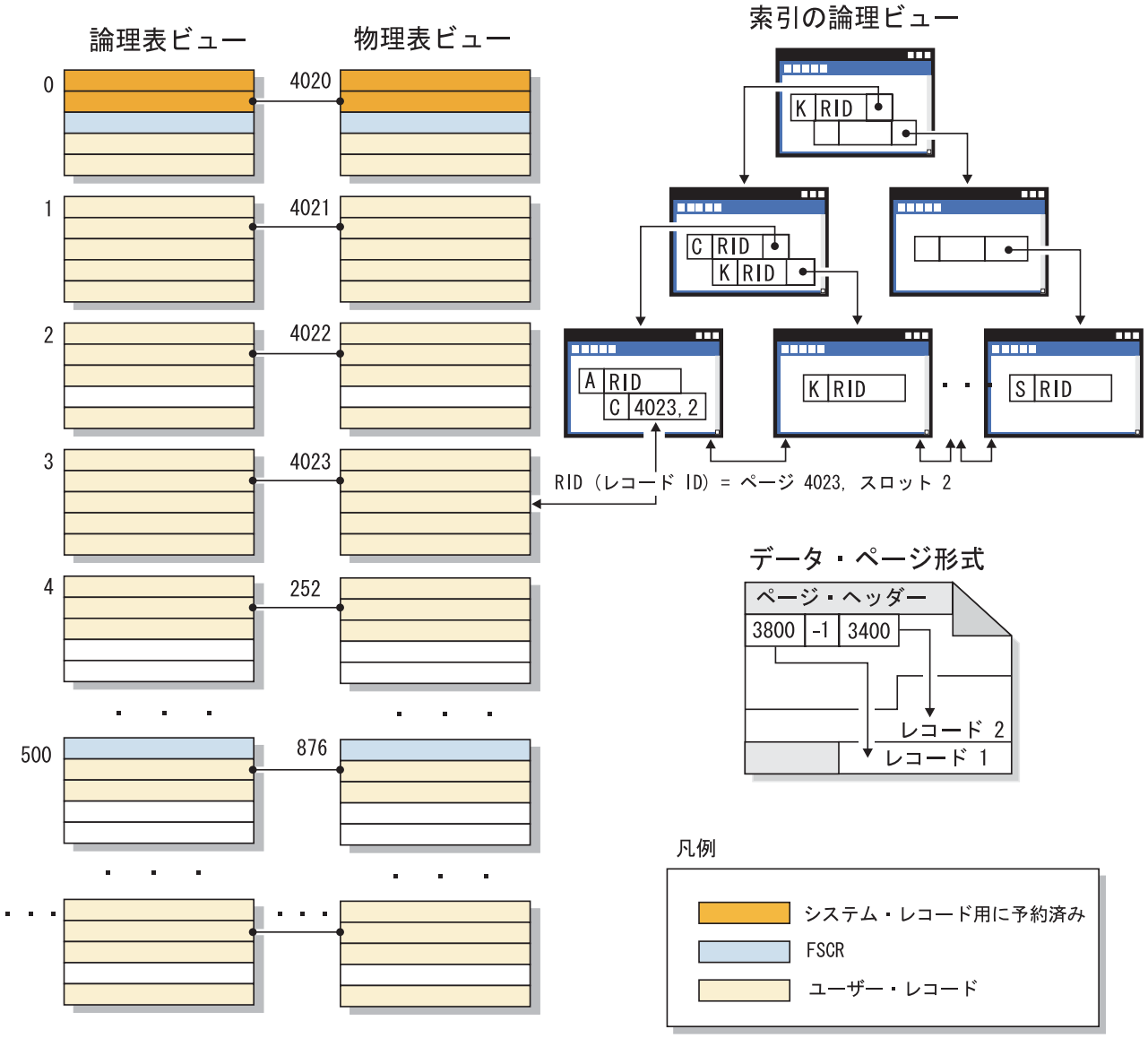


図 4. 標準の表の論理表、レコード、および索引構造

標準の表では、データはデータ・ページのリストとして論理的に編成されます。これらのデータ・ページは、表スペースのエクステント・サイズに基づいて論理的にグループ分けされます。たとえば、エクステント・サイズが 4 の場合、 0 ページから 3 ページが最初のエクステントに入り、 4 ページから 7 ページが 2 番目のエクステントに入るようになります。

それぞれのデータ・ページに入るレコードの数は、データ・ページのサイズやレコードのサイズによって異なります。 1 ページに最大 255 個のレコードが入ります。大半のページには、ユーザー・レコードしか入りません。ただし、いくつかのページには特殊な内部レコードが含まれます。これは表を管理するのに DB2® によって使用されます。たとえば、標準の表の場合に、データ・ページで 500 ページごとにフリー・スペース制御レコード (FSCR) があるとします。これらのレコード

は、FSCR に続く 500 個のデータ・ページ (次の FSCR まで) それぞれに存在する新しいレコードのフリー・スペースの量をマップします。この使用可能なフリー・スペースは、表にレコードを挿入する際に使用されます。

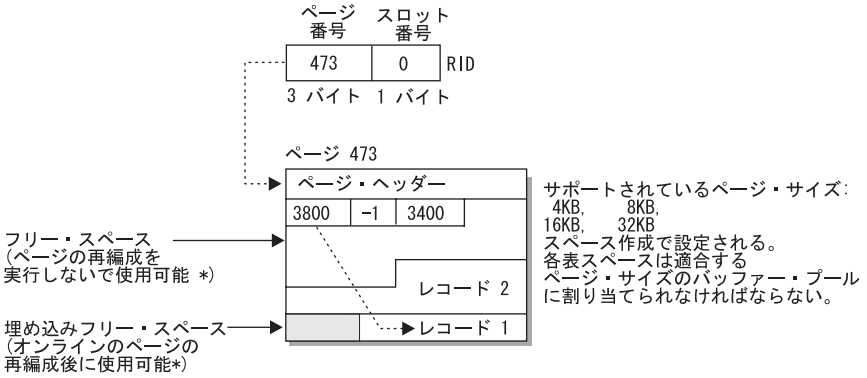
論理的には、索引ページは B ツリーとして編成されています。 B ツリーでは、特定のキー値を持つ表にレコードを効率的に配置できます。索引ページでのエンティティーの数は固定しておらず、キーのサイズによって異なります。 DMS 表スペースの表では、索引ページにあるレコード ID (RID) は、オブジェクト相対ページ番号ではなく表スペース相対ページ番号を使用します。これによって、**索引スキャン**は、マップのためにエクステント・マップ・ページ (EMP) を要求することなく、データ・ページに直接アクセスできるようになります。

各データ・ページのフォーマットは同じです。データ・ページの先頭にはページ・ヘッダーがあります。ページ・ヘッダーの後には、スロット・ディレクトリーがあります。スロット・ディレクトリーの各項目は、ページの異なるレコードに対応します。項目自体は、レコードが開始するデータ・ページへのバイト・オフセットです。マイナス 1 (-1) の項目は、削除されたレコードに対応します。

レコード ID とページ

レコード ID (RID) は、 3 バイトのページ番号の後に 1 バイトのスロット番号が付いたものです。タイプ 2 の索引レコードには、ridFlag という追加のバイトも含まれます。 ridFlag は、このキーがマークされているか、削除されているかなど、索引中のキーの状況についての情報を保管します。索引を使用して RID が識別されると、その RID は正確なデータ・ページおよびそのページのスロット番号を取得するのに使用されます。レコードに割り当てられた RID は、表の再編成まで変わりません。

データ・ページと RID 形式



* 例外: 非コミット DELETE により予約されるスペースは使用できない。

図 5. データ・ページとレコード ID (RID) 形式

表ページが再編成されるとき、レコードが物理的に削除された後でページに残された埋め込みフリー・スペースは、使用可能なフリー・スペースに変換されます。 RID がデータ・ページのレコードの移動に基づいて再定義され、使用可能なフリー・スペースが活用されます。

DB2 では、さまざまなページ・サイズがサポートされています。行に順次アクセスする傾向のあるワークロードには大きいページ・サイズを使用します。たとえば、順次アクセスが意思決定支援アプリケーションに使用される場合や、一時表が集中的に使用される場合などです。アクセスがランダムである傾向のワークロードには、小さいページ・サイズを使用してください。たとえば、ランダム・アクセスは OLTP 環境で使用されます。

標準の表における索引の管理

DB2 索引は、書き込み先行ロギングを使用した効率的かつ並行性の高い索引管理メソッドに基づく、最適化された B ツリー・インプリメンテーションを使用します。

最適化された B ツリー・インプリメンテーションでは、双方向のポインターがリーフ・ページにあるため、単一索引で前方または後方のいずれの方向でもスキャンできます。通常、索引ページは半々に分割されます。例外として、上位キー・ページでは、90/10 の分割が使用されます。つまり、索引キーの上位 10 % は、新しいページに入れられます。このタイプの索引ページの分割は、新しい上位キーを使用して INSERT 要求が頻繁に実行されるワークロードの場合に役立ちます。

バージョン 8.1 以降、DB2 では タイプ 2 索引が使用されています。以前のバージョンの DB2 から移行している場合、索引の再編成や、タイプ 1 の索引をタイプ 2 に変換する他の処置を実行しない限り、タイプ 1 とタイプ 2 の索引の両方が使用中です。索引タイプは、削除されたキーが索引ページから物理的に除去される方法を決定します。

- タイプ 1 の索引の場合、ページの最後の索引キーが除去されると、キー削除中に索引ページからキーが除去され、索引内のページが解放されます。
- タイプ 2 の索引の場合、索引キーは、表に X ロックがある場合のみ、キー削除中にページから除去されます。キーがすぐに除去できない場合、削除のマークが付けられ、後で物理的に除去されます。詳細については、タイプ 2 の索引について説明しているセクションを参照してください。

索引作成時に、MINPCTUSED 文節をゼロより大きい値に設定して、オンライン索引デフラグを使用可能にしてある場合、索引リーフ・ページはオンラインでマージできます。指定された値は、索引リーフ・ページで使用されるスペースの最小パーセンテージのしきい値です。索引ページからキーが除去された後で、ページで使用されているスペースが指定された値以下である場合には、データベース・マネージャーは残りのキーを近隣のページのキーにマージしようとします。空きが十分にある場合には、マージが実行され、リーフ・ページが削除されます。オンライン索引デフラグを使用することによってスペース再利用は改善されるかもしれませんが、MINPCTUSED 値が大きすぎると、マージの試行に要する時間が長くなり、マージが正常に実行される可能性は低くなります。この文節の推奨値は、50 % 以下です。

注: オンライン・デフラグは索引ページからキーを除去する場合のみに発生するので、タイプ 2 の索引では、キーが単に削除のマークを付けられているだけで、ページから物理的に削除されていなければ行われません。

CREATE INDEX ステートメントの INCLUDE 文節を使用すると、指定されたカラムをキー・カラムに加えて索引リーフ・ページに組み込むことができます。これで、索引のみのアクセスで適格である照会の数を増やすことができます。ただし、

同時に索引スペースの要件も増やすことになり、組み込まれた列が頻繁に更新される場合には、索引の保守にかかるコストも増える可能性があります。組み込み列更新の保守にかかるコストは、キー列の更新の場合よりも低いですが、索引に表示されない列の更新の保守にかかるコストよりは高くなります。索引 B ツリーの順序付けは、キー・カラムを使用することによってのみ実行できます。組み込みカラムでは実行できません。

関連概念:

- 「*管理ガイド: プランニング*」の『データベース・オブジェクトのスペース所要量』
- 「*管理ガイド: プランニング*」の『表の表スペースを選択する際の考慮事項』
- 「*管理ガイド: プランニング*」の『マルチディメンション・クラスタリング (MDC) 表の設計』
- 23 ページの『MDC 表のための表および索引管理』
- 「*管理ガイド: プランニング*」の『マルチディメンション・クラスタリング (MDC) 表の作成、配置、および使用』
- 285 ページの『索引のクリーン・アップおよび保守』
- 28 ページの『挿入処理』

MDC 表のための表および索引管理

マルチディメンション・クラスタリング (MDC) 表の表および索引の編成は、標準の表編成と同じ論理構造に基づいています。標準の表と同じく、 MDC 表はデータの行を含むページに編成されて、列に分割され、各ページの行は行 ID (RID) によって識別されます。しかしそれに加えて、MDC 表のページはエクステント・サイズのブロックにグループ化されます。たとえば、エクステント・サイズが 4 の表を示す下の例で、 0 ～ 3 の番号が付けられた最初の 4 ページは表の最初のブロックとなります。 4 ～ 7 の番号が付けられた次のページのセットは、表の 2 番目のブロックとなります。

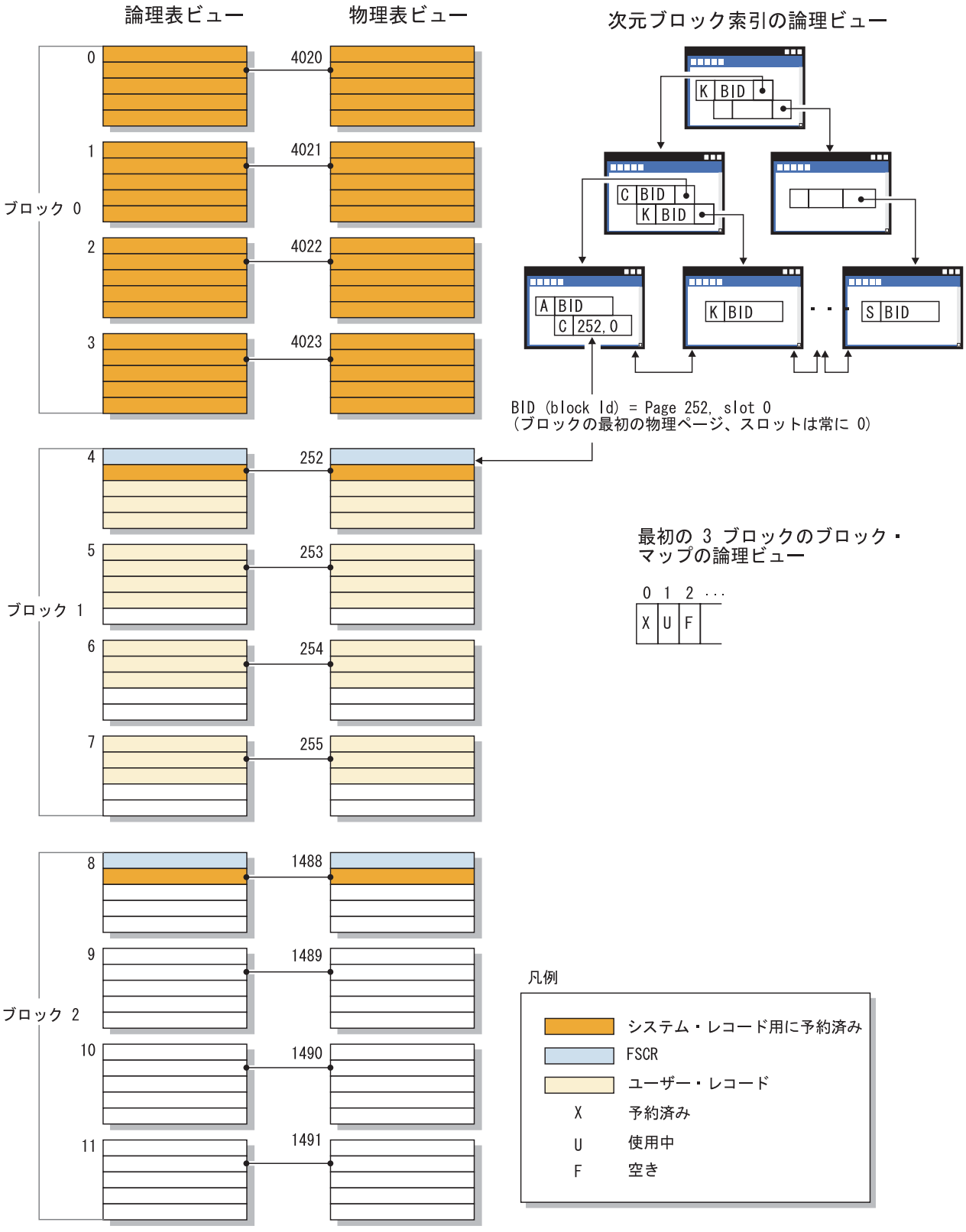


図 6. MDC 表の論理表、レコード、および索引構造

最初のブロックには、DB2® が表を管理するために使用する、フリー・スペース制御レコード (FSCR) を含む特殊な内部レコードが含まれます。続くブロックでは、

最初のページに FSCR が含まれます。 FSCR はブロック内の各ページに存在する新規のレコード用にフリー・スペースをマップします。この使用可能なフリー・スペースは、表にレコードを挿入する際に使用されます。

名前が暗黙に示すように、MDC 表は複数のディメンションのデータをクラスター化します。各ディメンションは、CREATE TABLE ステートメントの ORGANIZE BY DIMENSIONS 文節で指定した列または列のセットによって決まります。 MDC 表を作成するとき、以下の 2 種類の索引が自動的に作成されます。

- 単一のディメンションの各ブロックに対するポインターを含む、ディメンション・ブロック索引。
- すべてのディメンションのキー列を含む、複合ブロック索引。複合ブロック索引を使用して、挿入および更新の際のクラスタリングを保守することができます。

オプティマイザーは、特定の照会に最適のアクセス・プランを判別する際にディメンション・ブロック索引を利用するアクセス・プランを検討します。照会にディメンション値についての述部があるとき、オプティマイザーはディメンション・ブロック索引を使用して、これらの値を含むエクステントを識別し、そこからフェッチします。エクステントはディスク上の物理的に連続したページなので、これによりパフォーマンスが向上して入出力が最少になります。

さらに、データ・アクセス・プランの分析によって特定の RID 索引が照会のパフォーマンスを改善することが示された場合、その索引を作成することができます。

ディメンション・ブロック索引および複合ブロック索引に加えて、 MDC 表は各ブロックの可用性状況を示すビットマップを含むブロック・マップを保守します。以下の属性は、ビットマップ・リスト内にコード化されています。

- X (予約済み): 最初のブロックには表のシステム情報だけが含まれます。
- U (使用中): このブロックはディメンション・ブロック索引と共に使用され、それに関連付けられています。
- L (ロード済み): このブロックは現行のロード操作によってロードされました。
- C (チェック制約): このブロックはロード中の増分制約チェックを指定するためにロード操作によって設定されました。
- T (表のリフレッシュ): このブロックは AST 保守が必要であることを指定するためにロード操作によって設定されました。
- F (フリー): 他の属性が設定されていない場合、ブロックはフリーと見なされます。

各ブロックはブロック・マップ・ファイル内に項目があるので、表が拡大するとファイルも拡大します。この表は別個のオブジェクトとして保管されます。 SMS 表スペース内で、これは新規のファイル・タイプです。 DMS 表スペース内で、これはオブジェクト表に新規のオブジェクト記述子を持ちます。

関連概念:

- 「*管理ガイド: プランニング*」の『データベース・オブジェクトのスペース所要量』
- 「*管理ガイド: プランニング*」の『マルチディメンション・クラスタリング (MDC) 表の設計』

- 「*管理ガイド: プランニング*」の『マルチディメンション・クラスタリング (MDC) 表の作成、配置、および使用』
- 28 ページの『挿入処理』

索引の構造

データベース・マネージャーは、索引の保管に B+ ツリー構造を使用します。 B+ ツリーには、1 つ以上のレベルがあり、それについては次の図に示します。ここで、RID は行 ID を意味します。

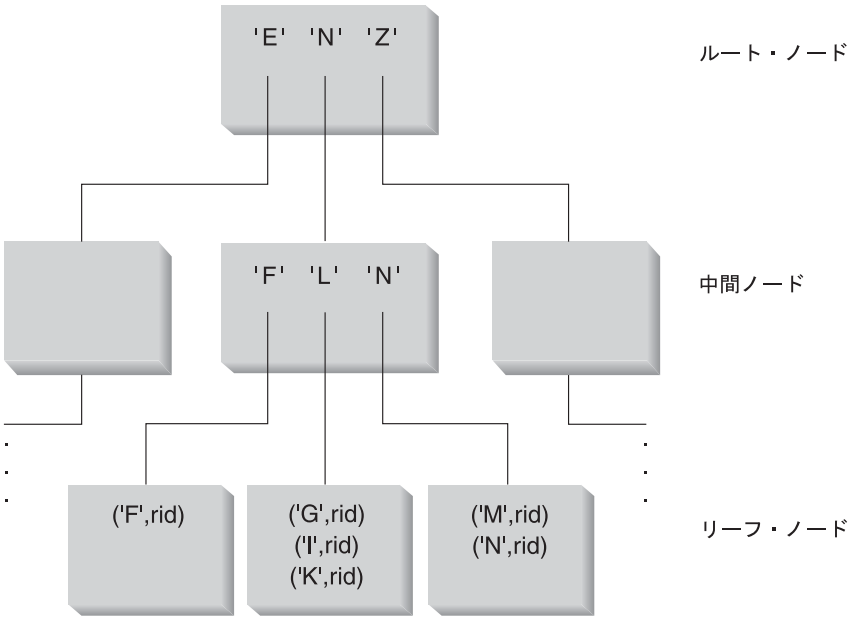


図 7. B+ ツリー構造

最上レベルは*ルート・ノード* と呼ばれます。最下レベルは、*リーフ・ノード* で構成され、ここに索引キー値と、そのキー値のある表の行に対するポインターが保管されます。ルート・ノードとリーフ・ノードの間のレベルは、*中間ノード* と呼ばれます。

特定の索引キー値を検出するとき、索引マネージャーは、ルート・ノードから開始して、その索引ツリーを検索します。ルートには、その次のレベルの各ノードごとに 1 つずつキーが含まれています。それらの各キーの値は、その次のレベルの対応するノードに存在する最大のキー値です。たとえば、図に示すように、索引に 3 つのレベルがある場合に索引キー値を検出するには、索引マネージャーは、ルート・ノードから、目的のキー以上のキー値のうちの最初のものを探索します。ルート・ノード・キーは特定の中間ノードを指しています。索引マネージャーは必要とする索引キーを持つリーフ・ノードが見つかるまで、中間ノードにこの手順を行います。

この図で検索するキーは「I」です。ルート・ノードの中で、「I」以上の最初のキーは「N」です。これはその次レベルの真ん中のノードを指しています。その中間ノードで「I」以上の最初のキーは「L」です。これは「I」の索引キーとそれに対応する行 ID の含まれる特定のリーフ・ノードを指します。その行 ID は基本表内の対応

する行を識別します。リーフ・ノード・レベルには直前のリーフ・ノードへのポインターが入っている場合もあります。こうしたポインターによって、索引マネージャーはどちらの方向にでもリーフ・ノードをスキャンして、範囲内の 1 つの値を見つけた後、値の範囲を検索することができます。いずれの方向にもスキャンを実行する機能は、 ALLOW REVERSE SCANS 文節を使用して索引が作成された場合にのみ使用可能です。

マルチディメンション・クラスタリング (MDC) 表では、表のために指定したクラスタリング・ディメンションごとにブロック索引が自動的に作成されます。もう 1 つの複合ブロック索引も作成されますが、その中には表のディメンションに関係した列ごとのキーの部分が含まれます。これらの索引には RID ではなく、ブロック ID (BID) へのポインターが含まれ、データ・アクセスを改善します。

DB2® バージョン 8.1 以降では、索引はタイプ 1 かタイプ 2 のいずれかです。*タイプ 1 の索引* は古い索引スタイルです。 DB2 の以前のバージョンで作成した索引はこの種類です。

タイプ 2 の索引はタイプ 1 の索引よりいくらか大きく、次のキーのロッキングを最小にする機能を提供します。タイプ 2 の索引のリーフ・ページで RID ごとに保存される 1 バイトの *ridFlag* バイトは、RID を論理的に除去されたとしてマークを付け、後で物理的に除去できるようにします。索引に含まれる可変長列ごとに、1 つの追加のバイトが列の値の実際の長さを保管します。削除されたとマークされていてもまだ索引ページから物理的に除去されていないキーがあるために、タイプ 2 の索引がタイプ 1 の索引より大きくなる場合もあります。 DELETE または UPDATE トランザクションのコミット後に削除済みとマークされたキーをクリーンアップすることができます。

関連概念:

- 277 ページの『索引の利点と欠点』
- 287 ページの『索引の再編成』
- 289 ページの『オンライン索引のデフラグ』

プロセス

次のセクションでは、DB2 プロセスの一般的な説明を示します。

ログ処理

すべてのデータベースが、データベースの変更のレコードが保管されるログ・ファイルを保守します。ロギング・ストラテジーは、以下の 2 つから選択できます。

- 循環ロギング。ログ・レコードがログ・ファイルを満たすと、最初のログ・ファイルの最初のログ・レコードが上書きされます。上書きされたログ・レコードをリカバリーすることはできません。
- 保存ログ・レコード。ログ・レコードがログ・ファイルを満たすと、そのログ・ファイルがアーカイブされます。新しいログ・ファイルが、ログ・レコード用に使用可能になります。ログ・ファイルの保存により、**ロールフォワード・リカバリー**が可能になります。ロールフォワード・リカバリーは、ログに記録されている完了した作業単位 (トランザクション) に基づいてデータベースに変更を適用し

なおします。このロールフォワード・リカバリーは、ログの最後まで実行するか、またはそれより前の特定の時点で終了するかを指定することができます。

ロギング・ストラテジーに関係なく、正規データおよび索引ページになされた変更はすべて、ログ・バッファーに書き込まれます。ログ・バッファーにあるデータは、ロガー・プロセスによってディスクに書き込まれます。次の状況では、照会処理は、ログ・データがディスクに書き込まれるのを待機する必要があります。

- COMMIT で。
- 対応するデータ・ページがディスクに書き込まれる前。これは DB2® が書き込み先行ロギングを使用するためです。書き込み先行ロギングの利点は、COMMIT ステートメントの実行によるトランザクションの完了時、変更されたデータおよび索引ページを必ずしもすべてディスクに書き込まなくてもよいということです。
- 何らかの変更がメタデータに加えられる前。ほとんどの結果は、DDL ステートメントの実行からのものです。
- ログ・バッファーがいっぱいである場合、ログ・バッファーにログ・レコードを書き込むとき。

DB2 は、処理の遅延を最小限に抑えるために、このような方法でログ・データのディスクへの書き込みを管理します。たくさんの短い並行トランザクションが発生する環境では、ほとんどの処理の遅延は、ログ・データがディスクに書き込まれるのを待機しなければならない COMMIT ステートメントが原因です。結果として、ロガー・プロセスは頻繁に少量のログ・データをディスクに書き込むので、ログ入出力のオーバーヘッドによってさらに遅延が生じます。このようなロギングの遅延に対してアプリケーション応答時間のバランスを取るには、 *mincommit* データベースを 1 より大きい値に設定します。この設定により、いくつかのアプリケーションからの COMMIT の遅延はさらに長くなることもありますが、 1 つの操作で書き込まれるログ・データの量は多くなるかもしれません。

ラージ・オブジェクト (LOB) および LONG VARCHAR への変更は、シャドー・ページングによりトラックされます。ログ保存が指定され、LOB カラムが CREATE TABLE ステートメントで NOT LOGGED 文節なしで定義されていないかぎり、 LOB カラムの変更はログに記録されません。 LONG または LOB データ・タイプの割り当てページへの変更は、正規データ・ページと同様にログに記録されます。

関連概念:

- 30 ページの『更新処理』
- 31 ページの『クライアント/サーバー処理モデル』

関連資料:

- 469 ページの『mincommit - グループ化するコミット数』

挿入処理

SQL ステートメントで INSERT を使って新しい情報を表に挿入する場合、INSERT アルゴリズムは最初にフリー・スペース制御レコード (FSCR) を検索して、十分なスペースのあるページを見つけます。ただし、FSCR にフリー・スペースが十分にあると示される場合でも、他のトランザクションの非コミット DELETE

によって予約されているために、このスペースを使用できない可能性もあります。非コミットのフリー・スペースを確実に使用可能にするためには、トランザクションを頻繁に COMMIT する必要があります。

DB2MAXFSCRSEARCH レジストリー変数の設定値は、 INSERT の際に表内で検索される FSCR の数を決定します。このレジストリー変数のデフォルト値は 5 です。指定された数の FSCR の中にスペースが見つからない場合、挿入されるレコードは表の末尾に付加されます。 INSERT の速度を最適化するために、 2 つのエクステントがいっぱいになるまで、後続のレコードは表の末尾に追加されます。この 2 つのエクステントが満杯になったら、次の INSERT では、先回終了した位置から FSCR の検索が再開されます。

注: INSERT の速度を最適化するには、 DB2MAXFSCRSEARCH レジストリー変数を小さい値に設定してください (ただし、表がすぐに大きくなる可能性があります)。スペースの再使用を最適化するには、 DB2MAXFSCRSEARCH を大きい値に設定してください (ただし、 INSERT 速度が遅くなる可能性があります)。

表内のすべての FSCR がこのように検索された後、それ以上の検索は行われずに、挿入されるレコードが付加されます。 FSCR の検索は、(たとえば DELETE 後に) 表のどこかにスペースが作成されるまでは、再実行されません。

このほか、以下の 2 つの INSERT アルゴリズム・オプションがあります。

- APPEND MODE (付加モード)

  このモードでは、新しい行が常に表の末尾に追加されます。 FSCR の検索と保守は行われません。このオプションは ALTER TABLE APPEND ON ステートメントを使用して有効にすることができ、単純に大きくなる表 (たとえばジャーナル) のパフォーマンスを改善します。
- クラスタリング索引を表に定義する

  この場合、データベース・マネージャーは、索引キー値が類似している他のレコードと同じページにレコードを挿入しようとします。このページにスペースがない場合には、周辺のページにレコードを入れるよう試行されます。それも成功しない場合は、上記の FSCR 検索アルゴリズムが使用されます。ただし、ここでは最初に見つかったスペースが使用されるのではなく、最も大きさが異なるスペースが使用されます。このアプローチは、フリー・スペースがより大きいページを選択する傾向があります。この方式により、このキー値を使って行の新しいクラスター域が確立されます。

  表にクラスタリング索引を定義する場合には、表のロードまたは再編成の前に ALTER TABLE... PCTFREE を使用してください。 PCTFREE 文節は、ロードや再編成の後、表のデータ・ページに残しておくべきフリー・スペースのパーセントを指定します。これによって、クラスター索引操作の際に適切なページにフリー・スペースが検出される可能性が高くなります。

関連概念:

- 20 ページの『標準の表における表および索引の管理』
- 30 ページの『更新処理』
- 23 ページの『MDC 表のための表および索引管理』

更新処理

エージェントがページを更新すると、データベース・マネージャーは次のプロトコルを使って、トランザクションが必要とする入出力を最小限にし、リカバリー可能性を保証します。

1. 更新されるページがピンされ、排他ロックでラッチされます。ログ・バッファーにログ・レコードが書き込まれ、この変更を再実行したり、取り消したりする方法が記述されます。このアクションの一部として、ログ・シーケンス番号 (LSN) が取得され、更新されているページのページ・ヘッダーに保管されます。
2. ページが変更されます。
3. ページがアンラッチおよび固定解除されます。

   このページは、「ダーティー」ページと見なされます。ページに加えられた変更が、ディスクに書き出されていないためです。
4. ログ・バッファーが更新されます。

   ログ・バッファーおよび「ダーティー」データ・ページの両方とも、ディスクに強制書き込みされます。

パフォーマンスを向上させるため、これらの入出力は、適当なとき、たとえば、システム負荷が落ち着いたときまで、あるいはリカバリー可能性を保証しなければならなくなったときまで、または限界リカバリー時間まで後回しにされます。厳密に言うと、以下の場合に「ダーティー」ページがディスクに強制書き込みされます。

- 他のエージェントがこれをスワップアウトされるページとして選択した場合。
- 次の結果として、ページでページ・クリーナーが実行される場合。
  - 他のエージェントがこれをスワップアウトされるページとして選択している。
  - *chngpgs_thresh* データベース構成パラメーターのパーセンテージ値を超えた。この値を超えると、非同期ページ・クリーナーが起動し、変更されたページをディスクに書き込みます。

    先行ページ・クリーニングを使用可能である場合、この値は関係がなく、ページ・クリーニングを起動しません。
  - *softmax* データベース構成パラメーターのパーセンテージ値を超えた。この値を超えると、非同期ページ・クリーナーが起動し、変更されたページをディスクに書き込みます。

    先行ページ・クリーニングがデータベースに対して使用可能にされており、ページ・クリーナーの数がデータベースに対して適正に構成されていれば、この値を超えることはありません。
  - ヘイト・リスト上のクリーン・ページの数が極端に低くなっている。ページ・クリーナーは、先行ページ・クリーニング方式のこの条件に対してのみ反応します。
  - ダーティー・ページが現在 LSNGAP 条件に関与している、または関与することが見込まれている。ページ・クリーナーは、先行ページ・クリーニング方式のこの条件に対してのみ反応します。

- NOT LOGGED INITIALLY 文節が呼び出された表の一部としてページが更新されており、 COMMIT ステートメントが発行される場合。 COMMIT ステートメントが実行されると、変更されたページすべてがディスクにフラッシュされてリカバリー可能性が保証されます。

関連概念:

- 27 ページの『ログ処理』
- 31 ページの『クライアント/サーバー処理モデル』

関連資料:

- 471 ページの『softmax - リカバリー範囲およびソフト・チェックポイント・インターバル』
- 428 ページの『chngpgs_thresh - 変更済みページしきい値』

クライアント/サーバー処理モデル

ローカルおよびリモート・アプリケーション・プロセスは、同一のデータベースを処理できます。リモート・アプリケーションとは、データベース・マシンから離れているマシンからデータベース・アクションを開始するアプリケーションのことです。ローカル・アプリケーションは、サーバー・マシンでデータベースに直接アタッチされています。

注: DB2® がクライアント接続を管理する方法は、接続コンセントレーターがオンかオフのどちらかによって異なります。接続コンセントレーターは、*max_connections* データベース・マネージャー構成パラメーターが *max_coordagents* 構成パラメーターより大きい値に設定されている場合、オンになっています。

- 接続コンセントレーターがオフの場合、それぞれのクライアント・アプリケーションには、コーディネーター・エージェント と呼ばれるユニークな EDU が割り当てられます。これは、アプリケーションの処理を調整し、アプリケーションと通信します。
- 接続コンセントレーターがオンになっている場合、各コーディネーター・エージェントは、 たくさんのクライアント接続を一度に 1 つずつ管理することができ、他の作業エージェントがこの作業を実行するように調整することもあります。関連する一時的な接続がたくさんあるインターネット・アプリケーション、または比較的小さいトランザクションがたくさんある同様のインターネット・アプリケーションの場合、接続コンセントレーターは、より多くのクライアント・アプリケーションの接続を許可することにより、パフォーマンスを向上させます。また、各接続ごとのシステム・リソースの使用を削減します。

以下の図にある円は、エンジン・ディスパッチ可能単位 (EDU) を示します。これらは、UNIX® プラットフォームでは「プロセス」、 Windows® NT では「スレッド」と呼ばれます。

アプリケーションとデータベース・マネージャーとの間の通信の手段は、アプリケーションがデータベースで実行しようとしている作業が完了する前に確立しておかなければなりません。

以下の図の A1 では、ローカル・クライアントはまず db2ipccm を介して通信を確立します。 A2 で、db2ipccm は db2agent EDU を処理します。これは、ローカル・クライアントからのアプリケーション要求のコーディネーター・エージェントになります。その後、コーディネーター・エージェントは、A3 でクライアント・アプリケーションに接触して、クライアント・アプリケーションとコーディネーターとの間の共用メモリー通信を確立します。 A4 で、ローカル・クライアントのアプリケーションは、データベースに接続されます。

以下の図の B1 では、リモート・クライアントは、db2tcpcm EDU を使って通信を確立します。他の通信プロトコルが選択されると、適切な通信マネージャーが使用されます。 db2tcpcm EDU は、クライアント・アプリケーションと db2tcpcm の間に TCP/IP 通信を確立します。次に、B2 で db2agent を処理します。これは、アプリケーションのコーディネーター・エージェントになり、接続をこのエージェントに渡します。 B3 で、コーディネーター・エージェントはリモート・クライアント・アプリケーションと接触し、データベースに接続されます。

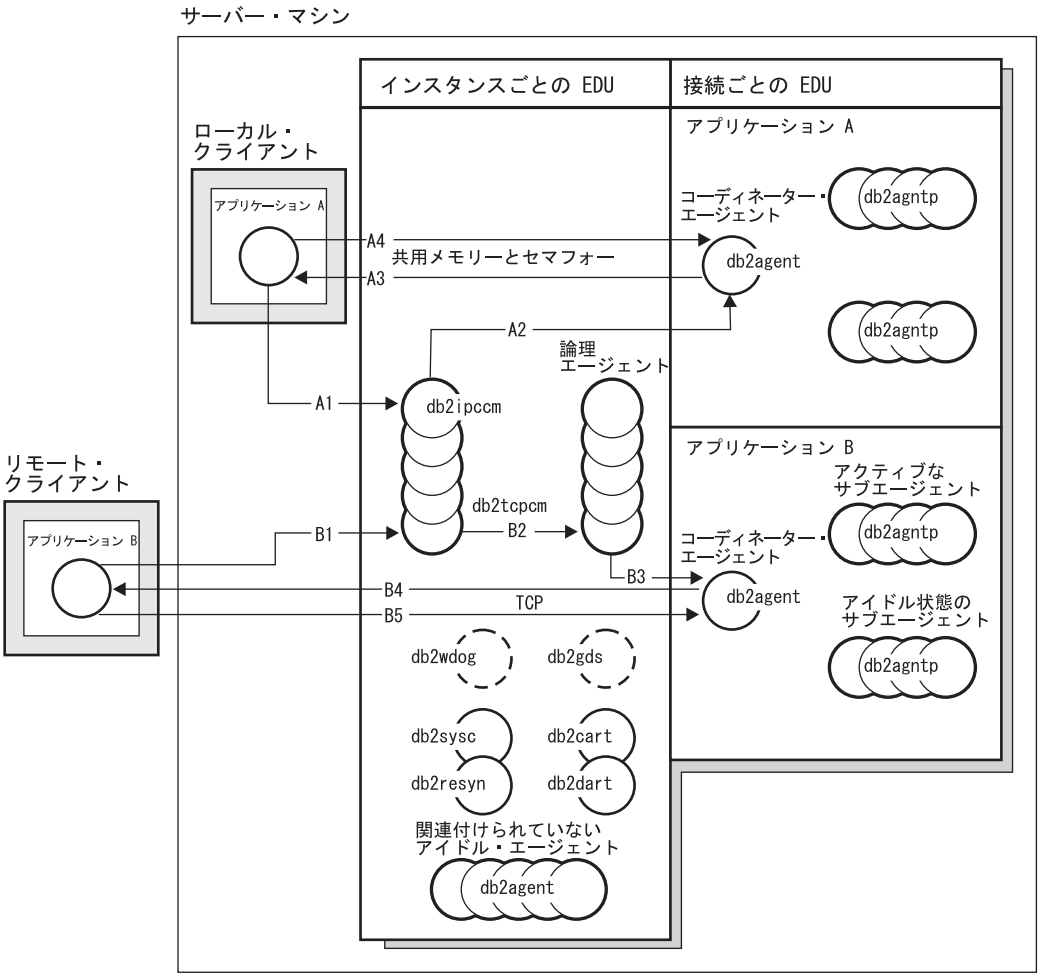


図 8. プロセス・モデルの概要

この図で他に注意することは、以下のとおりです。

- 作業エージェントは、アプリケーション要求を実行します。

- 作業エージェントには、4 つのタイプがあります。アクティブなコーディネーター・エージェント、アクティブなサブエージェント、関連付けられたサブエージェント、およびアイドル・エージェントです。
- 各クライアント接続は、アクティブなコーディネーター・エージェントにリンクされます。
- パーティション・データベース環境、およびパーティション内並列処理環境では、コーディネーター・エージェントは、データベース要求をサブエージェント (db2agntp) に配布します。サブエージェントは、アプリケーションへの要求を実行します。
- エージェント・プール (db2agent) では、アイドル・エージェントおよびプールされたエージェントが新しい作業が来るのを待機します。
- その他の EDU は、クライアント接続、ログ、2 フェーズ COMMIT、バックアップおよびリストア・タスク、およびその他のタスクを管理します。

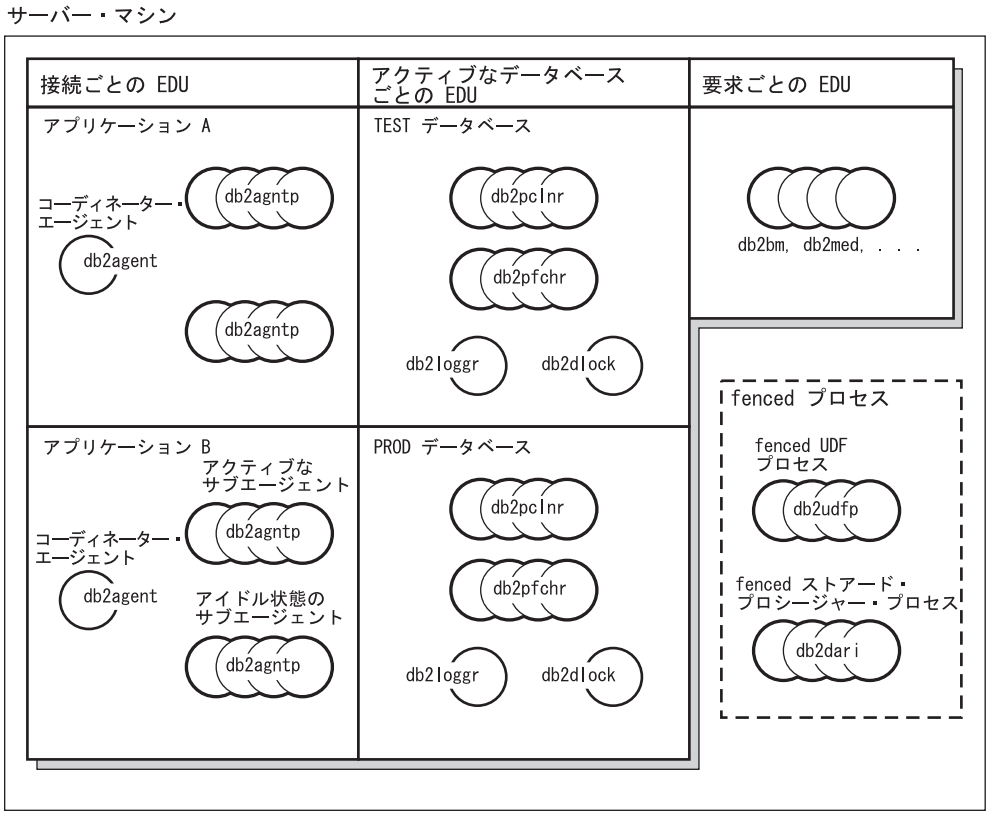


図 9. プロセス・モデル 2

この図は、サーバー・マシン環境の一部である追加のエンジン・ディスパッチ可能単位 (EDU) を示します。アクティブなデータベースには、それぞれプリフェッチャー (db2pfchr) とページ・クリーナー (db2pclnr) の共用プール、そして独自のロガー (db2loggr) およびデッドロック検出機能 (db2dlock) があります。

図には示されていませんが、分離されたユーザー定義関数 (UDF) およびストアード・プロシージャーは、その作成と破棄に関連するコストを最小限に抑えるために管理されます。 *keepfenced* データベース・マネージャー構成パラメーターのデフォ

ルトは「YES」であり、ストアード・プロシージャー・プロセスを、次のストアード・プロシージャー呼び出しで再使用できます。

注: unfenced UDF およびストアード・プロシージャーは、パフォーマンスを向上させるため、エージェントのアドレス・スペースで直接実行します。ただし、エージェントのアドレス・スペースへのアクセスに制限がないため、使用前には厳しくテストする必要があります。

複数パーティション処理モデルは、単一のパーティション処理モデルの論理拡張です。実際、いずれのモードの操作も、共通のコード・ベースでサポートされています。以下の図では、上記の 2 つの図に示されたような単一パーティション処理モデルと、複数パーティション処理モデルとの類似点や相違点を示します。

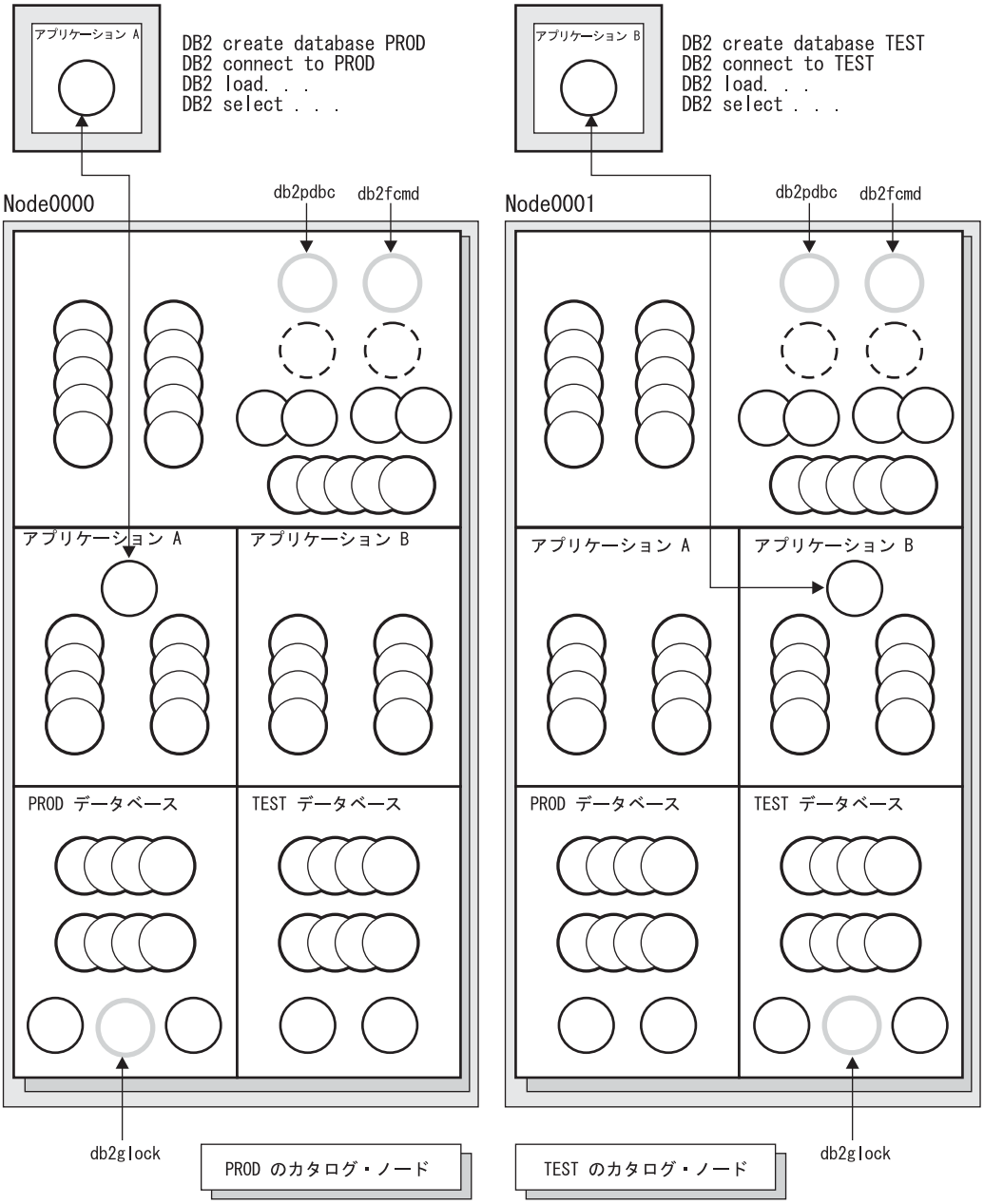


図 10. プロセス・モデルおよび複数のパーティション

エンジン・ディスパッチ可能単位 (EDU) の大半は、単一パーティション処理モデルと、複数パーティション処理モデルで同じです。

複数パーティション (またはノード) 環境では、パーティションの 1 つがカタログ・ノードになります。カタログには、データベース内のオブジェクトに関する情報すべてが保持されています。

上の図で示すとおり、アプリケーション A は Node0000 で PROD データベースを作成するため、 PROD データベースのカタログがこのノードに作成されます。同様に、アプリケーション B が Node0001 に TEST データベースを作成するため、TEST データベースのカタログがこのノードに作成されます。ご使用のシステム環

境のノード間で各データベースのカタログに関連するアクティビティーの均衡を保つために、異なる複数ノード上にデータベースを作成することもできます。

ここには、インスタンスに関連付けられる追加の EDU (db2pdbc および db2fcmd) があり、これらは、複数パーティション・データベース環境の各ノードに見つかります。これらの EDU は、データベース・パーティション間の要求を調整し、高速コミュニケーション・マネージャー (FCM) を使用可能にするのに必要です。

さらに、データベースのカタログ・ノードに関連した追加の EDU (db2glock) があります。この EDU は、アクティブなデータベースがあるノード間のグローバル・デッドロックを制御します。

アプリケーションからの各 CONNECT は、接続を処理するためにコーディネーター・エージェントと関連する接続によって示されます。*コーディネーター・エージェント* は、アプリケーションと通信し、要求を受信し応答を送信するエージェントです。コーディネーター・エージェント自体で要求を満たすことも、複数のサブエージェントを調整して要求上で作業するようにすることもできます。コーディネーター・エージェントが存在するパーティションは、そのアプリケーションの*コーディネーター・ノード* と呼ばれます。コーディネーター・ノードは、SET CLIENT CONNECT_NODE コマンドでも設定できます。

アプリケーションからのデータベース要求のパーツは、コーディネーター・プログラム・ノードにより他のパーティションのサブエージェントに送られます。次いで、他のパーティションからの結果すべてがコーディネーター・プログラム・ノードで統合されてから、アプリケーションに戻されます。

CREATE DATABASE コマンドが出されたデータベース・パーティションは、データベースの「カタログ・ノード」と呼ばれます。カタログ表は、このデータベース・パーティションに保管されます。通常、すべてのユーザー表は一連のノード間でパーティション化されます。

注: 複数のパーティションを、同じマシンで稼働するように構成できます。これは、「複数論理パーティション」、あるいは「複数論理ノード」構成と呼ばれます。このような構成は、巨大なメイン・メモリーのある大きい対称マルチプロセッサー (SMP) マシンで大変役立ちます。この環境では、パーティション間の通信は、共用メモリーおよびセマフォーを使用するように最適化されます。

関連概念:

- 9 ページの『DB2 アーキテクチャーおよびプロセスの概要』
- 27 ページの『ログ処理』
- 30 ページの『更新処理』
- 37 ページの『メモリー管理』
- 295 ページの『クライアント接続用の接続コンセントレーターの改善』

メモリー管理

パフォーマンス・チューニング・タスクとして最初に行うべきことは、データベース内のエリア間で、使用可能メモリーを分割する方法を決定することです。このセクションで説明される、キー構成パラメーターを設定することによって、このメモリーの分割を調整します。

パーティションにあるエンジン・ディスパッチ可能単位 (EDU) はすべて、インスタンス共用メモリーにアタッチされています。データベース内で作業するすべての EDU は、そのデータベースのデータベース共用メモリーにアタッチされています。特定のアプリケーションに代わって作業するすべての EDU は、そのアプリケーションのアプリケーション共用メモリー領域にアタッチされています。このタイプの共用メモリーは、パーティション内またはパーティション間並列処理が使用可能になっている場合にのみ割り当てられます。これで、各 EDU には専用メモリーがあります。

データベースが開始されると、インスタンス共用メモリー (データベース・マネージャー共用メモリーともいう) が割り当てられます。他のすべてのメモリーは、インスタンス共用メモリーからアタッチされるか、または割り振られます。高速コミュニケーション・マネージャー (FCM) が使用される場合には、このメモリーから取り出されたバッファーが使用されます。 FCM は、特定のデータベース環境におけるデータベース・サーバー間およびデータベース・サーバー内での内部通信 (主にメッセージ) に使用されます。最初のアプリケーションがデータベースに接続すると、データベース共用、アプリケーション共用、およびエージェント専用メモリー・エリアが割り当てられます。インスタンス共用メモリーは、*instance_memory* 構成パラメーターによって制御できます。デフォルトでは、このパラメーターは `automatic` に設定されているので、 DB2® はインスタンスに割り振られるメモリーの量を計算できます。

データベースが最初に活動化または接続されると、データベース共用メモリー (データベース・グローバル・メモリーともいう) が割り当てられます。このメモリーは、このデータベースに接続するすべてのアプリケーションが使用できます。データベース共用メモリーは、*database_memory* 構成パラメーターによって制御できます。デフォルトでは、このパラメーターは *automatic* に設定されているので、 DB2 はデータベースに割り振られるメモリーの量を計算できます。

データベース共用メモリーには、以下のものを含む数多くの異なるメモリー・エリアがあります。

- バッファー・プール
- ロック・リスト
- データベース・ヒープ - これには、ログ・バッファーが含まれます。
- ユーティリティー・ヒープ
- パッケージ・キャッシュ
- カタログ・キャッシュ

注: メモリーは、データベースの実行中、異なるエリア間で、割り振り、解放、および交換することができます。たとえば、カタログ・キャッシュを減少させてから、その分だけ指定されたバッファー・プールを増加させることができま

す。しかし、構成パラメーターを動的に変更する前に、そのデータベースに接続されていなければなりません。上記のリストのメモリー・エリアは動的に変更できますが、ロック・リスト・メモリー・エリアを動的に増やすことのみ可能で減らすことはできません。

データベース・マネージャー構成パラメーター *numdb* は、並行してアクティブ化できるローカル・データベースの数を指定します。 *numdb* パラメーターの値は、割り当てられるメモリーの合計量に影響を与えます。

アプリケーションがパーティション・データベース環境でのみ、または内部並列処理が可能な非パーティション・データベースで、または接続コンセントレーターが使用可能な場合にデータベースに接続すると、アプリケーション共用メモリー (アプリケーション・グローバル・メモリーともいう) が割り当てられます。このメモリーは、データベースに接続されるクライアントが要求する作業を実行するエージェントが使用します。

データベース・マネージャー構成パラメーター *max_connections* は、データベースに接続可能なアプリケーションの数の上限を設定します。データベースにアタッチする各アプリケーションは一部のメモリーの割り振りに関係するので、並行アプリケーションの数を多くすると、使用されるメモリーも増えます。

アプリケーションの最大数は、ある程度、データベース・マネージャー構成パラメーター *maxagents*、パーティション環境の場合は *max_coordagents* によっても制御されています。 *maxagents* パラメーターは、パーティションにあるデータベース・マネージャー・エージェントの合計数の上限を設定します。これらのデータベース・マネージャー・エージェントには、アクティブなコーディネーター・エージェント、サブエージェント、非アクティブなエージェント、およびアイドル・エージェントが含まれます。

エージェント専用メモリーは、エージェントの作成時に、このエージェントに対して割り当てられます。エージェント専用メモリーには、その特定のエージェントによってのみ使用される割り当てメモリー (ソート・ヒープやアプリケーション・ヒープなど) が含まれます。

共用メモリーには、いくつか特別なタイプがあります。

- エージェント/ローカル・アプリケーション共用メモリー。このメモリーは、エージェントとそのクライアント・アプリケーションとの間の SQL 要求および応答通信に使用されます。
- UDF/エージェント共用メモリー。このメモリーは、fenced UDF またはストアード・プロシージャーを実行するエージェントによりアタッチされます。これは、通信エリアとして使用されます。
- 拡張記憶。拡張バッファー・プールとして使用される、通常は大変大きい (4 GB を超える) 共用メモリーの領域。エージェント/プリフェッチャー/ページ・クリーナーは永続的にはこれにアタッチされないものの、必要に応じて個々のセグメントにアタッチされます。

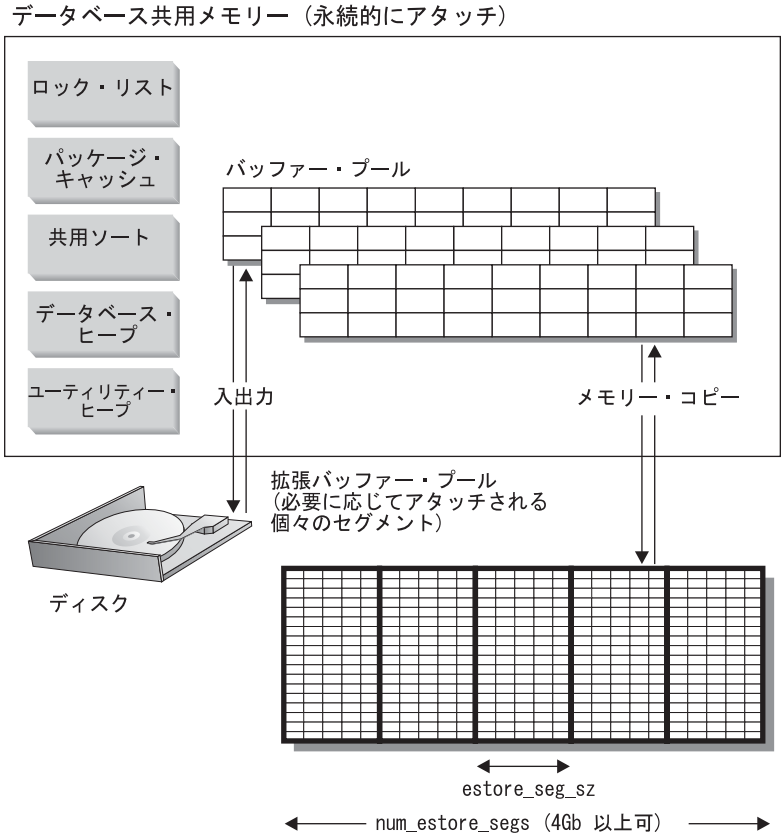


図 11. バッファー・プールが拡張記憶を使用する方法

拡張記憶は、メイン・バッファー・プールの拡張検索バッファーとして機能します。これは 4 GB よりもずっと大きいものです。大量のメイン・メモリーがある 32 ビットのコンピューターの場合、検索バッファーは、そのようなメモリー・パフォーマンスの向上を活用できます。拡張記憶キャッシュは、メモリー・セグメントとして定義されます。64 ビットのコンピューターの場合、すべての使用可能メモリーにアクセスするためにそのような方法は必要ありません。

ただし、実アドレス可能メモリーの一部を拡張記憶域キャッシュとして使用しようとする場合には、ジャーナル・ファイルシステム・キャッシュまたはプロセス専用アドレス・スペースなどの他の目的ではこのメモリーをマシン上で使用できなくなることに注意してください。拡張記憶キャッシュに追加の実アドレス可能メモリーを割り当てると、システム・ページングが増加する可能性があります。

以下に示すデータベース構成パラメーターは、拡張記憶で使用可能なメモリーの量とサイズに影響を与えます。

- *num_estore_segs* は、拡張記憶メモリー・セグメントの数を定義します。
- *estore_seg_sz* は、各拡張メモリー・セグメントのサイズを定義します。

各表スペースには、バッファー・プールが割り当てられます。拡張記憶キャッシュは、常に 1 つまたは複数の特定のバッファー・プールに関連付けておく必要があります。拡張記憶キャッシュのページ・サイズは、関連付けられたバッファー・プールのページ・サイズに適合しなければなりません。

関連概念:

- 237 ページの『メモリー使用の編成』
- 240 ページの『データベース・マネージャー共用メモリー』
- 243 ページの『グローバル・メモリーとその制御パラメーター』
- 248 ページの『バッファー・プール管理』
- 250 ページの『32 ビット・プラットフォームにおける拡張メモリー内の 2 次バッファー・プール』
- 245 ページの『メモリー使用に影響を与えるパラメーターのチューニングのガイドライン』
- 295 ページの『クライアント接続用の接続コンセントレーターの改善』

関連資料:

- 431 ページの『estore_seg_sz - 拡張ストレージ・メモリー・セグメント・サイズ』
- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 432 ページの『num_estore_segs - 拡張ストレージ・メモリー・セグメント数』
- 441 ページの『maxagents - エージェントの最大数』
- 537 ページの『numdb - ホストおよび iSeries データベースを含めた並行アクティブ・データベースの最大数』
- 438 ページの『max_connections - クライアント接続の最大数』
- 420 ページの『instance_memory - インスタンス・メモリー』
- 388 ページの『database_memory - データベース共用メモリー・サイズ』

第 2 部 アプリケーション・パフォーマンスのチューニング

第 3 章 アプリケーションについての考慮事項

アプリケーション実行時のパフォーマンスには、いくつもの要因が影響を与えていると考えられます。この章では、アプリケーションを設計してコーディングする場合、および後でそのパフォーマンスを調整する場合に考慮すべき要因のいくつかを説明します。

並行性制御と分離レベル

次のセクションでは、異なる分離レベルが並行性に与える影響について説明しています。

並行性の問題

多数のユーザーがリレーショナル・データベース内のデータにアクセスしたり、変更したりするため、データベース・マネージャーは、ユーザーがこれらの変更を行い、かつデータ保全性が保持されていることを確認できるようにしなければなりません。*並行性* とは、複数の対話式ユーザーまたはアプリケーション・プログラムが同時にリソースを共用することです。データベース・マネージャーはこのアクセスを制御し、次のような望ましくない結果を防ぎます。

- **更新の消失**。 2 つのアプリケーション A および B が、両方ともデータベースから同じ行を読み取り、読み取ったデータに基づいてその 1 つの列の新しい値をそれぞれが計算することがあるかもしれません。 A がその行を新しい値で更新し、次に B もその行を更新すると、 A によって行われた更新が失われてしまいます。
- **コミットしていないデータへのアクセス**。アプリケーション A がデータベース内の値を更新し、それがコミットされる前に、アプリケーション B がその値を読み取ることがあるかもしれません。この場合、後になって A による値がコミットされずに取り消されるなら、 B が実行する計算は、この非コミットの (そしておそらく無効な) データに基づくものとなってしまいます。
- **反復不能読み取り**。アプリケーションによっては、次のような順序でイベントが生じるものがあります。つまり、まずアプリケーション A がデータベースからデータを読み取り、次いで他の SQL 要求を処理します。その間、アプリケーション B はその行を修正または削除し、その変更をコミットします。その後、アプリケーション A が元の行を再び読み取ろうとすると、修正された行を受け取り、元の行はすでに削除されていることがわかります。
- **幻像読み取り現象**。幻像読み取り現象は、以下の場合に生じます。
  1. アプリケーションが照会を実行し、一定の検索基準に従って行のセットを読み取る。
  2. 別のアプリケーションが、新しいデータを挿入するか、現在のアプリケーションの照会を満足する既存データを更新します。
  3. 最初のアプリケーションは (同じ作業単位内で) ステップ 1 から照会を繰り返す。

最初に照会を実行したとき (ステップ 1) には戻されなかった追加の行 (「幻像読み取り行」) が結果表の一部として戻されます。

注: 宣言済み一時表は、これらを宣言したアプリケーションでのみ使用できるため、並行性の問題はありません。このタイプの表は、アプリケーションがこれを宣言してから、これを完了または切断するまでのあいだのみ存在します。

フェデレーテッド・データベース・システム内での並行性の制御

フェデレーテッド・データベース・システムでは、アプリケーションやユーザーが 1 つのステートメント内で 2 つ以上のデータベース管理システム (DBMS) またはデータベースを参照する SQL ステートメントをサブミットできます。データ・ソース (DBMS およびデータから成る) を参照するには、 DB2® はニックネーム を使用します。ニックネームは、その他のデータベース・マネージャー内でのオブジェクトの別名です。フェデレーテッド・システムでは、 DB2 は、要求されたデータのホストとなるデータベース・マネージャーの並行性制御プロトコルに依存しています。

DB2 フェデレーテッド・システムは、データベース・オブジェクトの*位置透過性*を提供します。たとえば、表およびビューについての情報が移動する場合に、ニックネームによるその情報への参照を、その情報を要求するアプリケーションに変更を加えることなく更新することができます。アプリケーションがニックネームによってデータにアクセスすると、 DB2 はデータ・ソース・データベース・マネージャーのデータの並行性制御プロトコルにより、分離レベルを保証します。 DB2 は要求されたデータ・ソースでの分離レベルを論理的に同等のものと一致させようと試みますが、結果はデータ・ソースの機能によって異なることがあります。

関連概念:

- 44 ページの『分離レベルのパフォーマンスへの影響』

関連タスク:

- 48 ページの『分離レベルの指定』

関連資料:

- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』

分離レベルのパフォーマンスへの影響

分離レベル は、データがアクセスされている間、データが他のプロセスからどのようにロックされ分離されるかを決めるものです。分離レベルは、作業単位の持続期間中で有効です。 WITH HOLD 文節を使用して DECLARE CURSOR ステートメントによって宣言されたカーソルを使用するアプリケーションは、 OPEN CURSOR が実行された作業単位の持続期間中、選択した分離レベルを保ちます。 DB2® でサポートする分離レベルは、次のとおりです。

- 反復可能読み取り
- 読み取り固定
- カーソル固定
- 非コミット読み取り

注: 一部のホスト・データベース・サーバーはコミットなし 分離レベルをサポートします。その他のデータベースでは、この分離レベルはコミットされない読み取り分離レベルと同様に動作します。

これ以降では、それぞれの分離レベルの詳細について、パフォーマンスへの影響の大きい順に説明されています。ただし、データにアクセスしたりデータを変更したりする場合には、影響が小さいほど注意が必要になります。

反復可能読み取り

反復可能読み取り (RR) では、ある作業単位の中でアプリケーションが参照する行すべてにロックがかけられます。 反復可能読み取りを使用した場合、カーソルがオープンされたのと同じ作業単位の中でアプリケーションが同じ SELECT ステートメントを 2 回発行しても、それぞれ同じ結果になります。反復可能読み取りを使用すると、更新が失われている場合、コミットされていないデータおよび幻像読み取り行へのアクセスはできなくなります。

反復可能読み取りアプリケーションは、その作業単位が完了するまでに、必要な回数だけ行の検索および操作を行えます。 しかしそれ以外のアプリケーションは、その作業単位が完了するまで、結果表に影響を与える可能性のある行を更新、削除、または挿入することができなくなります。反復可能読み取りアプリケーションでは、他のアプリケーションの非コミット変更を表示することはできません。

反復可能読み取りを使用すると、検索中の行だけでなく、参照される行すべてにロックがかかります。 適正にロッキングがかけられることによって、最初の照会で参照した後で再び照会を実行したときに、その間に別のアプリケーションが行を挿入または追加するということはできなくなります。これにより、幻像読み取り行が発生しなくなります。たとえば、10 000 個の行をスキャンしてそれらに述部を適用する場合、たとえ 10 行しか適格でなくても、それら 10 000 個の行すべてにロックがかけられます。

注: 反復可能読み取りの分離レベルでは、戻されるデータすべてをアプリケーションが見る までは、一時表や行ブロッキングが使用されている場合であっても、それらの行はすべて未変更のままになります。

反復可能読み取りでは、相当数のロックを獲得し保持するため、それらのロックが、 *locklist* と *maxlocks* の構成パラメーターの結果として使用可能なロックの数を超えることがあります。ロック・エスカレーションが非常に発生しやすいと判断した場合、ロック・エスカレーションを避けるために、オプティマイザーは索引のスキャンのために単一表レベル・ロックを直接獲得することがあります。これは、データベース・マネージャーが LOCK TABLE ステートメントを発行したかのように機能します。表レベルのロックをかけたくない場合は、トランザクションに十分な数のロックを使用できるようにするか、または読み取り固定分離レベルを使用します。

読み取り固定

読み取り固定 (RS) では、ある作業単位においてアプリケーションが検索する行にロックをかけます。 これにより、ある作業単位で適格とされて読み取られた行は、その作業単位が完了するまで他のアプリケーションによって変更されないようになり、また他のアプリケーションのプロセスによって変更されたすべての行は、その

プロセスによって変更がコミットされるまで読み取られないようになります。つまり、「反復不能読み取り」の処理は**あり得なくなります**。

反復可能読み取りとは異なり、読み取り固定では、アプリケーションが同じ照会を 2 回以上出すと、新たな*幻像読み取り* 行 (*幻像読み取り現象*) が生じる可能性があります。前述の 10 000 個の行をスキャンする例を考えると、読み取り固定を使う場合は適格な行だけがロックされます。したがって、読み取り固定を使用すると、10 行だけが検索され、ロックはそれら 10 行だけにしかかけられません。これとは対照的に反復可能読み取りを使用すると、この例の場合は、ロックが 10 000 行すべてにかけられます。保持することができるロックは、共用、次回共用、更新、または排他ロックです。

注: 読み取り固定分離レベルを使う場合、一時表や行ブロッキングを使用する場合でも、返されるデータすべてをアプリケーションが見る までは、それらのデータはすべて未変更になっています。

読み取り固定分離レベルの目的は、データの表示を一定にするとともに、高度の並行性を提供することにあります。 この目的を達成するため、オプティマイザーは、ロック・エスカレーションが発生するまで表レベル・ロックがかけられないようにします。

次のことすべてを行うアプリケーションでは、読み取り固定分離レベルが最適です。
- 並行環境で操作する
- 作業単位の間、適格となる行を一定にしておく必要がある
- 作業単位において同じ照会を 2 回以上発行しない、または同じ作業単位において同じ照会を 2 回以上発行するときに、同じ応答を入手することを要求しない。

カーソル固定

カーソル固定 (CS) は、アプリケーションのトランザクションがアクセスする行にカーソルがある間、その行をロックします。このロックは、次の行が取り出されるか、またはトランザクションが終了する時まで有効です。しかし、行の中の何らかのデータが変更された場合、変更がデータベースにコミットされるまで、ロックが保持されていなければなりません。

カーソル固定アプリケーションが検索した行に更新可能なカーソルがある間、他のアプリケーションはその行を更新したり削除したりできません。 カーソル固定のアプリケーションでは、他のアプリケーションによる非コミット変更を見ることができません。

前述の 10 000 行をスキャンする例を考えてみると、カーソル固定を使う場合は、現在カーソルが位置している行だけにロックがかかります。 ロックは、その行から移動すると (その行を更新しなければ) 解除されます。

カーソル固定を使うと、反復不能読み取りと幻像読み取り現象は両方とも発生する可能性があります。 カーソル固定はデフォルトの分離レベルですが、コミットされた行だけを他のアプリケーションから見る間は並行性を最大にしたいという場合に、ぜひ使用するようにしてください。

非コミット読み取り

非コミット読み取り (UR) では、アプリケーションが他のトランザクションの非コミットの変更にアクセスできます。アプリケーションは、他のアプリケーションが表をドロップまたは変更しようとするのでない限り、読み取り中の行について他のアプリケーションに対するロックをかけません。非コミット読み取りの動作は、読み取り専用カーソルと更新可能カーソルとで違います。

読み取り専用カーソルは、他のトランザクションのほとんどの非コミットの変更にアクセスすることができます。しかし、トランザクションの処理中に、他のトランザクションによって作成またはドロップされている表、ビュー、および索引にアクセスすることはできません。他のトランザクションによるその他の変更は、コミットまたはロールバックされる前に読み取ることができます。

注: 非コミット読み取り分離レベルで更新可能な操作を行っているカーソルは、カーソル固定分離レベルの場合と同じ働きをします。

分離レベル UR を使用してプログラムを実行するとき、アプリケーションは分離レベル CS を使用できます。これは、アプリケーション・プログラムで使用されるカーソルが未確定であるために起こります。 BLOCKING オプションにより、未確定カーソルを分離レベル CS にエスカレーションすることができます。 BLOCKING オプションのデフォルトは、UNAMBIG です。これは、未確定カーソルが更新可能として扱われ、分離レベルが CS にエスカレーションされることを意味します。このエスカレーションを防ぐには、以下の 2 つの選択肢があります。

- アプリケーション・プログラム内のカーソルが未確定とならないように変更します。 SELECT ステートメントを変更して、FOR READ ONLY 文節を組み込みます。
- アプリケーション・プログラム内でカーソルを未確定のままにし、 BLOCKING ALL オプションを使ってプログラムをプリコンパイルまたはバインドします。これにより、プログラム実行時に未確定カーソルはすべて読み取り専用として扱われます。

10 000 行をスキャンする反復可能読み取りの例の場合と同様に、非コミット読み取りを使用すると、行ロックは獲得されません。

非コミット読み取りを使用すると、反復不能読み取り動作と幻像読み取り現象の両方とも発生する可能性があります。非コミット読み取り分離レベルは、読み取り専用表での照会に対して、または SELECT ステートメントのみを実行しており、かつ他のアプリケーションから非コミット・データを見るかどうかが問題にはならない場合に、最もよく用いられています。

分離レベルのまとめ

以下の表には、さまざまな分離レベルの望ましくない影響が要約されています。

表 1. 分離レベルのまとめ

分離レベル	コミットしていないデータへのアクセス	反復不能読み取り	幻像読み取り現象
反復可能読み取り (RR)	不可能	不可能	不可能
読み取り固定 (RS)	不可能	不可能	可能

表 1. 分離レベルのまとめ (続き)

分離レベル	コミットしていないデータへのアクセス	反復不能読み取り	幻像読み取り現象
カーソル固定 (CS)	不可能	可能	可能
非コミット読み取り (UR)	可能	可能	可能

以下の表には、アプリケーションの初期分離レベルを決定するのに役立つ簡単な発見的手法が示されています。この表はあくまで参考用です。さまざまなレベルの要因について述べたこれまでの説明に従えば、他の分離レベルの方が適切である場合もあります。

表 2. 分離レベルを選択する指針

アプリケーションのタイプ	高度のデータ安定度が必要	高度のデータ安定度が不要
読み書きトランザクション	RS	CS
読み取り専用トランザクション	RR または RS	UR

アプリケーションに適した分離レベルを選択することは、そのアプリケーションに許容されていない現象を回避するためにとても重要です。分離レベルは、アプリケーション間の分離の程度に影響を与えるだけでなく、ロックの獲得と解放に必要な CPU とメモリーのリソースが分離レベルごとに異なるため、個々のアプリケーションのパフォーマンス特性にも影響を与えます。デッドロック状態になる可能性は、分離レベルごとに異なります。

関連概念:

- 43 ページの『並行性の問題』

関連タスク:

- 48 ページの『分離レベルの指定』

分離レベルの指定

分離レベルは、データがアクセスされている間、データが他のプロセスからどのようにロックされ分離されるかを決めるものなので、並行性の要件とデータ保全性の要件のバランスを取る分離レベルを選択する必要があります。分離レベルは、作業単位の持続期間中で有効です。

分離レベルはいくつかの異なった方法で指定することができます。どの分離レベルが SQL ステートメントのコンパイルに使用されるかを判別するために、以下のヒューリスティックが使用されています。

静的 SQL

- ステートメントに分離文節が指定されている場合には、その文節の値が使用されます。
- ステートメントで指定されている分離文節がない場合には、使用される分離レベルは、パッケージがデータベースにバインドされた際にそのパッケージ用に指定されたレベルです。

動的 SQL

- ステートメントに分離文節が指定されている場合には、その文節の値が使用されます。
- ステートメントで指定されている分離文節がなく、 SET CURRENT ISOLATION ステートメントが現行セッション中に発行されている場合には、 CURRENT ISOLATION 特殊レジスターの値が使用されます。
- ステートメントで指定されている分離文節がなく、SET CURRENT ISOLATION ステートメントが現行セッション中に発行されていない場合には、使用される分離レベルは、パッケージがデータベースにバインドされた際にそのパッケージ用に指定されたレベルです。

注: 商業用に作成された数多くのアプリケーションでは、分離レベルを選択する方式を提供しています。詳細については、アプリケーション資料を参照してください。

手順:

分離レベルを指定するには、以下のようにします。

1. **プリコンパイル時またはバインド時:**

   サポートされるコンパイル言語で作成されたアプリケーションの場合、コマンド行プロセッサーの PREP または BIND コマンドの ISOLATION オプションを使用します。 PREP または BIND API を使用して、分離レベルを指定することもできます。

   - プリコンパイル時にバインド・ファイルが作成される場合、分離レベルはそのバインド・ファイル内に保管されます。バインド時に分離レベルが指定されない場合のデフォルトは、プリコンパイル時に使用された分離レベルです。
   - 分離レベルを指定しない場合、デフォルトとしてカーソル固定が使用されます。

   注: 次の照会を実行すると、パッケージの分離レベルを調べることができます。

   ```
   SELECT ISOLATION FROM SYSCAT.PACKAGES
     WHERE PKGNAME = 'XXXXXXXX'
     AND PKGSCHEMA = 'YYYYYYYY'
   ```

   XXXXXXXX はパッケージの名前、*YYYYYYYY* はパッケージのスキーマ名です。これらの名前は両方ともすべて大文字でなければなりません。

2. **REXX をサポートするデータベース・サーバーの場合:**

   データベースを作成すると、 REXX に含まれる SQL のさまざまな分離レベルのサポートに使用される複数のバインド・ファイルがデータベースにバインドされます。他のコマンド行プロセッサー・パッケージも、データベースの作成時にデータベースにバインドされます。

   データベースへの REXX とコマンド行プロセッサーによる接続では、カーソル固定のデフォルト分離レベルが使用されます。異なる分離レベルに変更しても、接続状態は変更しません。このコマンドは、CONNECTABLE AND UNCONNECTED 状態または IMPLICITLY CONNECTABLE 状態のときに実行する必要があります。

REXX アプリケーションが使用している分離レベルを確認するには、 SQLISL REXX 変数の値を調べてください。その値は、CHANGE SQLISL コマンドが実行されるたびに更新されます。

3. **ステートメント・レベルの場合:**

   WITH 文節を使用します。ステートメント・レベルの分離レベルは、ステートメントがあるパッケージに指定された分離レベルをオーバーライドします。

   以下の SQL ステートメントの分離レベルを指定することができます。

   - SELECT
   - SELECT INTO
   - Searched (検索条件付き) DELETE
   - INSERT
   - Searched (検索条件付き) UPDATE
   - DECLARE CURSOR

   以下の条件が、ステートメントに指定された分離レベルに適用されます。

   - WITH 文節を副照会で使用することはできません。
   - WITH UR オプションは、読み取り専用の操作にのみ適用されます。その他の場合には、ステートメントは UR から CS に自動的に変更されます。

4. **実行時の CLI または実行時の ODBC からの場合:**

   CHANGE ISOLATION LEVEL コマンドを使用します。DB2 コール・レベル・インターフェース (DB2 CLI) の場合は、 DB2 CLI 構成の一部として分離レベルを変更できます。実行時に、*SQLSetConnectAttr* 関数を SQL_ATTR_TXN_ISOLATION 属性と共に使用し、 *ConnectionHandle* が参照する現行接続のトランザクション分離レベルを設定してください。 db2cli.ini ファイル内で TXNISOLATION キーワードを使用することもできます。

5. **ランタイムの JDBC または SQLJ で作業する場合:**

   注: JDBC および SQLJ は、DB2 上の CLI を使用してインプリメントされています。つまり、db2cli.ini の設定は、 JDBC や SQLJ を使用して作成および実行されることに影響します。

   java.sql インターフェース接続で setTransactionIsolation 方式を使用します。

   SQLJ では、db2profc SQLJ オプティマイザーを実行して、パッケージを作成します。このパッケージに指定できるオプションには、その分離レベルが含まれます。

6. **現行セッション内での動的 SQL 用**

   SET CURRENT ISOLATION ステートメントを使用して、セッション内で発行される動的 SQL のために、分離レベルを設定します。このステートメントを発行すると、CURRENT ISOLATION 特殊レジスターは、現行セッション内で発行されるすべての動的 SQL の分離レベルを指定する値に設定されます。いったん設定されると、CURRENT ISOLATION 特殊レジスターは、どのパッケージがステートメントを発行したかに関係なく、そのセッション内でコンパイルされる後続のすべての動的 SQL ステートメントにその分離レベルを提供します。この分離

レベルは、セッションが終了するまで、または SET CURRENT ISOLATION ステートメントが RESET オプションと共に発行されるまで、適用されます。

関連概念:

- 43 ページの『並行性の問題』

関連資料:

- 「*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*」の『SQLSetConnectAttr 関数 (CLI) - 接続属性の設定』
- 「*SQL リファレンス 第 2 巻*」の『CONNECT (タイプ 1) ステートメント』
- 「*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*」の『ステートメント属性 (CLI) のリスト』

並行性制御とロッキング

次のセクションでは、ロッキングのさまざまな種類とレベル、およびこれらのロッキングがどのように決定され、データ・アクセス・パフォーマンスに影響を与えるかについて説明しています。

ロックおよび並行性の制御

並行性制御を提供し、制御されていないデータへのアクセスを回避するために、データベース・マネージャーは、バッファー・プール、表、表ブロック、または表行をロックします。 *ロック* は、データベース・マネージャー・リソースを *lock owner* というアプリケーションに関連付け、そのリソースへの他のアプリケーションからのアクセス方法を制御する手法です。

たいていのロックは表や行に対するものですが、バッファー・プールの作成、変更、またはドロップ時には、バッファー・プールのロックが設定されます。このロックで使用されるモードは EXCLUSIVE (X) です。このロックは、コマンド行プロセッサー (CLP) を使用してスナップショットを取る場合にも実行されることがあります。このスナップショットを表示すると、使用されているロック名がバッファー・プールそのものの ID であることが分かります。

データベース・マネージャーは、以下に基づいてレコード・レベルのロッキングおよび表レベルのロッキングを適宜使用します。

- プリコンパイル時、またはアプリケーションがデータベースにバインドされるときに指定される分離レベル。分離レベルは以下のいずれかです。
  - 非コミット読み取り (UR)
  - カーソル固定 (CS)
  - 読み取り固定 (RS)
  - 反復可能読み取り (RR)

  これらの異なる分離レベルは、非コミット・データへのアクセス、更新消失の防止、データの反復不能読み取りの許可、および幻像読み取りの防止を制御するために使用されます。ご使用のアプリケーションが必要とする最低の分離レベルを使用してください。

- オプティマイザーにより選択されるアクセス・プラン。表スキャン、索引スキャン、および他のデータ・アクセス方式のそれぞれについて、異なるタイプのデータ・アクセスが必要です。
- 表の LOCKSIZE 属性。 ALTER TABLE ステートメントの LOCKSIZE 文節で、表にアクセスする際に使用するロックの細分性を示します。 ROW (行ロックの場合) または TABLE (表ロックの場合) を選択できます。読み取り専用の表には ALTER TABLE... LOCKSIZE TABLE を使用してください。これにより、データベース・アクティビティーで必要なロックの数を減らすことができます。
- ロッキング専用のメモリーの量。ロッキング専用のメモリーの量は、 locklist データベース構成パラメーターにより制御されます。ロック・リストが満杯になると、ロック・エスカレーションおよびデータベースでの共用オブジェクトの並行性が悪化することが原因で、パフォーマンスが低下する可能性があります。ロック・エスカレーションが頻繁に発生する場合、 locklist か maxlocks のいずれか、あるいはその両方の値を増やしてください。

すべてのトランザクションが頻繁に COMMIT され、保持されているロックが解放されるようにしてください。

通常、以下の場合以外は、レコード・レベルのロッキングが使用されます。

- 分離レベルに非コミット読み取り (UR) が選択されている場合。
- 選択された分離レベルが反復可能読み取り (RR) で、アクセス・プランに、述部なしのスキャンが必要な場合。
- 表の LOCKSIZE 属性が「TABLE」である場合。
- ロック・リストが満杯で、エスカレーションが発生している場合。
- LOCK TABLE ステートメントを介して獲得された明示的な表ロックがある場合。 LOCK TABLE ステートメントを使用すると、並行アプリケーション・プロセスが表を変更したり表を使用したりできないようになります。

データベース内の行および表にあるロック数が maxlocks データベース構成パラメーターによって指定されたロック・リストのパーセンテージと等しいとき、*ロック・エスカレーション* が発生します。ロック・エスカレーションは、エスカレーションを起動するロックを獲得する表には影響を与えません。ロック数を、ロック・エスカレーションが始まるときに保持しているロック数の約半分に削減するために、データベース・マネージャーは、ラージ・オブジェクト (LOB) または long VARCHAR エレメント上のロックから始めて、多数の小さい行ロックをすべてのアクティブ表の表ロックに変換し始めます。*排他ロック・エスカレーション* とは、獲得された表ロックが排他ロックであるロック・エスカレーションのことです。ロック・エスカレーションは並行性を低下させます。 ロック・エスカレーションは並行性を低下させるので、回避する必要があります。

行ロッキングの期間は、使用されている分離レベルによって異なります。

- UR スキャン。行データが変更されていない限り行ロックは保持されません。
- CS スキャン。カーソルが行に位置している場合にのみ行ロックが保持されます。
- RS スキャン。トランザクション中、限定した行ロックのみ保持されます。
- RR スキャン。トランザクションの間、すべての行ロックが保持されます。

関連概念:

- 53 ページの『ロック属性』
- 55 ページの『ロックとパフォーマンス』
- 60 ページの『ロッキングのガイドライン』

関連資料:

- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 424 ページの『dlchktime - デッドロック・チェック・インターバル』
- 527 ページの『diaglevel - 診断エラー・キャプチャー・レベル』
- 425 ページの『locktimeout - ロック・タイムアウト』
- 66 ページの『ロック・タイプの互換性』
- 68 ページの『標準の表用のロック・モードおよびアクセス・パス』
- 70 ページの『MDC 表の表および RID 索引スキャンのロック・モード』
- 74 ページの『MDC 表のブロック索引スキャンのロック』

ロック属性

データベース・マネージャーのロック機能には、以下のような基本属性があります。

モード ロックの所有者に許可されるアクセスの種類、そしてロックの対象の並行ユーザーに許可されるアクセスの種類。これは、しばしばロックの*状態* と呼ばれます。

オブジェクト
ロックするリソース。明示的にロックできるオブジェクトの唯一のタイプは表です。このほかにもデータベース・マネージャーでは、行、表、表スペースなど、他のタイプのリソースにもロックをかけます。 マルチディメンション・クラスタリング (MDC) 表の場合には、ブロック・ロックをかけることもできます。ロックされているオブジェクトは、ロックの*細分性* を表します。

期間 ロックが保持される時間的長さ。障害が実行される分離レベルは、ロック期間に影響を与えます。

以下の表では、モードとその効果が示されています。ここに示す順に、リソースへの制御が大きくなります。様々なレベルでのロックに関する詳細については、ロック・モード参照表を参照してください。

表 3. ロック・モードのサマリー

ロック・モード	適用できるオブジェクト・タイプ	説明
IN (意図なし)	表スペース、ブロック、表	ロックの所有者は、非コミット・データを含め、オブジェクト内のすべてのデータを読み取ることができますが、更新はできません。 同時に実行される他のアプリケーションは、その表を読み取ったり更新したりできます。
IS (意図共用)	表スペース、ブロック、表	ロック所有者は、ロックされている表のデータを読み取ることはできますが、更新はできません。 他のアプリケーションは、その表を読み取ったり更新したりできます。

表 3. ロック・モードのサマリー (続き)

ロック・モード	適用できるオブジェクト・タイプ	説明
NS (ネクスト・キー共用)	行	ロック所有者とすべての並行アプリケーションは、ロックされた行を読み取ることはできますが、更新はできません。このロックは、S ロックの代わりに、表の行に対して獲得されます。この場合、アプリケーションの分離レベルは、RS か CS のいずれかです。 NS ロック・モードは、ネクスト・キー・ロッキングには使用されません。このモードは、これらのスキャンにおける次のキー・ロッキングの影響を最小化するために CS および RS スキャンの際に S モードの代わりに使用されます。
S (共用)	行、ブロック、表	ロック所有者とすべての並行アプリケーションは、ロックされたデータを読み取ることはできますが、更新はできません。
IX (意図排他)	表スペース、ブロック、表	ロック所有者と同時に実行されるアプリケーションとは、データを読み取ったり更新したりできます。同時に実行される他のアプリケーションは、その表を読み取ったり更新したりできます。
SIX (意図排他共用)	表、ブロック	ロック所有者は、データを読み取ったり更新したりできます。 同時に実行される他のアプリケーションは、その表を読み取ることができます。
U (更新)	行、ブロック、表	ロック所有者は、データを更新できます。他の作業単位はロックされたオブジェクトのデータを読み取ることはできますが、更新は行えません。
NW (ネクスト・キー弱排他)	行	行が索引に挿入されるとき、NW ロックが次の行に獲得されます。タイプ 2 索引の場合、次の行が現在 RR スキャンによってロックされている場合にのみ、これが発生します。ロック所有者は、ロックされた行の読み取りはできますが更新はできません。このロック・モードは、W および NS ロックと互換性があることを除けば、X ロックと類似した働きをします。
X (排他)	行、ブロック、表、バッファー・プール	ロック所有者は、ロックされたオブジェクトのデータを読み取ったり更新したりできます。ロックされたオブジェクトにアクセスできるのは、非コミット読み取りアプリケーションだけです。
W (弱排他)	行	このロックは、タイプ 2 索引が定義されていない表に行を挿入するときに、その行に対して獲得されます。ロック所有者は、ロックされた行を変更することができます。重複値が見つかるときに重複値がコミットされたかどうかを判別するために、ユニーク索引に挿入中にもこのロックが使用されます。このロックは、NW ロックと互換性があることを除けば、X ロックと類似した働きをします。 ロックされた行にアクセスできるのは、非コミット読み取りアプリケーションだけです。
Z (超排他)	表スペース、表	このロックが表上で獲得されるのは、表が変更またはドロップされるとき、表の索引が作成またはドロップされるとき、あるいは表の一部のタイプが再編成されるときなど、特定の状況においてです。同時に実行される他のアプリケーションは、その表を読み取ったり更新したりできません。

関連概念:

- 51 ページの『ロックおよび並行性の制御』
- 55 ページの『ロックとパフォーマンス』

関連資料:

- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
- 66 ページの『ロック・タイプの互換性』
- 68 ページの『標準の表用のロック・モードおよびアクセス・パス』

ロックとパフォーマンス

いくつかの関連要素が、ロックの使用およびアプリケーション・パフォーマンスに影響を与えます。この部分では、次の点について説明します。

- 並行性と細分性
- ロックの互換性
- ロックの変換
- ロック・エスカレーション
- ロックの待機とタイムアウト
- デッドロック

並行性と細分性

1 つのアプリケーションがデータベース・オブジェクト上のロックを保持している場合、別のアプリケーションはそのオブジェクトにアクセスできません。そのため、行レベルのロックは、表レベルのロックよりも最大の並行性が得られます。ただし、ロックには、ストレージや処理時間が必要なので、単一の表ロックはロック・オーバーヘッドを最小化します。

ALTER TABLE ステートメントの LOCKSIZE 文節は、行または表レベルのいずれかでロックの有効範囲 (細分性) を指定します。デフォルトでは、行ロックが使用されます。 S (共用) および X (排他) ロックのみがこれらの定義済みの表ロックによって要求されます。 ALTER TABLE ステートメントの LOCKSIZE ROW 文節を使用しても、通常のロック・エスカレーションに支障はありません。

次のような場合には、 LOCK TABLE ステートメントを使用して単一トランザクション表をロックするよりも、 ALTER TABLE を使用して永続的な表ロックをすることができます。

- 使用している表が読み取り専用であり、S ロックが必ず必要な場合。その他のユーザーは、表に S ロックを獲得することもできます。
- 表は、通常、読み取り専用アプリケーションによってアクセスされますが、時折、簡単な保守のために単一のユーザーによってアクセスされます。したがって、そのユーザーには X ロックが必要です。保守プログラムが実行されている間、読取専用アプリケーションはロックアウトされますが、その他の環境では、読み取り専用アプリケーションは、最小のロッキング・オーバーヘッドで並行して表にアクセスすることができます。

ALTER TABLE ステートメントは、グローバルにロックを指定するので、その表にアクセスするすべてのアプリケーションおよびユーザーが影響を受けます。個々のアプリケーションは、その代わりにアプリケーション・レベルで表ロックを指定するために LOCK TABLE ステートメントを使用することができます。

ロックの互換性

ロックの互換性は、表の並行アクセスにおける別の重要な要素です。ロックの互換性は、オブジェクト上の現行ロックおよびそのオブジェクトで要求されているロックのタイプを表しており、要求を許可できるかどうかを判別します。

アプリケーション A がある表のロックを保持していて、その表へのアクセスをアプリケーション B も待っているとします。データベース・マネージャーはアプリケーション B のために、ある特定のモードのロックを要求します。 A が保持しているロックのモードが、 B の要求しているロックを許可するものである場合、2 つのロック (またはモード) は、互換性があると言います。

アプリケーション B が要求したロック・モードとアプリケーション A が保持しているロックとの間に互換性がない場合には、アプリケーション B はそれ以上継続できません。その代わり、アプリケーション A がロックを解放し、さらに既存の非互換のロックがすべて解放されるまで待機する必要があります。

ロックの変換

すでに保持しているロック・モードの変更は、*変換* と呼ばれます。ロックの変換が行われるのはあるプロセスがすでにロックを保持しているデータ・オブジェクトにアクセスする場合に、そのアクセスのモードが、すでに保持しているロックよりさらに制約の大きいロックを必要とするものである場合です。照会によって間接的に 1 プロセスの中で同じデータ・オブジェクトに、何度もロックを要求できますが、1 つのデータ・オブジェクトのロックは 1 回に 1 つしか保持できません。

一部のロック・モードは表にのみ適用され、その他のロック・モードは行またはブロックにのみ適用されます。行またはブロックの場合、通常、X が必要なときに S または U (更新) ロックを保持している場合に、変換が行われます。

しかし、IX (意図排他) と S (共用) ロックは、ロック変換に関しては特殊なケースです。 S と IX はどちらも他方よりも制約が大きいとは見なされないため、これらのロックの一方を保持しているときに他方が必要になった場合には、結果として SIX (意図排他共用) ロックに変換されることになります。 他のすべての変換の結果は、要求されているモードの制限がより大きい場合は、要求されたロック・モードが、保持するロックのモードになります。

照会が行を更新するとき、二重変換が発生する場合もあります。索引アクセスによって行が読み取られ、S としてロックされる場合、行を含む表には、目的ロックが含まれています。ただし、ロック・タイプが IX ではなく IS である場合、後で行が変更されると、表ロックは IX に変換され、行ロックは X に変換されることになります。

ロック変換は、通常、照会が実行されるときに暗黙的に行われます。各種の照会および表と索引の組み合わせの下で発行されるロックの種類について知っておくことは、アプリケーションの設計と調整に役立つでしょう。

ロック・エスカレーション

ロック・エスカレーションは、保持されるロックの数を減らすための内部機構です。単一表では、ロックは多数の行ロックから表ロックにエスカレーションされ、マルチディメンション・クラスタリング (MDC) 表では、多数の行またはブロック・ロックから表ロックにエスカレーションされます。ロック・エスカレーションは、

アプリケーションのロック・タイプの保持が多すぎるときに起こります。ロック・エスカレーションは、特定のデータベース・エージェントがロック・リストのそのエージェントに割り振られた分を超過した場合にも、起こることがあります。 このようなエスカレーションは内部的に処理されるので、 1 つまたは複数の表への並行アクセスが少なくなるのは、外部から検出できる唯一の結果です。

適正に構成されたデータベースでは、ロック・エスカレーションはほとんど行われません。ロック・エスカレーションが起きる場合の例としては、アプリケーション設計者が大きな表で索引を作成してパフォーマンスと並行性を向上させようとしたが、そのトランザクションはその表の大部分のレコードにアクセスするというような場合があげられます。この場合、データベース・マネージャーは、表の大部分がロッキングされることを予測できないため、 S または X の表だけをロッキングするのではなく、各レコードを個別にロッキングします。このような場合の解決策として、データベース設計者は、アプリケーション設計者とも相談した上で、このトランザクションに LOCK TABLE ステートメントを使用するように推奨することができます。

時折、内部エスカレーション要求を受信するプロセスは、表にほとんど行ロックを保持していないか全く行ロックを保持していなくても、 1 つ以上のプロセスが多数のロックを保持しているためにロックがエスカレーションされることがあります。そのプロセスは、トランザクション終了時を除いて、別のロックを要求したりデータベースにアクセスしたりしないかもしれません。そして、別のプロセスが、エスカレーション要求を引き起こすロックを要求する場合があります。

注: ロック・エスカレーションは、デッドロックを引き起こすこともあります。たとえば、読み取り専用アプリケーションと更新アプリケーションが両方とも同じ表にアクセスするとします。更新アプリケーションが表の多数の行に排他ロックを持っている場合、データベース・マネージャーはこの表のロックを排他表ロックにエスカレーションしようとします。ただし、読み取り専用アプリケーションが保持している表ロックによって、排他ロックのエスカレーション要求が待機させられることがあります。更新アプリケーションがすでにロックしている行に、読み取り専用アプリケーションが行ロックを要求する場合、デッドロックが作成されます。この種類の問題を回避するには、開始時に、表を排他的にロックするように更新アプリケーションをコーディングするか、またはロック・リストのサイズを大きくしてください。

ロックの待機とタイムアウト

ロックのタイムアウト検出は、ロックが異常な状況で解放されるまで無限にアプリケーションが待機しなくて済むようにするデータベース・マネージャーの機能です。たとえば、あるトランザクションが別のユーザーのアプリケーションによって保持されているロックを待機しており、一方でそのユーザーがトランザクションをアプリケーションがコミットできるようにロックを解放するということをせずに作業場から席をはずしてしまっているかもしれません。こうしたケースでアプリケーションが停止しないようにするには、 *locktimeout* 構成パラメーターを使用して、アプリケーションがロックを獲得するまで待機する最大待ち時間を設定します。

このパラメーターを設定すると、特に分散作業単位 (DUOW) アプリケーションにおいては、グローバル・デッドロックを避けることができます。 ロック要求がペンディングにされている時間が *locktimeout* 値より長くなると、要求しているアプリケー

ションはエラーを受け取り、トランザクションがロールバックされます。たとえば、*program1* が、すでに *program2* によって保持されているロックを獲得しようとするときにタイムアウトになると、 *program1* は SQLCODE -911 と理由コード 68 を返します。 *locktimeout* のデフォルト値は -1 です。これにより、ロックのタイムアウト検出はオフになります。

注: アプリケーションは、SET CURRENT LOCK TIMEOUT を使用して、表、行、および MDC ブロック・ロックに対するデータベース・レベルの *locktimeout* 設定をオーバーライドできます。

管理通知ログ内にロック要求タイムアウトの詳細をログに記録するには、データベース・マネージャーの構成パラメーター *notifylevel* を 4 に設定します。ログに記録された情報には、オブジェクト、ロック・モード、およびロックを保持しているアプリケーションが含まれます。また、現行の動的 SQL ステートメントまたは静的パッケージ名もログに記録されている可能性があります。動的 SQL ステートメントは、*notifylevel* が 4 である場合にのみログに記録されます。

デッドロック

ロックの競合により、デッドロックが生じます。たとえば、プロセス 1 が X (排他) モードで表 A にロックをかけ、プロセス 2 が X モードで表 B にロックをかけるとします。次にプロセス 1 が X モードで表 B にロックをかけようとし、さらにプロセス 2 が X モードで表 A にロックをかけようとすると、それらのプロセスはデッドロックに陥ります。デッドロック状態になると、プロセスは両方とも、それらの 2 度目のロック要求が受け入れられるまで中断状態になり、しかもどちらの要求も、いずれかのプロセスがコミットまたはロールバックを実行するまで受け入れられることはありません。 外部エージェントがいずれかのプロセスを活動化し、強制的にロールバックを実行させるまで、この状態が無限に続きます。

デッドロックを処理するには、データベース・マネージャーは、デッドロック検出機能と呼ばれる非同期システム背景プロセスを使用します。デッドロック検出機能は、 *dlchktime* 構成パラメーターによって指定された周期でアクティブになります。デッドロック検出機能がアクティブになると、ロック・システムがデッドロック状態になっていないかどうかを調べます。パーティション・データベースでは、それぞれのパーティションは、システム・カタログ・ビューを含むデータベース・パーティションにロック・グラフ を送信します。グローバルなデッドロック検出は、このパーティション上で行われます。

デッドロックが見つかる場合、デッドロック検出機能は、1 つのデッドロック・プロセスを、ロールバックする*選択されたプロセス* として選択します。選択されたプロセスはアクティブにされ、呼び出し側アプリケーションに SQLCODE -911 (SQLSTATE 40001) 理由コード 2 で戻されます。データベース・マネージャーは選択されたプロセスを自動的にロールバックします。ロールバックが完了すると、選択されたプロセスに属するロックは解除され、それによってそのデッドロックにかかわっていた他のプロセスが先に進めるようになります。

デッドロック検出機能の適切な時間間隔を選択することは、適正なパフォーマンスを確保するうえで必要なことです。時間間隔が短すぎると不必要なオーバーヘッドが生じ、長すぎるとデッドロックによるプロセスの遅延が長くなってしまいます。たとえば、5 分というウェイクアップ・インターバルを選択すると、デッドロック

はほぼ 5 分間存在することができます。これは、短いトランザクション処理の場合には長い時間のように思われます。デッドロックを解決するための遅延と、デッドロックを検出するオーバーヘッドとの間でバランスを保つ必要があります。

パーティション・データベースでは、 *dlchktime* 構成パラメーター・インターバルは、カタログ・ノードにのみ適用されます。大量のデッドロックがパーティション・データベースで検出される場合には、 *dlchktime* パラメーターの値を大きくして、ロック待機や通信待機を解決するようにしてください。

データベースにアクセスする独立したプロセスが複数個あるアプリケーションが、デッドロックが生じやすい構造になっている場合は、別の問題が起きる可能性があります。 その一例は、いくつかのプロセスが同じ表にアクセスして、読み取りを行ってから書き込みを行うようになっているアプリケーションです。それらのプロセスが最初に読み取り専用 SQL 照会を行い、次に同じ表に対する SQL 更新を行うと、プロセス相互間で同じデータに対する競合が生じる可能性があるため、デッドロックの起きる可能性が大きくなります。たとえば、2 つのプロセスが同じ表を読み取った後でその表の更新を行うと、プロセス A がある行に対する X ロックを入手しようとしているが、プロセス B がその行に対する S ロックを持っている (あるいは、その反対) という状況になります。こうしたデッドロックを避けるため、修正する目的でデータにアクセスするアプリケーションは、以下のいずれかを実行してください。

- 選択を実行するときに、FOR UPDATE OF 文節を使用する。それによって、プロセス A がデータを読み取ろうとするとき、 U ロックがかけられるようになります。しかし、行ブロッキングは使用不可になります。
- 照会を実行するときに、WITH RR USE AND KEEP UPDATE LOCKS または WITH RS USE AND KEEP UPDATE LOCKS 文節を使用する。どちらかの文節を使用すると、プロセス A がデータを読み取ろうとするとき、 U ロックがかけられ、行ブロッキングが可能になります。

注: デッドロックが生じるときを記録するモニターを定義することを考慮する場合があります。 SQL ステートメント CREATE EVENT を使用して、モニターを作成してください。

データベースが作成されると同時に、詳細デッドロック・イベント・モニターも作成されます。他のモニターと同様に、このイベント・モニターにも関連したオーバーヘッドがあります。詳細デッドロック・イベント・モニターを必要としない場合は、以下のコマンドを使用してイベント・モニターをドロップできます。

```
DROP EVENT MONITOR db2detaildeadlock
```

このイベント・モニターが消費するディスク・スペースの量を制限するために、出力ファイルの最大数に達すると、イベント・モニターが非アクティブになり、メッセージが管理通知ログに書き込まれます。必要のない出力ファイルを除去すると、イベント・モニターは次のデータベースの活動化時にアクティブになります。

アプリケーションがニックネームにアクセスするフェデレーテッド・システム環境では、アプリケーションによって要求されたデータが、データ・ソースでデッドロックが起こっているために使用できない可能性があります。 このことが起こると、

DB2® はデータ・ソースでのデッドロック処理機能により、ロックを解決します。複数のデータ・ソースにまたがるデッドロックが生じる場合は、 DB2 はデータ・ソースのタイムアウト機構により、デッドロックを解除します。

デッドロックに関する詳細をログに記録するには、データベース・マネージャーの構成パラメーター *notifylevel* を 4 に設定します。管理通知ログには、オブジェクト、ロック・モード、およびこのオブジェクトへのロックを保持しているアプリケーションを含む情報が保管されます。また、現行の動的 SQL ステートメントまたは静的パッケージ名もログに記録されている可能性があります。動的 SQL ステートメントは、 *notifylevel* が 4 である場合にのみログに記録されます。

関連概念:

- 51 ページの『ロックおよび並行性の制御』
- 11 ページの『アプリケーション間のデッドロック』

関連タスク:

- 62 ページの『ロック・エスカレーション問題の修正』

関連資料:

- 527 ページの『diaglevel - 診断エラー・キャプチャー・レベル』
- 425 ページの『locktimeout - ロック・タイムアウト』

ロッキングのガイドライン

並行性およびデータ保全性のためにロッキングを調整するときは、以下の指針を考慮してください。

- 多数のユーザーによるデータの並行アクセスを促進する頻繁な COMMIT ステートメントを含む小さな作業単位を作成します。

  アプリケーションが論理的に一貫しているとき、つまり変更したデータが一貫したものになっているときには、 COMMIT ステートメントを含めます。
  COMMIT が発行されると、ロックは解除されます。ただし、 WITH HOLD と宣言されたカーソルに関連する表ロックは除きます。
- 適切な分離レベルを指定します。

  アプリケーションが行を読み取るだけでもロックは獲得されるので、読み取り専用の作業単位をコミットすることは非常に重要です。これは、共用ロックが、読み取り専用アプリケーション内で反復可能読み取り、読み取り固定、およびカーソル固定によって獲得されるからです。反復可能読み取りと読み取り固定の場合は、 WITH RELEASE 文節を使用してカーソルをクローズしない限り、すべてのロックは COMMIT が出されるまで保持されて、ロックされたデータが他のプロセスによって更新されることのないようにします。加えて、カタログ・ロックは動的 SQL を使用している非コミット読み取りアプリケーション内であっても獲得されます。

  データベース・マネージャーでは、非コミット読み取り分離レベルが使用されていない限り、アプリケーションが非コミットのデータ (他のアプリケーションによって更新されたが、まだコミットされていない行) を検索しないようにしています。

- LOCK TABLE ステートメントを正しく使用します。

  このステートメントは表全体にロックをかけます。 LOCK TABLE ステートメントに指定された表だけがロックされます。指定された表の親表と従属表はロックされません。アクセス可能な他の表をロッキングすることが望ましい結果になるかどうかを、並行性とパフォーマンスの観点から判断する必要があります。その作業単位がコミットまたはロールバックされるまで、ロックは解除されません。

  LOCK TABLE IN SHARE MODE

  時間の点で一致した データにアクセスしたい、つまり特定の時点での表の現行データにアクセスしたい場合。表への活動が頻繁な場合、表全体を一定であるようにするための唯一の方法は、表をロックすることです。たとえば、アプリケーションで表のスナップショットを取りたいとします。しかし、アプリケーションが表のいくつかの行を処理する必要があるときに、他のアプリケーションが未処理の表を更新しています。反復可能読み取りではこれが可能ですが、この処置は希望するものとは異なります。

  別の手段として、アプリケーションは LOCK TABLE IN SHARE MODE ステートメントを発行できます。行が取り出されたかどうかに関係なく、行は変更できません。それら取り出した行が検索前と比べて変更されていないことが分かっているので、必要なだけの行を取り出すことができます。

  LOCK TABLE IN SHARE MODE を使用すると、他のユーザーは表からデータを検索できますが、表内の行を更新、削除、または挿入することはできません。

  LOCK TABLE IN EXCLUSIVE MODE

  表の大部分を更新したい場合。他のすべてのユーザーがその表をアクセスできないようにすることは、更新する各行ごとにロックをかけ、変更をすべてコミットした後で行をアンロックするより安価で効率的です。

  LOCK TABLE IN EXCLUSIVE MODE を使用すると、他のユーザーはすべてロックアウトされます。他のアプリケーションは、非コミット読み取りアプリケーションでない限り、表にアクセスすることはできません。

- アプリケーションで ALTER TABLE ステートメントを使用します。

  LOCKSIZE パラメーターを指定した ALTER TABLE ステートメントは、 LOCK TABLE ステートメントの代わりになります。 LOCKSIZE パラメーターでは、次の表アクセスの行ロックまたは表ロックのロックの細分性を指定できます。

  ROW ロックを選択することは、表が作成されるときにデフォルト・ロック・サイズを選択することと何ら変わりありません。 TABLE ロックを選択すると、獲得する必要のあるロックの数が限定されることによって、照会のパフォーマンスが向上します。ただし、ロックはすべて表全体にかけられるので、並行性は低下します。どちらを選択しても、通常のロック・エスカレーションは妨げられません。

- カーソルをクローズして、カーソルが保持しているロックを解放します。

  CLOSE CURSOR ステートメント (その中に WITH RELEASE 文節が入っている) を使用してカーソルをクローズすると、データベース・マネージャーは、そのカーソルのために保持された読み取りロックをすべて解放しようとします。表

読み取りロックには、表ロック IS、S、および U があります。行読み取りロックには、行ロック S、NS、および U があります。ブロック読み取りロックには、ブロック・ロック IS、S、および U があります。

WITH RELEASE 文節は、CS または UR の分離レベルで作動しているカーソルには影響がありません。 RS または RR 分離レベルで作動するカーソルに関して WITH RELEASE 文節を指定すると、それらの分離レベルの保証はいくつか終了してしまいます。特に、RS カーソルでは*非反復可能読み取り* 現象が起こり、RR カーソルでは*非反復可能読み取り* か*幻像読み取り* 現象が起こることがあります。

もともと RR または RS であるカーソルが、 WITH RELEASE 文節を使ってクローズされた後に再度オープンされた場合は、新たに読み取りロックが獲得されます。

DB2® CLI 接続属性 SQL_ATTR_CLOSE_BEHAVIOR を CLI アプリケーションで使用すると、 CLOSE CURSOR WITH RELEASE と同じ結果を得ることができます。

- パーティション・データベース内でロッキングに影響を与える構成パラメーターを変更するときは、すべてのパーティションに変更を加えたかどうか確認してください。

関連概念:

- 51 ページの『ロックおよび並行性の制御』
- 53 ページの『ロック属性』
- 55 ページの『ロックとパフォーマンス』
- 76 ページの『ロッキングに影響を与える要因』

ロック・エスカレーション問題の修正

データベース・マネージャーは、行またはブロック・レベルから表レベルにロックを自動的にエスカレーションすることができます。 *maxlocks* データベース構成パラメーターは、ロック・エスカレーションが起動されるときを指定します。ロック・エスカレーションを起動するロックを獲得する表には影響を与えません。ラージ・オブジェクト (LOB) および LONG VARCHAR 記述子がロックされている表から始めて、まず、大部分のロックを持つ表のロックがエスカレーションされ、その後、次に高いロック数を持つ表というように、保持しているロック数が *maxlocks* で指定された値の約半分になるまでエスカレーションされます。

適切に設計されているデータベースでは、ロック・エスカレーションはめったに起こりません。ただし、ロック・エスカレーションによって並行性が受諾不能なレベルまで下がると、問題を分析し、問題の解決方法を判別する必要があります。

前提条件:

ロック・エスカレーション情報が記録されていることを確認します。データベース・マネージャーの構成パラメーター *notifylevel* を 3 (デフォルト) または 4 に設定します。 2 という *notifylevel* では、エラー SQLCODE のみが報告されます。 3 または 4 という *notifylevel* では、ロック・エスカレーションが失敗するとき、エラ

一 SQLCODE およびエスカレーションが失敗した表に関する情報が記録されます。現行の SQL ステートメントは、現在実行している場合にのみログに記録され、動的 SQL ステートメントおよび *notifylevel* は 4 に設定されます。

手順:

受諾不能なロック・エスカレーションの原因を診断するには、以下の一般的なステップに従い、予防手段を適用してください。

1. ロックがエスカレーションされているすべての表の管理通知ログ内を分析します。このログ・ファイルには、以下の情報が含まれています。
   - 現在保持されているロックの数。
   - ロック・エスカレーションが完了する前に必要なロックの数。
   - エスカレーションされているそれぞれの表の表 ID 情報と表名。
   - 現在保持されている非表ロックの数。
   - エスカレーションの一部として獲得される新しい表レベルのロック。 通常、「S」(共用ロック) または「X」(排他ロック) が獲得されます。
   - 新しい表ロック・レベルを獲得した結果として生じる内部戻りコード。
2. 管理通知ログ内の情報を使用して、エスカレーション問題の解決方法を判別します。以下の可能性を考慮します。
   - データベース構成ファイルの *maxlocks* パラメーターまたは *locklist* パラメーター (あるいはその両方) の値を大きくすることによって、グローバルに許可されるロックの数を増やす。パーティション・データベースでは、すべてのパーティションでこの変更を行ってください。

     他のプロセスからの表への並行アクセスが最重要である場合には、これが適切な処置となります。しかし、レコード・レベルのロックの取得によるオーバーヘッドのために、表への並行アクセスによって節約できる量を超えて、他のプロセスに遅延が誘発されることもあります。
   - エスカレーションを引き起こしたプロセスをエスカレーションします。これらのプロセスの場合、明示的に LOCK TABLE ステートメントを発行することができます。
   - 分離の程度を変更します。しかしこの場合、並行性が低下する可能性があります。
   - コミットの頻度を増やして、特定の時間に存在するロックの数を減らす。
   - LONG VARCHAR または様々な種類のラージ・オブジェクト (LOB) データが必要なトランザクションに対する頻繁な COMMIT ステートメントを考慮します。結果セットがマテリアライズされるまで、この種類のデータはディスクから検索されませんが、まずデータが参照されるときに記述子がロックされます。その結果、一般的なデータを含む行のロックよりもさらに多くのロックが保持されている場合があります。

関連資料:

- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
- 527 ページの『diaglevel - 診断エラー・キャプチャー・レベル』

ロック据え置きによる非コミット・データの評価

並行性を向上させる目的で、DB2® では、レコードが照会の述部を満たしたことがわかるようになるまで、場合によっては CS または RS 分離スキャンの行ロックを据え置くことができます。デフォルトでは、表または索引スキャン中に行ロックが実行されると、 DB2 はスキャンされる各行をロックしてから、行が照会の条件を満たしているかどうかを判別します。スキャンの並行性を向上させるために、行が照会の条件を満たしていると判別されるまで行ロックを据え置くことができます。

この機能を利用するには、DB2_EVALUNCOMMITTED レジストリー変数を使用可能にしてください。

この変数を使用可能にすると、非コミット・データに対して述部評価を行えるようになります。これは、非コミット更新を含む行が照会を満たしていなくても、更新されたトランザクションが完了してから述部評価を行うようにすれば、行が照会を満たしている場合があるということを意味します。さらに、削除された非コミット行は、表スキャン中にスキップされます。 DB2_SKIPDELETED レジストリー変数を使用可能にすると、 DB2 は削除されたキーをタイプ 2 索引スキャンでスキップします。

これらのレジストリー変数の設定は、動的 SQL の場合はコンパイル時に適用され、静的 SQL の場合はバインド時に適用されます。つまり、レジストリー変数を実行時に使用可能にしても、 DB2_EVALUNCOMMITTED をバインド時に使用可能にしない限り、ロック回避ストラテジーは採用されません。レジストリー変数を実行時ではなくバインド時に使用可能にすると、ロック回避ストラテジーは有効になります。静的 SQL の場合は、パッケージを再バインドすると、バインド時のレジストリー変数の設定が適用されます。静的 SQL を暗黙的に再バインドすると、DB2_EVALUNCOMMITTED の現在の設定が使用されます。

各種アクセス・プランの非コミットの評価の適用度

表 4. RID 索引単独アクセス

述部	非コミットの評価
なし	不可
SARGable 述部	可

表 5. データ単独アクセス (リレーショナルまたは据え置き RID リスト)

述部	非コミットの評価
なし	不可
SARGable 述部	可

表 6. RID 索引 + データ・アクセス

述部		非コミットの評価	
索引	データ	索引アクセス	データ・アクセス
なし	なし	不可	不可
なし	SARGable 述部	不可	不可
SARGable 述部	なし	可	不可

表 6. RID 索引 + データ・アクセス (続き)

述部		非コミットの評価	
索引	**データ**	**索引アクセス**	**データ・アクセス**
SARGable 述部	SARGable 述部	可	不可

表 7. ブロック索引 + データ・アクセス

述部		非コミットの評価	
索引	**データ**	**索引アクセス**	**データ・アクセス**
なし	なし	不可	不可
なし	SARGable 述部	不可	可
SARGable 述部	なし	可	不可
SARGable 述部	SARGable 述部	可	可

例

以下の例では、デフォルトのロック動作と新規の非コミットの評価動作を比較します。

以下の表は、SAMPLE データベースの ORG 表です。

```
DEPTNUMB DEPTNAME       MANAGER DIVISION   LOCATION
-------- -------------- ------- ---------- -------------
      10 Head Office        160 Corporate  New York
      15 New England         50 Eastern    Boston
      20 Mid Atlantic        10 Eastern    Washington
      38 South Atlantic      30 Eastern    Atlanta
      42 Great Lakes        100 Midwest    Chicago
      51 Plains             140 Midwest    Dallas
      66 Pacific            270 Western    San Francisco
      84 Mountain           290 Western    Denver
```

この表に対して、以下のトランザクションがデフォルトのカーソル固定 (CS) 分離レベルで実行されます。

表 8. ORG 表に対する CS 分離レベルでのトランザクション

セッション 1	セッション 2
SAMPLE への接続	SAMPLE への接続
+c update org set deptnumb=5 where manager=160	
	select * from org where deptnumb >= 10

セッション 1 の非コミット UPDATE は、表の最初の行に対する排他的レコード・ロックを保持し、セッション 1 で更新中の行が現在セッション 2 の照会を満たしていない場合でも、セッション 2 の SELECT 照会が戻らないようにしています。これは、照会でアクセスされる行は、その行にカーソルが置かれている間はロックされなければならないと CS 分離レベルによって命令されているためです。セッション 2 は、セッション 1 がロックを解除するまで最初の行をロックできません。

表のスキャン中は、非コミットの評価機能を使用してセッション 2 でのロック待機を回避できます。この機能は、最初に述部を評価してから、実際の述部の評価を行うために行をロックします。このように、セッション 2 の照会では表の最初の行を

ロックしようとしないので、アプリケーション並行性が向上します。これは、セッション 2 の述部は、セッション 1 の deptnumb=5 という非コミット値に関して評価されるという意味でもあります。セッション 2 の照会では、セッション 1 の更新のロールバックがセッション 2 の照会を満たしているにもかかわらず、結果セットの最初の行が省略されます。

操作の順序を逆にすると、非コミットの評価で並行性がさらに向上する可能性があります。デフォルトのロッキング動作では、セッション 2 で最初に行ロックが獲得され、セッション 2 の照会でロックされる行をセッション 1 の UPDATE が変更しない場合でも、セッション 1 の検索済み UPDATE が実行されないようにします。セッション 1 の検索済み UPDATE が最初に行を検査して、条件を満たしている場合にのみ行をロックしようとする場合は、セッション 1 の照会は非ブロッキングになります。

制約事項

この新機能には、以下の外部制約事項が適用されます。

- レジストリー変数 DB2_EVALUNCOMMITTED を使用可能にする必要があります。
- 分離レベルは CS または RS でなければなりません。
- 行ロックが発生します。
- SARGable 述部評価述部が存在します。
- 非コミットの評価はカタログ表でのスキャンに適用できません。
- MDC 表の場合は索引スキャンのためにブロック・ロッキングを据え置くことができますが、表スキャンの場合はブロック・ロッキングを据え置くことはできません。
- 据え置きロッキングは、インプレース表 REORG を実行している表に対しては発生しません。
- 索引がタイプ 1 の索引スキャンでは、据え置きロッキングは発生しません。
- Iscan-Fetch プランの場合は、行ロッキングはデータ・アクセスのために据え置かれるのではなく、表の中の行に移動する前に索引アクセス中に行がロックされます。
- 削除された行は表スキャン中に無条件でスキップされますが、削除されたタイプ 2 索引キーはレジストリー変数 DB2_SKIPDELETED が使用可能になっている場合にのみスキップされます。

ロック・タイプの互換性

以下の表では、特定の状態で別のプロセスが同じリソースにロックを保持または要求しているときに、ロック要求が認可される状態に関する情報を表示します。**いいえ**は、非互換ロックが他のプロセスによってすべて解除されるまで、要求側が待機しなければならないことを示します。要求側がロック待機中に、タイムアウトになることがあるので注意してください。**はい**は、ロックが認可されることを示します (ただし、他のだれかがそのリソースを待っていない場合に限ります)。

表 9. ロック・タイプの互換性

要求されている状態	保持されているリソースの状態											
	なし	**IN**	**IS**	**NS**	**S**	**IX**	**SIX**	**U**	**X**	**Z**	**NW**	**W**
なし	はい	はい	はい	はい	はい	はい	はい	はい	はい	はい	はい	はい
IN	はい	はい	はい	はい	はい	はい	はい	はい	はい	いいえ	はい	はい
IS	はい	はい	はい	はい	はい	はい	はい	はい	いいえ	いいえ	いいえ	いいえ
NS	はい	はい	はい	はい	はい	いいえ	いいえ	はい	いいえ	いいえ	はい	いいえ
S	はい	はい	はい	はい	はい	いいえ	いいえ	はい	いいえ	いいえ	いいえ	いいえ
IX	はい	はい	はい	いいえ	いいえ	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
SIX	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
U	はい	はい	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
X	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
Z	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ
NW	はい	はい	いいえ	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい
W	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	いいえ

注:

I 意図
N なし
NS ネクスト・キー共用
S 共用
X 排他
U 更新
Z 超排他
NW ネクスト・キー弱排他
W 弱排他

注:

- はい - 要求されたロックがただちに**付与される**
- いいえ - 保持しているロックを解放するまで、またはタイムアウトになるまで**待機する**

関連概念:

- 51 ページの『ロックおよび並行性の制御』
- 53 ページの『ロック属性』
- 55 ページの『ロックとパフォーマンス』

関連資料:

- 68 ページの『標準の表用のロック・モードおよびアクセス・パス』
- 74 ページの『MDC 表のブロック索引スキャンのロック』

標準の表用のロック・モードおよびアクセス・パス

このトピックには、異なるデータ・アクセス・プランに関する標準の表のロッキング方式に関する参照情報が含まれています。

以下の表では、異なるアクセス・プランのそれぞれのレベルの標準の表で獲得されるロックのタイプをリストしています。それぞれの項目は、表ロックおよび行ロックの 2 つの部分から成り立っています。ダッシュは、特定のレベルのロッキングは行われないことを示します。

注:

1. マルチディメンション・クラスタリング (MDC) 環境では、さらにロック・レベル BLOCK が使用されます。
2. ロック・モードは、SELECT ステートメントのロック要求文節を使用して明示的に変更できます。

表 10. 表スキャンのロック・モード

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
アクセス方式: *述部なしの表スキャン*					
RR	S/-	U/-	SIX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
アクセス方式: *述部での表スキャン*					
RR	S/-	U/-	SIX/X	U/-	SIX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

注: タイプ 1 索引の IN ロックを持つ UR 分離レベルの場合、または索引の組み込み列に述部がある場合、分離レベルは、CS および IS 表ロックや NS 行ロックへのロックにアップグレードされます。

表 11. RID 索引スキャンのロック・モード

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
アクセス方式: *述部なしの RID 索引スキャン*					
RR	S/-	IX/S	IX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X

表 11. RID 索引スキャンのロック・モード (続き)

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
アクセス方式: *単一修飾行の RID 索引スキャン*					
RR	IS/S	IX/U	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
アクセス方式: *開始述部と停止述部のみの索引スキャン*					
RR	IS/S	IX/S	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
アクセス方式: *索引およびその他の述部 (sargs、resids) のみの索引スキャン*					
RR	IS/S	IX/S	IX/X	IX/S	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

次の表には、データ・ページの読み取りの際のロック・モードが据え置かれ、行のリストに対して以下の処理ができることが示されています。

- 複数の索引を使用して、さらに条件を付ける
- 効果的なプリフェッチが行えるようにソートする

表 12. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
アクセス方式: *述部なしの RID 索引スキャン*					
RR	IS/S	IX/S		X/-	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	
アクセス方式: *据え置きデータ・ページ・アクセス (述部なしの RID 索引スキャン後)*					
RR	IN/-	IX/S	IX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X

表 12. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード (続き)

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
アクセス方式: *述部 (sargs、residds) での RID 索引スキャン*					
RR	IS/S	IX/S		IX/S	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	
アクセス方式: *開始述部と停止述部のみの RID 索引スキャン*					
RR	IS/S	IX/S		IX/X	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	
アクセス方式: *据え置きデータ・ページ・アクセス (開始述部と停止述部のみの RID 索引スキャン後)*					
RR	IN/-	IX/S	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IS/-	IX/U	IX/X	IX/U	IX/X
アクセス方式: *据え置きデータ・ページ・アクセス (述部での RID 索引スキャン後)*					
RR	IN/-	IX/S	IX/X	IX/S	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

関連概念:

- 53 ページの『ロック属性』
- 55 ページの『ロックとパフォーマンス』

関連資料:

- 66 ページの『ロック・タイプの互換性』
- 70 ページの『MDC 表の表および RID 索引スキャンのロック・モード』
- 74 ページの『MDC 表のブロック索引スキャンのロック』

MDC 表の表および RID 索引スキャンのロック・モード

マルチディメンション・クラスタリング (MDC) 環境では、さらにロック・レベル BLOCK が使用されます。以下の表には、種々のアクセス・プランの各レベルで取

得されるロックのタイプをリストしています。各項目は、表ロック、ブロック・ロック、行ロックの 3 つの部分から成ります。ダッシュは、特定のレベルのロッキングは使用されないことを示します。

注: ロック・モードは、SELECT ステートメントのロック要求文節を使用して明示的に変更できます。

表 13. 表スキャンのロック・モード

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
アクセス方式: *述部なしの表スキャン*					
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
アクセス方式: *ディメンション列のみについての述部での表スキャン*					
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
アクセス方式: *他の述部 (sargs、resids) での表スキャン*					
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

以下の 2 つの表には、 MDC 表上の RID 索引のロック・モードが表されています。

表 14. RID 索引スキャンのロック・モード

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
アクセス方式: *述部なしの RID 索引スキャン*					
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X
アクセス方式: *単一修飾行の RID 索引スキャン*					
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X

表 14. RID 索引スキャンのロック・モード (続き)

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X
アクセス方式: *開始述部と停止述部のみの RID 索引スキャン*					
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
アクセス方式: *索引述部のみでの索引スキャン*					
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
アクセス方式: *他の述部 (sargs、resids) での索引スキャン*					
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

注: 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モードを示す以下の表において、 IN ロックを含む UR 分離レベルでは、タイプ 1 索引の場合や索引の組み込み列上に述部が存在している場合、分離レベルは CS にアップグレードされ、ロックは IS 表ロック、IS ブロック・ロック、および NS 行ロックにアップグレードされます。

表 15. 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モード

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
アクセス方式: *述部なしの RID 索引スキャン*					
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
アクセス方式: *据え置きデータ・ページ・アクセス (述部なしの RID 索引スキャン後)*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 15. 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モード (続き)

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
アクセス方式: *述部 (sargs、resids) での RID 索引スキャン*					
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
アクセス方式: *据え置きデータ・ページ・アクセス (述部 (sargs、resids) での RID 索引スキャン後)*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
アクセス方式: *開始述部と停止述部のみの RID 索引スキャン*					
RR	IS/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
アクセス方式: *据え置きデータ・ページ・アクセス (開始述部と停止述部のみの RID 索引スキャン後)*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

関連概念:

- 51 ページの『ロックおよび並行性の制御』
- 53 ページの『ロック属性』
- 55 ページの『ロックとパフォーマンス』

関連資料:

- 66 ページの『ロック・タイプの互換性』
- 68 ページの『標準の表用のロック・モードおよびアクセス・パス』
- 74 ページの『MDC 表のブロック索引スキャンのロック』

MDC 表のブロック索引スキャンのロック

以下の表には、種々のアクセス・プランの各レベルで取得されるロックのタイプをリストしています。各項目は、表ロック、ブロック・ロック、行ロックの 3 つの部分から成ります。ダッシュは、特定のレベルのロッキングは行われないことを示します。

注: ロック・モードは、SELECT ステートメントのロック要求文節を使用して明示的に変更できます。

表 16. 索引スキャンのロック・モード

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
アクセス方式: *述部なし*					
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--
アクセス方式: *ディメンション述部のみ*					
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
アクセス方式: *ディメンション開始述部とディメンション停止述部のみ*					
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
アクセス方式: *述部ありの索引スキャン*					
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

以下の表には、据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モードをリストしています。

表 17. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
アクセス方式: *述部なしのブロック索引スキャン*					
RR	IS/S/--	IX/IX/S		X/--/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	
アクセス方式: *据え置きデータ・ページ・アクセス (述部なしのブロック索引スキャン後)*					
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--
アクセス方式: *ディメンション述部のみのブロック索引スキャン*					
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	
アクセス方式: *据え置きデータ・ページ・アクセス (ディメンション述部のみのブロック索引スキャン後)*					
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
アクセス方式: *開始述部と停止述部のみのブロック索引スキャン*					
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	
Access Method: *据え置きデータ・ページ・アクセス (開始述部と停止述部のみのブロック索引スキャン後)*					
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	
UR	IS/--/--	IN/IN/--		IN/IN/--	
アクセス方式: *他の述部 (**sargs**、**resids**) でのブロック索引スキャン*					
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	

表 17. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード (続き)

分離レベル	読取専用および未確定のスキャン	カーソル操作		検索条件付き更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
UR	IN/IN/--	IN/IN/--		IN/IN/--	
アクセス方式: 据え置きデータ・ページ・アクセス (他の述部 (sargs、resids) でのブロック索引スキャン後)					
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

関連概念:

- 55 ページの『ロックとパフォーマンス』

関連資料:

- 66 ページの『ロック・タイプの互換性』
- 68 ページの『標準の表用のロック・モードおよびアクセス・パス』
- 70 ページの『MDC 表の表および RID 索引スキャンのロック・モード』

ロッキングに影響を与える要因

以下の要素は、データベース・マネージャーのロックのモードおよび細分性に影響を与えません。

- アプリケーションが実行する処理のタイプ
- データ・アクセス方式
- 索引がタイプ 2 またはタイプ 1 であるかどうか
- 様々な構成パラメーター

関連概念:

- 51 ページの『ロックおよび並行性の制御』
- 53 ページの『ロック属性』
- 55 ページの『ロックとパフォーマンス』
- 60 ページの『ロッキングのガイドライン』
- 285 ページの『索引のクリーン・アップおよび保守』
- 77 ページの『アプリケーション・プロセスのロックとタイプ』
- 78 ページの『ロックとデータ・アクセス方式』
- 79 ページの『索引タイプとネクスト・キー・ロッキング』

ロッキングに影響を与える要因

アプリケーション・プロセスのロックとタイプ

ロックの属性を決めるためにアプリケーション処理を次のタイプのいずれかに分類することができます。

- 読み取り専用

  このタイプには、本質的に読み取り専用である SELECT ステートメント、明示的な FOR READ ONLY 文節を含む SELECT ステートメント、あるいは、明示されていないものの、 PREP コマンドまたは BIND コマンドで指定された BLOCKING オプションの値に基づいて SQL コンパイラーが読み取り専用と見なす SELECT ステートメントがすべて含まれます。この処理タイプは、共用ロック (S、NS、または IS) のみを必要とします。

- 変更を意図

  このタイプには、FOR UPDATE 文節、USE AND KEEP UPDATE LOCKS 文節、 USE AND KEEP EXCLUSIVE LOCKS 文節を使用する SELECT ステートメント、または (暗黙に変更が意図された) 未確定ステートメントとして SQL コンパイラーに解釈される SELECT ステートメントがすべて含まれます。このタイプは、共用および更新ロック (行では S、U、X、ブロックでは IX、U、X、S、表では IX、U、X) を使用します。

- 変更

  このタイプには UPDATE、INSERT、および DELETE が含まれますが、UPDATE WHERE CURRENT OF または DELETE WHERE CURRENT OF は含まれません。このタイプには排他ロック (X または IX) が必要です。

- カーソル制御

  このタイプには UPDATE WHERE CURRENT OF および DELETE WHERE CURRENT OF が含まれます。これにも排他ロック (X または IX) が必要です。

副選択ステートメントの結果に基づいてターゲット表にデータを挿入、更新、または削除するステートメントは、2 種類の処理を行います。副選択ステートメントで戻される表のロックは、読み取り専用処理の規則によって決定されます。ターゲット表のロックは、変更処理の規則によって決定されます。

関連概念:

- 51 ページの『ロックおよび並行性の制御』
- 53 ページの『ロック属性』
- 55 ページの『ロックとパフォーマンス』
- 60 ページの『ロッキングのガイドライン』
- 11 ページの『アプリケーション間のデッドロック』
- 78 ページの『ロックとデータ・アクセス方式』
- 79 ページの『索引タイプとネクスト・キー・ロッキング』

関連タスク:

- 62 ページの『ロック・エスカレーション問題の修正』

関連資料:

- 66 ページの『ロック・タイプの互換性』

ロックとデータ・アクセス方式

アクセス・プラン とは、特定の表からデータを取得するためにオプティマイザーが選択する方式です。アクセス・プランは、ロック・モードに大きな影響を与える可能性があります。たとえば、索引スキャンを使ってある特定の行を見つける場合、オプティマイザーは表に関する行レベル・ロッキング (IS) をおそらく選択するでしょう。たとえば、従業員番号 EMPNO に関する索引が EMPLOYEE 表に含まれる場合、 1 人の従業員についての情報を選び出すために、以下の SELECT 文節を含むステートメントを使用し、索引を介してアクセスできます。

```
SELECT *
  FROM EMPLOYEE
  WHERE EMPNO = '000310';
```

索引を使用しない場合には、選択された行を検出するために表全体を順次にスキャンしなければならないので、単一表レベル・ロック (S) を獲得することになります。たとえば、SEX (性別) 列に関する索引が存在しない場合、表のスキャンを使用し、以下のような SELECT 文節を含むステートメントによってすべての男性従業員を選び出すことができます。

```
SELECT *
  FROM EMPLOYEE
  WHERE SEX = 'M';
```

注: カーソル制御される処理の場合、アプリケーションが更新または削除対象の行を見つけるために、基礎となるカーソルのロック・モードが使われます。この種の処理では、カーソルのロック・モードが何であっても、更新や削除を行うときは必ず排他ロックが獲得されます。

レンジ・クラスター表のロックインの作業は、標準キーまたはネクスト・キー・ロッキングの作業とは若干異なります。レンジ・クラスター表内で行範囲にアクセスする場合、範囲指定した行の一部が空であっても、範囲内のすべての行がロックされます。標準キーまたはネクスト・キー・ロッキングの場合、既存のレコードが指定されている行だけがロックされます。

アクセス・プランの種類ごとにどのようなロックが獲得されるかの詳細は、参照表に示されています。

データ・ページの据え置きアクセスでは、行に対するアクセスが 2 つのステップで行われ、それによりロッキングのシナリオがさらに複雑になることを示唆しています。ロック獲得のタイミングおよびロックの持続性は、分離レベルに依存します。反復可能読み取り分離レベルはすべてのロックをトランザクションの終了まで保持するため、最初のステップで獲得したロックが保持され、 2 番目のステップでロックをさらに獲得する必要はありません。読み取り固定分離レベルおよびカーソル固定分離レベルでは、 2 番目のステップでロックを獲得する必要があります。 並行性を最大化するには、最初のステップでロックを獲得しないで、修飾行だけが確実に戻されるように、すべての述部を必ず再度適用します。

関連概念:

- 51 ページの『ロックおよび並行性の制御』
- 53 ページの『ロック属性』
- 55 ページの『ロックとパフォーマンス』
- 60 ページの『ロッキングのガイドライン』
- 77 ページの『アプリケーション・プロセスのロックとタイプ』
- 79 ページの『索引タイプとネクスト・キー・ロッキング』

関連タスク:

- 62 ページの『ロック・エスカレーション問題の修正』

関連資料:

- 66 ページの『ロック・タイプの互換性』
- 68 ページの『標準の表用のロック・モードおよびアクセス・パス』
- 70 ページの『MDC 表の表および RID 索引スキャンのロック・モード』
- 74 ページの『MDC 表のブロック索引スキャンのロック』

索引タイプとネクスト・キー・ロッキング

トランザクションによってタイプ 1 索引の内容が変更されるため、ある程度のネクスト・キー・ロッキングが発生します。タイプ 2 索引の場合、ネクスト・キー・ロッキングの発生は最低限に抑えられます。

- タイプ 2 索引のネクスト・キー・ロッキング

  ネクスト・キー・ロッキングは、索引にキーが挿入されるときに発生します。

  キーが索引に挿入されるとき、索引内で新しいキーの次のキーに対応する行が RR 索引スキャンによって現在ロックされている場合にのみ、その行がロックされます。ネクスト・キーロックで使用されるロック・モードは NW です。ネクスト・キーロックは、キー挿入が実際に行われる前に解放されます。キー挿入は、表に行が挿入されるときに発生します。

  さらに、行の更新によってその行の索引キー値が変更された場合にも、元のキー値が削除済みとマークされ、新しいキー値が索引に挿入されるため、キー挿入が発生します。索引の組み込み列だけに影響を与える更新の場合、キーをインプレースで更新することができ、ネクスト・キー・ロッキングは発生しません。

  RR スキャンの際、スキャン範囲の終わりの次に来るキーに対応する行は、 S モードでロックされます。スキャン範囲の後にキーが存在しない場合には、索引の末尾をロックするために、表末ロックが獲得されます。スキャン範囲の後に存在するキーが削除済みとマークされている場合には、削除済みとマークされていないキーが見つかるまで、対応する行のロックが保持されます (その後、見つかったキーに対応する行がロックされます)。または、索引の末尾がロックされるまで、行のロックが保持されます。
- タイプ 1 索引のネクスト・キー・ロッキング

ネクスト・キー・ロックは、索引での挿入や削除の際、および索引スキャンの際に発生します。表の行が更新、削除、または挿入されるとき、その行に関して X ロックが獲得されます。挿入の場合は、W ロックにダウングレードされる場合もあります。

表索引においてキーが削除または挿入されるとき、索引内の削除または挿入対象のキーの次に来るキーに対応する行がロックされます。キー値に影響を与える更新の場合、元のキー値がまず削除された後、新しい値が挿入されるため、2 つのネクスト・キーのロックが獲得されます。これらのロックの存続期間は、次のように決定されます。

- 索引キーの削除の場合、ネクスト・キーのロック・モードは X で、ロックはコミット時まで保持されます。
- 索引キーの挿入の場合、ネクスト・キーのロック・モードは NW です。このロックは、ロックの競合がある場合にのみ獲得されます。その場合、キーが実際に索引に挿入される前に、ロックが解放されます。
- RR スキャンの場合、索引スキャン範囲の終わりの次に来るキーに対応する表行は、 S モードでロックされ、コミット時まで保持されます。
- CS/RS スキャンの場合、索引スキャン範囲の終わりの次に来るキーに対応する行は、 NS モードでロックされます (ロックの競合が存在する場合)。このロックは、スキャン範囲の末尾が検証されたときに解放されます。

タイプ 1 索引で挿入や削除が行われるとき、ネクスト・キー・ロッキングによってデッドロックが発生する可能性もあります。以下の例は、2 つのトランザクションによってデッドロックが発生する例を示しています。タイプ 2 索引の場合、このようなデッドロックは発生しません。

以下の例では、 6 つの行に関する値 1 5 6 7 8 12 を持つ索引について考えます。

1. トランザクション 1 はキー値 8 の行を削除します。値 8 の行が X モードでロックされます。対応するキーが索引から削除されるとき、値 12 の行が X モードでロックされます。
2. トランザクション 2 はキー値 5 の行を削除します。値 5 の行が X モードでロックされます。対応するキーが索引から削除されるとき、値 6 の行が X モードでロックされます。
3. トランザクション 1 が、キー値 4 の行を挿入します。この行は W モードでロックされます。新しいキーを索引に挿入しようとすると、値 6 の行が NW モードですでにロックされています。このロックは、トランザクション 2 がこの行に関して獲得している X ロックを待機しようとします。
4. トランザクション 2 が、キー値 9 の行を挿入します。この行は W モードでロックされます。新しいキーを索引に挿入しようとすると、キー値 12 の行が NW モードですでにロックされています。このロックは、トランザクション 1 がこの行に関して獲得している X ロックを待機しようとします。

タイプ 1 索引を使用する場合、このようにしてデッドロックが発生し、いずれかのトランザクションがロールバックされます。

関連概念:

- 277 ページの『索引の利点と欠点』

- 282 ページの『索引のパフォーマンスのヒント』
- 26 ページの『索引の構造』
- 287 ページの『索引の再編成』
- 289 ページの『オンライン索引のデフラグ』
- 285 ページの『索引のクリーン・アップおよび保守』

最適化の要因

このセクションでは、照会の最適化クラスを指定するときに考慮すべき要因について説明します。

最適化クラスのガイドライン

SQL 照会をコンパイルするときは、オプティマイザーによる、その照会のための最も効率的なアクセス・プランの選択方法を決定する最適化クラスを指定できます。個別に最適化手法を指定して照会の実行時のパフォーマンスを向上することはできますが、指定する最適化手法が多いほど、照会のコンパイルに要する時間とシステム・リソースは多くなります。

注: フェデレーテッド・データベースの照会では、リモート・オプティマイザーに最適化クラスは適用されません。

最適化クラスを設定することには、特に以下の場合に最適化手法を明示的に指定できるという益があります。

- 非常に小さなデータベースまたは非常に単純な動的照会を管理する
- コンパイル時のデータベース・サーバー上のメモリー制限に対処する
- PREPARE などの照会のコンパイル時間を削減する

ほとんどのステートメントは、妥当な量のリソースの場合、デフォルトの照会最適化クラスである最適化クラス 5 を使用することによって十分に最適化できます。特定の最適化クラスでの照会コンパイル時間とリソース消費は、主として、照会の複雑度、特に結合と副照会の数によって影響を受けます。しかし、コンパイル時間とリソースの使用量も、実行される最適化の数によって影響を受けます。

照会最適化クラス 1、2、3、5、および 7 はすべて、汎用に適しています。クラス 0 の使用は、照会のコンパイル時間をさらに削減する必要があり、 SQL ステートメントが非常に単純な場合にのみ考慮してください。

ヒント: 長い時間がかかる照会を分析するには、その照会を db2batch を使用して実行し、コンパイルに使われる時間と実行に使われる時間を判別してください。コンパイルにより多くの時間が必要な場合は、最適化のクラスを下げてください。実行により多くの時間が必要な場合は、より高い最適化クラスの使用を考慮してください。

最適化クラスを選択する際は、以下の一般指針を考慮してください。

- 始めは、デフォルトの照会最適化クラスのクラス 5 を使用してみます。
- デフォルト以外のクラスを使用する場合は、まず 1、2、3 のいずれかを試す。クラス 0、1、および 2 は、貪欲型結合列挙アルゴリズムを使用します。

- 同じ列上にたくさんの結合述部を持つ表が多数ある場合で、コンパイル時間に制約があるという場合には、最適化クラス 1 または 2 を使用します。
- 実行時間が 1 秒未満の非常に短い照会には、低い最適化クラス (0 または 1) を使用します。この種の照会は、以下の特性をもつ傾向があります。
  - 一つか二つの表だけにアクセスします。
  - 単一の行または少しの行だけを取り出す。
  - 完全修飾したユニーク索引を使用します。

  オンライン・トランザクション処理 (OLTP) のトランザクションは、この種の SQL の良い例です。
- 30 秒以上かかる実行時間の長い照会には、高い最適化クラス (3、5、または 7) を使用します。
- クラス 3 以上は、動的プログラミング結合列挙アルゴリズムを使用します。このアルゴリズムではより多くの代替プランを考慮に入れようとするので、特に表の数が増えるにつれて、クラス 0、1、2 に比べてコンパイル時間が大幅に長くなる可能性があります。
- 最適化クラス 9 は、照会に対する特別な最適化要件がある場合にのみ使用します。

複雑な照会では、最適なアクセス・プランを選択するのにさまざまな量の最適化が必要になる場合があります。以下の特性を示す照会には、より高度の最適化クラスを使用してください。
- 大きい表へのアクセス
- 述部の数が多い
- 副照会が多い
- 結合が多い
- UNION や INTERSECT などの集合オペレーターが多い
- 修飾行が多い
- GROUP BY および HAVING 操作
- ネストした表式
- ビューの数が多い

完全に正規化されたデータベースに対する意思決定支援照会または月末報告照会のようなものは、少なくともデフォルト照会最適化クラスが使用される複合照会の良い例です。

照会生成プログラムによって生成された SQL には、もっと高度な照会最適化クラスを使用してください。多くの照会生成プログラムは非効率な SQL を生成します。照会生成プログラムによって生成されたものも含めて、適切に作成されていない照会の場合は、さらに最適化を行って良好なアクセス・プランを選択しなければなりません。照会最適化クラス 2 以上を使用すると、このような SQL 照会でも改善することができます。

関連概念:
- 153 ページの『照会の最適化に影響を与える構成パラメーター』
- 347 ページの『ベンチマーク・テスト』
- 193 ページの『パーティション内並列処理の最適化ストラテジー』
- 195 ページの『MDC 表の最適化ストラテジー』

関連タスク:

- 85 ページの『最適化クラスの設定』

関連資料:

- 83 ページの『最適化クラス』

最適化クラス

SQL 照会をコンパイルするときは、以下のいずれかの最適化クラスを指定できます。

0 - このクラスは、オプティマイザーが最小限の最適化を使ってアクセス・プランを生成するよう指示します。この最適化クラスには、次の特性があります。

- オプティマイザーは、非均一分散統計を考慮しない。
- 基本的な照会書き直し規則のみを適用します。
- 貪欲型結合列挙を行う。
- ネスト・ループ結合および索引スキャン・アクセス方式のみが使用可能になります。
- リスト・プリフェッチおよび索引 ANDing 結合を、生成されたアクセス方式で使用されないようにします。
- スター型結合方式は考慮に入れない。

このクラスを使用するのは、照会コンパイル・オーバーヘッドを最小限にすることが必要な環境の場合だけにしてください。適切に索引付けされた表にアクセスする非常に単純な動的 SQL ステートメントだけで構成されるアプリケーションでは、照会最適化クラス 0 が適しています。

1 - この最適化クラスには、次の特性があります。

- オプティマイザーは、非均一分散統計を考慮しない。
- 照会書き直し規則のサブセットのみを適用します。
- 貪欲型結合列挙を行う。
- 索引 ANDing 結合はスター型結合に使用される半結合を処理するときにまだ使用するが、生成されたアクセス方式でリスト・プリフェッチと索引 ANDing 結合は使用しない。

最適化クラス 1 は、マージ・スキャン結合と表スキャンも使用可能である点を除けば、クラス 0 と同じ働きをします。

2 - このクラスは、最適化をクラス 1 よりも大幅に向上させ、複雑な照会の場合のコンパイル・コストをクラス 3 以上に比べてはるかに低く抑えるような最適化を使用するよう、オプティマイザーに指示します。この最適化クラスには、次の特性があります。

- 使用可能な統計すべて (頻度と変位値の両方の非均一分散統計を含む) を使用します。
- 非常にまれな場合にしか適用できない、計算を多用する規則を除き、すべての照会再作成規則を適用します。これには、マテリアライズ照会表に対する照会の経路指定が含まれます。
- 貪欲型結合列挙を使用します。
- リスト・プリフェッチおよびマテリアライズ照会表の経路指定を含む広い範囲のアクセス方式が考慮されます。
- 該当する場合には、スター型結合方式が考慮されます。

最適化クラス 2 は、動的プログラミングではなく貪欲型結合列挙を使用する点を除けば、クラス 5 と同様な働きをします。このクラスは、貪欲型結合列挙アルゴリズムを使用するクラスの中では最も高度な最適化であり、複雑な照会の場合にあまり代替プランを考慮しないので、クラス 3 以上と比べてコンパイル時間は少なくて済みます。意思決定支援またはオンライン分析処理 (OLAP) 環境において非常に複雑な照会を行う場合には、クラス 2 をお勧めします。このような環境では、特定の照会が繰り返されることはめったにないので、その照会が次に行われるまでそのアクセス・プランがキャッシュに残っていることはほとんどありません。

3 - このクラスは、適度の最適化を要求します。このクラスは、DB2 for MVS/ESA OS/390 および z/OS の照会最適化の特性に最も近いものです。この最適化クラスには、次の特性があります。

- 使用可能であれば、頻繁に発生する値の追跡を行う非均一分散統計を使用します。
- 「副照会から結合への変換」を含めて、ほとんどの照会書き直し規則を適用します。
- 動的プログラミング結合列挙
  - 複合内部表の限定使用
  - 参照表に関係するスタースキーマに対するカルテシアン積の限定使用
- リスト・プリフェッチ、索引 ANDing 結合、スター型結合を含めた広範囲のアクセス方式が考慮されます。

このクラスは、広い範囲のアプリケーションに適しています。このクラスは、4 つ以上の結合を含む照会のアクセス・プランを改善します。しかし、オプティマイザーがより良いプランを考慮するのに失敗し、デフォルトの最適化クラスを選択することもあります。

5 - このクラスは、オプティマイザーが大幅な最適化を使ってアクセス・プランを生成するよう指示します。 この最適化クラスには、次の特性があります。

- 使用可能な統計すべて (頻度と変位値の分散統計を含む) を使用します。
- 照会のマテリアライズ照会表への経路指定を含めて、すべての照会書き直し規則を適用する (ただし、まれなケースにしか適用されない計算量が多い規則を除く)。
- 動的プログラミング結合列挙
  - 複合内部表の限定使用
  - 参照表に関係するスタースキーマに対するカルテシアン積の限定使用
- リスト・プリフェッチ、索引の AND 結合、およびマテリアライズ照会表経の経路指定を含めた広範囲のアクセス方式が考慮されます。

オプティマイザーが、複合動的 SQL 照会に追加のリソースおよび処理時間が保証されないことを検出すると、最適化は縮小されます。縮小のエクステントつまりサイズは、マシンのサイズと述部の数によって決まります。

照会オプティマイザーが照会最適化の量を縮小すると、通常は適用される照会書き直し規則をすべて適用し続けます。しかし、照会オプティマイザーは貪欲型結合列挙方法を使用するため、考慮されるアクセス・プランの組み合わせの数が少なくなります。

照会最適化クラス 5 は、トランザクションと複合的な照会の両方で構成される混合環境に適した選択です。この最適化クラスは、最も価値のある照会変換技法およびその他の照会最適化技法を、効率的な方法で適用させるよう設計されています。

7 - このクラスは、オプティマイザーが大幅な最適化を使ってアクセス・プランを生成するよう指示します。 複合動的 SQL 照会の照会最適化の量を縮小しないことを除けば、照会最適化クラス 5 と同じです。

9 - このクラスは、オプティマイザーが使用可能なすべての最適化技法を使用するよう指示します。それには、次のものが含まれます。

- すべての使用可能な統計
- すべての照会書き直し規則
- デカルト積および無制限の複合内部を含めて、結合列挙で可能なものすべて。
- すべてのアクセス方式

このクラスでは、オプティマイザーによって考慮される可能なアクセス・プランの数が大幅に拡張されます。このクラスは、より包括的な最適化によって、大型の表を使用する非常に複雑かつ非常に長時間実行する照会に適したアクセス・プランを生成できるかどうかを調べるために使用します。よりすぐれたプランが見つかったかどうかを調べるには、 Explain とパフォーマンスの測定値を使用します。

関連概念:

- 81 ページの『最適化クラスのガイドライン』
- 193 ページの『パーティション内並列処理の最適化ストラテジー』
- 206 ページの『フェデレーテッド・データベースにおけるリモート SQL 生成とグローバル最適化』
- 195 ページの『MDC 表の最適化ストラテジー』

関連タスク:

- 85 ページの『最適化クラスの設定』

最適化クラスの設定

最適化レベルを指定するときは、照会が静的または動的 SQL を使用しているか、また、同一の動的 SQL が繰り返し実行されるかどうかを考慮してください。静的 SQL の場合、照会コンパイル時間とリソースは 1 回だけ費やされ、その結果としての計画は大量の時間を使用する可能性があります。一般に、静的 SQL は常にデフォルトの照会最適化クラスを使用します。動的ステートメントはランタイムにバインドされ実行されるので、動的ステートメントのための追加の最適化のオーバーヘッドが生じても総合的なパフォーマンスが向上するのかどうかということを検討してください。ただし、同じ動的 SQL ステートメントが繰り返し実行される場合には、選択されたアクセス・プランはキャッシュされることになります。このステートメントには、静的 SQL ステートメントと同じ最適化レベルを使用することができます。

最適化を追加することで照会が便利になると考えられるが、その確証がないか、またはコンパイル時間およびリソース使用のことを考慮している場合は、何らかのベンチマーク・テストを実行することができます。

手順:

照会最適化クラスを指定するには、以下のステップに従ってください。

1. 以下のように、非公式に、あるいは正式なテストでパフォーマンス要因を分析します。
   - **動的** SQL ステートメントの場合は、そのテストで、ステートメントの平均実行時間を比較するようにしてください。平均実行時間を見積もるには、以下の公式を使用します。

     ```
     コンパイル時間 + すべての反復の実行時間の合計
     ----------------------------------------------------
                     反復回数
     ```

     この公式で、**反復回数**は SQL ステートメントをコンパイルするたびに SQL ステートメントが実行すると予期されている回数を表します。

     注: 初期コンパイルの後、動的 SQL ステートメントは、環境の変化に伴って再コンパイルが必要になると、再コンパイルされます。 SQL ステートメントがキャッシュされた後で環境が変化しなければ、後続の PREPARE ステートメントはキャッシュされたステートメントを再使用するので、その SQL ステートメントを再度コンパイルする必要はありません。
   - **静的** SQL ステートメントの場合は、ステートメント実行時間を比較するようにしてください。

     静的 SQL のコンパイル時間にも関心があるとしても、ステートメントのコンパイル時間と実行時間の合計を、意味のあるコンテキストで使用するのは難しいことです。合計時間の比較という方法では、静的 SQL ステートメントは 1 回バインドするたびに何度も実行可能であるという事実や、通常はランタイムにはバインドしないという事実は考慮されていません。
2. 最適化クラスを以下のとおりに指定します。
   - **動的** SQL ステートメントでは、 SQL ステートメント SET で設定される CURRENT QUERY OPTIMIZATION 特殊レジスターによって指定された最適化クラスが使用されます。たとえば、次のステートメントは最適化クラス 1 の設定を行います。

     ```
     SET CURRENT QUERY OPTIMIZATION = 1
     ```

     動的 SQL ステートメントが常に同じ最適化クラスを使用するようにするには、この SET ステートメントをアプリケーション・プログラムに組み入れることもできます。

     CURRENT QUERY OPTIMIZATION レジスターが設定されていない場合、動的ステートメントは、デフォルトの照会最適化クラスでバインドされます。動的および静的の両方の SQL のデフォルト値は、データベース構成パラメーター *dft_queryopt* の値によって決まります。クラス 5 はこのパラメーターのデフォルト値です。 BIND オプションおよび特殊レジスターのデフォルト値も、 *dft_queryopt* データベース構成パラメーターから読み取られます。

- **静的** SQL ステートメントでは、 PREP コマンドおよび BIND コマンドに指定される最適化クラスが使用されます。 SYSCAT.PACKAGES カタログ表内の QUERYOPT 列には、パッケージをバインドするのに使われる最適化クラスが記録されています。パッケージが暗黙のうちに、または REBIND PACKAGE コマンドを使って再バインドされる場合、この同じ最適化クラスが静的 SQL ステートメントに使用されます。この静的 SQL ステートメントの最適化クラスを変更するには、 BIND コマンドを使用します。最適化クラスを指定しなかった場合には、 DB2 は *dft_queryopt* データベース構成パラメーターによって指定されたデフォルトの最適化技法を使用します。

関連概念:

- 81 ページの『最適化クラスのガイドライン』

関連資料:

- 83 ページの『最適化クラス』

アプリケーションの調整

このセクションでは、アプリケーションが実行する照会を調整する際の指針を示します。

SELECT ステートメントの制限のガイドライン

オプティマイザーは、アプリケーションが必ず SELECT ステートメントで指定されているすべての行を検索することを想定します。 OLTP およびバッチ環境においては、この想定が最も適しています。しかし、「ブラウズ」アプリケーションにおいては、照会で定義されている結果の集合が大規模であっても、検索するのは最初のいくつかの行だけ、通常は画面をいっぱいにするのに十分な数の行だけであるということがよくあります。

このようなアプリケーションのパフォーマンスを改善するには、以下の方法で SELECT ステートメントを変更します。

- FOR UPDATE 文節を使用して、その後に置く UPDATE ステートメントで更新できる列を指定します。
- 戻される列を読み取り専用にするには、 FOR READ/FETCH ONLY 文節を使用します。
- 全結果セットの中の最初の *n* 行の検索を優先させるには、 OPTIMIZE FOR *n* ROWS 文節を使用します。
- 指定された数の行だけを検索するには、 FETCH FIRST *n* ROWS ONLY 文節を使用します。
- 一度に 1 つずつ行を検索するには、 DECLARE CURSOR WITH HOLD ステートメントを使用します。

注: FOR UPDATE、FETCH FIRST *n* ROWS ONLY 文節、または OPTIMIZE FOR *n* ROWS 文節を使用する場合、あるいはカーソルを SCROLLing として宣言する場合は、行ブロックに影響します。

以下のセクションでは、各方式のパフォーマンス上の利点について説明します。

FOR UPDATE 文節

FOR UPDATE 文節は、その後に置かれる UPDATE ステートメントで更新できる列だけを組み込むことによって、結果セットを制限します。 FOR UPDATE 文節を列名なしで指定する場合は、表またはビューのすべての更新可能な列が含まれることになります。列名を指定する場合、それぞれの名前は修飾されていてはならず、表またはビューの 1 つの列を識別している必要があります。

以下の場合は、FOR UPDATE 文節は使用できません。

- SELECT ステートメントに関連したカーソルを削除できない場合。
- 選択した列のうち少なくとも 1 つが、カタログ表の更新不能な列であって、FOR UPDATE 文節に含まれている場合。

CLI アプリケーションでは、 DB2® CLI 接続属性 SQL_ATTR_ACCESS_MODE を同じ目的に使用してください。

FOR READ または FETCH ONLY 文節

FOR READ ONLY 文節または FOR FETCH ONLY 文節は、戻される結果を読み取り専用にします。読み取り専用として定義されているビューでは SELECT の結果表も読み取り専用なので、この文節は許可されていますが、何の効果もありません。

データベース・マネージャーが排他ロックの代わりにデータのブロックを検索できる場合、更新と削除が許可されている結果表では、 FOR READ ONLY を指定すると、FETCH 操作のパフォーマンスが向上することがあります。指定した UPDATE または DELETE ステートメントで使用されている照会には、 FOR READ ONLY 文節は使用しないでください。

CLI アプリケーションでは、 DB2 CLI 接続属性 SQL_ATTR_ACCESS_MODE を同様の目的に使用することができます。

OPTIMIZE FOR n ROWS 文節

OPTIMIZE FOR 文節は、結果のサブセット 1 つだけを検索するのが目的なのか、または最初の数行だけの検索を優先的に行うのが目的なのかを宣言します。そうすると、オプティマイザーは、最初の数行を検索するための応答時間を最小化するアクセス・プランを優先することができるようになります。さらに、単一ブロックとしてクライアントに送られる行数は、 OPTIMIZE FOR 文節の「n」という値によってバインドされます。したがって、OPTIMIZE FOR 文節はサーバーがデータベースから修飾行を検索する方法と、修飾行をクライアントに戻す方法の両方に影響を与えます。

たとえば、従業員表で基本給が最高の従業員を照会するとします。

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
```

SALARY 列では降順索引を定義しました。しかし、従業員の順序は従業員番号順になっているため、給与索引のクラスター化が不十分であることが考えられます。オプティマイザーは、多数のランダム同期入出力が行われないように、すべての修飾

行の行 ID のソートを必要とするリスト・プリフェッチ・アクセス方式を使用することを選択します。このソートのため、最初の修飾行がアプリケーションに戻される前に遅延が起こります。この遅延を防止するため、以下のようにステートメントに OPTIMIZE FOR 文節を追加してください。

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
OPTIMIZE FOR 20 ROWS
```

この場合、最高給を得ている 20 人の従業員だけが検索されるので、オプティマイザーは、おそらく、SALARY 索引を直接使用することを選択します。ブロック化された行数に関係なく、行のブロックは 20 行ごとにクライアントに戻されます。

OPTIMIZE FOR 文節が指定されている場合、オプティマイザーは、大量データ操作を行わない、あるいはソートなどの行のフローを妨げるアクセス・プランを使用しようとします。 OPTIMIZE FOR 1 ROW を使用すると、アクセス・パスに最も影響を与えることになります。この文節の使用には、以下の効果があります。

- 複合内部表のある結合順序では一時表を必要とするため、使用することは少ない。
- 結合方法を変更できます。 NESTED LOOP 結合は、オーバーヘッド・コストが少なく、通常、少数の行を検索するのにより効果的なので、最もよく選択されます。
- ORDER BY にはソートが必要ないので、 ORDER BY 文節に一致する索引が選ばれやすい。
- リスト・プリフェッチはソートを必要とする方法であるため、このアクセス方式が選ばれることは少ない。
- 少数の行だけが必要であると理解しているので、順次プリフェッチが選ばれることは少ない。
- 結合照会では、外部表の索引が ORDER BY 文節に必要な ORDER BY を指定している場合は、 ORDER BY 文節にある列で成る表が外部表として選ばれやすい。

OPTIMIZE FOR 文節はすべての最適化レベルに適用されますが、 3 より下のクラスは貪欲型結合列挙方法を使用するので、最適化クラス 3 かそれ以上のクラスで最も効果的に作業します。貪欲型結合列挙方法を使用すると、最初の数行の素早い検索には役に立たない複数表結合のアクセス・プランになることがあります。

OPTIMIZE FOR 文節では、修飾行をすべて検索します。すべての修飾行を検索する場合、合計経過時間は、オプティマイザーが応答セット全体を最適化した場合よりも大幅に増える可能性があります。

パッケージ・アプリケーションがコール・レベル・インターフェース (DB2 CLI または ODBC) を使用している場合は、 db2cli.ini 構成ファイルにある OPTIMIZEFORNROWS キーワードを使用して、 DB2 CLI に OPTIMIZE FOR 文節を各照会ステートメントの終了に自動的に付加させることができます。

データがニックネームから選択されるとき、結果はデータ・ソース・サポートによって異なります。ニックネームによって参照されるデータ・ソースが OPTIMIZE FOR 文節をサポートしており、 DB2 オプティマイザーが照会全体をデータ・ソー

スにプッシュダウンする場合、この文節はデータ・ソースに送られたリモート SQL 内で生成されます。データ・ソースでこの文節がサポートされていない場合、またはオプティマイザーが、ローカルな実行が最もコストの低いプランであると判別した場合、 OPTIMIZE FOR 文節はローカルに適用されます。この場合、DB2 オプティマイザーは、照会の最初の数行を検索する応答時間を最小限にするアクセス・プランを優先して選びますが、プランの生成にオプティマイザーが利用できるオプションははっきり限定されず、 OPTIMIZE FOR 文節によるパフォーマンスの向上もほとんどありません。

FETCH FIRST 文節と OPTIMIZE FOR 文節の両方が指定されている場合は、どちらか低い方の値が通信バッファー・サイズに影響します。この 2 つの値は、最適化という目的のために独立して考慮されます。

FETCH FIRST n ROWS ONLY 文節

FETCH FIRST *n* ROWS ONLY 文節は、検索できる最大行数を設定します。 結果表を最初の数行に制限すると、パフォーマンスが向上します。 制限しない場合に結果セットに含まれる行数にかかわりなく、 *n* 行だけが検索されます。

FETCH FIRST 文節と OPTIMIZE FOR 文節の両方が指定されている場合は、どちらか低い方の値が通信バッファー・サイズに影響します。最適化という目的のため、この 2 つの値は互いに独立しています。

DECLARE CURSOR WITH HOLD ステートメント

WITH HOLD 文節を含む DECLARE CURSOR ステートメントを指定してカーソルを宣言すると、トランザクションがコミットされるときに、オープン・カーソルは開いたままの状態になり、オープン WITH HOLD カーソルの現行のカーソル位置を保護しているロック以外のロックはすべて解放されます。

トランザクションがロールバックされると、オープン・カーソルはすべてクローズされ、すべてのロックが解放されて LOB ロケーターが解放されます。

DB2 CLI 接続属性 SQL_ATTR_CURSOR_HOLD を CLI アプリケーションで使用すると、同じ結果を得ることができます。コール・レベル・インターフェース (DB2 CLI または ODBC) を使用するパッケージ・アプリケーションがある場合は、`db2cli.ini` 構成ファイルにある CURSORHOLD キーワードを用いて、 DB2 CLI にすべての宣言されているカーソルについて WITH HOLD 文節が指定されているものと自動的に想定させてください。

関連概念:

- 92 ページの『照会チューニングのガイドライン』
- 94 ページの『効率的な SELECT ステートメント』

オーバーヘッド削減のための行ブロッキングの指定

行のブロッキングは、単一の操作で複数の行からなるブロック を取り出し、カーソルに対するデータベース・マネージャーのオーバーヘッドを削減します。

注: 指定する行のブロックは、メモリー内のページ数になります。これは、ディスク上のエクステントに物理的にマップされるマルチディメンション (MDC) 表ブロックではありません。

行ブロッキング・レベルは、 BIND または PREP コマンドの以下の引き数によって指定されます。

UNAMBIG
読み取り専用カーソルと「FOR UPDATE OF」と指定されていないカーソルの場合に、ブロッキングが行われます。未確定のカーソルは、更新可能として扱われます。

ALL 読み取り専用カーソルと「FOR UPDATE OF」と指定されていないカーソルの場合に、ブロッキングが行われます。未確定のカーソルは、読み取り専用として扱われます。

NO どのカーソルの場合もブロッキングは行われない。未確定のカーソルは、読み取り専用として扱われます。

前提条件:

2 つのデータベース・マネージャー構成パラメーターは、正しく設定する必要があります。値は両方とも、メモリーのページ数として設定されます。これらのパラメーターの値は、ブロック・サイズの計算に使用できるよう、記録して置いてください。

- データベース・マネージャー構成パラメーター *aslheapsz* は、ローカル・アプリケーションのアプリケーション・サポート層ヒープ・サイズを示します。
- データベース・マネージャー構成パラメーター *rqrioblk* は、リモート・アプリケーションとデータベース・サーバー上のデータベース・エージェントとの間に置かれる通信バッファーのサイズを指定します。

手順:

行ブロッキングを指定するには、次のようにします。

1. *aslheapsz* および *rqrioblk* 構成パラメーターの値を使用して、各ブロックに対して戻される行数を見積もる。どちらの公式でも、 *orl* は出力行の長さ (バイト単位) です。
   - ローカル・アプリケーションには以下の公式を使用します。

     ```
     Rows per block = aslheapsz * 4096 / orl
     ```

     1 ページあたりのバイト数は、4 096 です。
   - リモート・アプリケーションには以下の公式を使用します。

     ```
     Rows per block = rqrioblk / orl
     ```
2. 行ブロッキングを使用可能にするため、 PREP または BIND コマンドの BLOCKING オプションに適切な引き数を指定します。

   BLOCKING オプションを指定しない場合、デフォルトの行ブロッキング・タイプは UNAMBIG です。コマンド行プロセッサーおよびコール・レベル・インターフェースの場合、デフォルトの行ブロッキングは ALL です。

注: SELECT ステートメントの中で、FETCH FIRST *n* ROWS ONLY 文節、または OPTIMIZE FOR *n* ROWS 文節を使用した場合、 1 ブロック当たりの行数は、以下の 2 つの値のうち小さい方になります。
- 上記の公式で計算される値
- FETCH FIRST 文節の *n* の値
- OPTIMIZE FOR 文節の *n* の値

関連資料:

- 413 ページの『aslheapsz - アプリケーション・サポート層ヒープ・サイズ』
- 416 ページの『rqrioblk - クライアント入出力ブロック・サイズ』

照会チューニングのガイドライン

照会チューニングの指針に従って、アプリケーション・プログラムの SQL ステートメントを細かく調整してください。この指針は、使用するシステム・リソースと、大規模な表および複雑な照会から値を戻すために必要な時間を最小限にとどめることを目的としています。

注: オプティマイザーが使用する最適化クラスにより、 SQL コンパイラーが SQL コードをより効率的な形式に再書き込みできるために、細かなチューニングは不要であることもあります。

オプティマイザーによるアクセス・プランの選択は、他の要因 (環境についての考慮事項やシステム・カタログ統計を含む) からも影響を受けるということにご注意ください。アプリケーションのパフォーマンスについてベンチマーク・テストを行えば、どのような調整によってアクセス・プランが改善できるかを知ることができます。

関連概念:

- 87 ページの『SELECT ステートメントの制限のガイドライン』
- 94 ページの『効率的な SELECT ステートメント』
- 96 ページの『コンパウンド SQL のガイドライン』
- 97 ページの『文字変換のガイドライン』
- 98 ページの『ストアード・プロシージャーのガイドライン』
- 99 ページの『アプリケーションの並列処理』
- 268 ページの『ソート・パフォーマンスのガイドライン』

関連タスク:

- 90 ページの『オーバーヘッド削減のための行ブロッキングの指定』

SQL 照会でのデータ・サンプリング

データベースは非常に大きくなり、これらのデータベースに対する照会も非常に複雑になっているので、 1 つの照会に関連したすべてのデータを検索することは実際的でない (あるいは、不必要な) 場合があります。ユーザーは全体的な傾向やパターンだけに関心があるかもしれません。この場合、一定の誤差の範囲内でおおまかな応答を出すだけで十分です。このような照会の速度を上げる 1 つの方法は、データベースの無作為標本 (ランダム・サンプル) に対して照会を実行することです。

DB2® では、SQL 照会によって効率的にデータ・サンプリングを実行できます。これによって、高度の正確さを維持しながら、非常に大きな照会のパフォーマンスを何十倍も改善することができます。

サンプリングの最も一般的な適用分野は、AVG、SUM、 COUNT などの集約照会です。この場合、データのサンプルからある程度正確な集約結果が得られます。監査のために表の実際の行からランダム・サブセットを取得する際にも、またデータ・マイニングおよび分析の操作の速度を上げるためにも、サンプリングを使用できます。

DB2 では、行レベルおよびブロック・レベルの 2 つのサンプリング方式が提供されています。

行レベルのベルヌーイ・サンプリング:

行レベルのベルヌーイ・サンプリングは、表の行の P パーセントのサンプルを取ります。その際、各行を P/100 の確率でサンプルに組み込み、 1-P/100 の確率で除外する SARGable 述部 (SARGable 述部) を使用します。

行レベルのベルヌーイ・サンプリングによって、データ・クラスタリングにかかわらず、常に有効な無作為標本が得られます。ただし、索引を使用できない場合、この種のサンプリングのパフォーマンスは低くなる可能性があります。すべての行を検索して、サンプリング述部を適用する必要があるためです。索引が存在しない場合、サンプリングしない照会の実行に比べて、 I/O はまったく節約されません。索引が使用される場合には、索引リーフ・ページ内の RIDS に対してサンプリング述部が適用されるため、この種のサンプリングのパフォーマンスは改善します。これには、通常、選択された RID ごとに 1 つの I/O、および索引リーフ・ページごとに 1 つの I/O が必要です。

システム・ページ・レベルのサンプリング:

システム・ページ・レベルのサンプリングは、行ではなくページのサンプルを取るという点を除いて、行レベルのサンプリングと同じです。あるページがサンプルに組み込まれるかどうかは、P/100 の確率で決定されます。ページが組み込まれる場合、そのページに含まれるすべての行が組み込まれます。

サンプルに組み込まれるページごとに 1 つの I/O だけが必要であるため、システム・ページ・レベルのサンプリングは非常に高いパフォーマンスを実現します。サンプリングしない場合に比べて、ページ・レベルのサンプリングはパフォーマンスを数十倍も改善します。ただし、集約見積もりの正確さは、行レベルのサンプリングに比べてページ・レベルのサンプリングの方が悪いという傾向があります。ブロックあたりの行数が多い場合や、ページ内で高いレベルのクラスター化が行われている列を照会が参照している場合には、その格差が最も顕著です。

特定のタスクに最適のサンプリング方式は、ユーザーの時間制約および求められている正確さの多重度によって決まります。

サンプリング方式の指定:

表からのデータの無作為標本に対して照会を実行するには、 SQL ステートメントの表参照文節で TABLESAMPLE 文節を使用できます。サンプリング方式を指定するには、キーワード BERNOULLI または SYSTEM を使用します。

キーワード BERNOULLI は、行レベルのベルヌーイ・サンプリングが実行されることを指定します。

キーワード SYSTEM は、システム・ページ・レベルのサンプリングが実行されることを指定します。ただし、行レベルのベルヌーイ・サンプリングの方が効率的だとオプティマイザーが判断した場合には、そちらが実行されます。

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『副選択』

効率的な SELECT ステートメント

SQL は柔軟な高水準言語なので、複数の異なる SELECT ステートメントを作成して、同じデータを検索することができます。しかし、ステートメントの形式が異なり、最適化のクラスが異なると、パフォーマンスが変化する可能性があります。

SELECT ステートメントについては、以下の指針を考慮してください。

- 必要な列だけを指定します。 1 つのアスタリスク (*) を使ってすべての列を指定するほうが簡単だとしても、不必要な処理がなされたり不要な列が戻されたりする可能性があります。
- 応答セットを必要な行のみに制限する述部を使用します。
- 必要な行数が戻される行数の合計よりかなり少なくなる場合には、 OPTIMIZE FOR を指定します。この文節は、アクセス・プランの選択と通信バッファーでブロック化される行数の両方に影響します。
- 検出する行数が少ない場合には、 OPTIMIZE FOR *k* ROWS 文節だけを指定します。 FETCH FIRST n ROWS ONLY 文節は不要です。ただし、*n* の値が大きいのではじめの *k* 行を取り出してから、後でまた *k* 行を取り出したい場合には、両方の文節を指定してください。通信バッファーのサイズとして *n* と *k* の中で小さいほうが使われます。以下の例は、両方の文節を表しています。

  ```
  SELECT EMPNAME, SALARY FROM EMPLOYEE
     ORDER BY SALARY DESC
     FETCH FIRST 100 ROWS ONLY
     OPTIMIZE FOR 20 ROWS
  ```

- 行ブロッキングを利用するには、 FOR READ ONLY または FOR FETCH ONLY 文節を指定してパフォーマンスを改善します。検出された行には排他ロックがかけられないので、これに伴って並行性も向上します。追加の照会の再書き込みも行えます。 BLOCKING ALL BIND オプションと一緒に FOR READ ONLY または FOR FETCH ONLY 文節を指定するなら、フェデレーテッド・システム内のニックネームに対する照会のパフォーマンスも同様に向上します。
- 位置指定の更新を使用して更新するカーソルに対しては、 FOR UPDATE OF 文節を指定して、データベース・マネージャー が初めにより適切なロッキング・レベルを選択して、発生する可能性のあるデッドロックを回避できるようにします。 FOR UPDATE カーソルは、行ブロッキングの利点は生かせません。

- 検索済み更新を使用して更新するカーソルに対しては、 FOR READ ONLY および USE AND KEEP UPDATE LOCKS 文節を使用して影響を受ける行を強制的に U ロックすると、デッドロックを回避し、行ブロッキングを引き続き使用できます。
- 可能な限り、数値データ・タイプの変換はしないようにします。値を比較するとき、同じデータ・タイプの項目を使うようにするなら、さらに効率的です。変換が必要な場合、精度の低さのために正確でなくなったり、ランタイム変換のためにパフォーマンスが低下したりする可能性があります。

  可能なら、以下のデータ・タイプを使用してください。
  - 短い列では、VARCHAR 型ではなく、CHAR 型
  - 浮動小数点数や 10 進数ではなく、整数
  - 文字列型ではなく、日時
  - 文字列型ではなく、数値
- ソート操作が発生する可能性を小さくするには、 DISTINCT または ORDER BY などの文節や操作を、必要でなければ省略します。
- 表に行があるかどうか調べるときには、単一の行を選択します。カーソルをオープンして 1 つの行を取り出すか、単一行 (SELECT INTO) 選択を実行してください。複数の行が検出される場合は、SQLCODE -811 のエラーを必ず調べてください。

  表が非常に小さいものであると分かっているのでない限り、以下のステートメントを使用して非ゼロ値を検査することはしないでください。

  ```
  SELECT COUNT(*) FROM  TABLENAME
  ```

  大規模な表の場合、すべての行のカウントを行うとパフォーマンスに影響します。
- 更新活動が低調で表が非常に大きい場合には、述部として頻繁に使用する列に索引を定義します。
- 複数の述部文節に同じ列が存在する場合には、 IN リストを使用することを考慮します。大きい IN リストをホスト変数と共に使用すると、ホスト変数のサブセットのループインでパフォーマンスが向上する可能性があります。

複数の表にアクセスする SELECT ステートメントには、特に以下のことが適用されます。

- 表を結合するには、結合述部を使用します。結合述部とは、1 つの結合において異なる表の 2 つの列を比較することです。
- 結合述部内で列に対して索引を定義し、それによって、その結合をさらに効率的に処理できるようにしてください。索引は、複数の表にアクセスする SELECT ステートメントを含む UPDATE ステートメントおよび DELETE ステートメントにも、効果があります。
- データベース・マネージャーは一部の結合手法を使用できないので、可能なら、結合述部で式または OR 文節を使用しないようにします。結果として最も効率的な結合方式を選択できない場合があります。
- 可能ならば、パーティション・データベース環境で、結合された表が両方とも結合列上でパーティション化されるようにしてください。

関連概念:

- 87 ページの『SELECT ステートメントの制限のガイドライン』
- 92 ページの『照会チューニングのガイドライン』

コンパウンド SQL のガイドライン

データベース・マネージャーのオーバーヘッドを削減するには、いくつかの SQL ステートメントを単一の実行可能ブロックにまとめます。ブロック内の SQL ステートメントは、個別に実行できるサブステートメントなので、この種のコードは *コンパウンド SQL* と呼ばれます。データベース・マネージャーのオーバーヘッドが削減されることに加え、コンパウンド SQL によって、リモート・クライアントでは、ネットワークを介して伝送する必要のある要求の数も削減されます。

コンパウンド SQL には、次の 2 つの種類があります。

- **アトミック**

  アプリケーションは、すべてのサブステートメントが正常完了したとき、または 1 つのサブステートメントがエラー終了したときに、データベース・マネージャーから応答を受信します。 1 つのサブステートメントがエラー終了すると、そのブロック全体がエラー終了したと見なされ、そのブロックの中でのデータベースに対する変更がすべてロールバックされます。

  アトミック・コンパウンド SQL は DB2 Connectではサポートされていません。

- **非アトミック**

  アプリケーションは、すべてのサブステートメントが完了したときにデータベース・マネージャーから応答を受信します。先行するサブステートメントが正常完了したかどうかに関係なく、ブロックの中のすべてのサブステートメントが実行されます。 このグループのステートメントをロールバックできるのは、「非アトミック」コンパウンド SQL を含む作業単位がロールバックされる場合だけです。

コンパウンド SQL は、 DARI ルーチンとしても知られるストアード・プロシージャーと、以下のアプリケーション開発プロセスでサポートされます。

- 組み込み静的 SQL
- DB2 コール・レベル・インターフェース
- JDBC

動的コンパウンド SQL ステートメント

動的コンパウンド・ステートメントは DB2® によって単一ステートメントとしてコンパイルされます。このステートメントは、制御フロー・ロジックはほとんど必要としないが、かなりのデータ・フローを必要とする短いスクリプトに対して効果的に使用することができます。ネストした複雑な制御フローを持つ大きな構成の場合、SQL プロシージャーの使用を検討してください。

動的コンパウンド・ステートメントでは、宣言に以下のエレメントを使用できます。

- サブステートメントの変数宣言の SQL 変数
- 条件宣言の SQLSTATE 値に基づくサブステートメントの条件

- 1 つ以上の SQL プロシージャー・ステートメント

動的コンパウンド・ステートメントは、 FOR ステートメント、IF ステートメント、ITERATE ステートメント、および WHILE ステートメントなどのいくつかのフロー・ロジック・ステートメントを使用することもできます。

動的コンパウンド・ステートメントでエラーが発生した場合、前の SQL ステートメントはすべてロールバックされ、動的コンパウンド・ステートメントの残りの SQL ステートメントは処理されません。

動的コンパウンド・ステートメントはトリガー、SQL 関数、または SQL メソッドに組み込むか、または、動的 SQL ステートメントを通して発行することができます。この実行可能ステートメントは、動的に準備することができます。ステートメントを呼び出す特権は必要ありませんが、そのステートメントに関連する許可 ID はコンパウンド・ステートメント内の SQL ステートメントを呼び出すために必要な特権を持っていなければなりません。

関連概念:

- 92 ページの『照会チューニングのガイドライン』

文字変換のガイドライン

アプリケーションとデータベースが同じコード・ページを使用していない場合、アプリケーションとアプリケーション・コード・ページの間でデータをマップするには、データ変換が必要になる可能性があります。マッピングとデータ変換には、さらに多くのオーバーヘッドが伴うので、アプリケーションとデータベースとが同じコード・ページか、または同じ照合シーケンスを使用していると、アプリケーションのパフォーマンスが向上します。

文字変換は次の場合に行われます。

- クライアントまたはアプリケーションが、アクセスしているデータベースのコード・ページとは別のコード・ページで実行しているとき。

  変換は、データを受信するデータベース・サーバー・マシン上で行われます。データベース・サーバーがデータを受信する場合は、アプリケーション・コード・ページからデータベース・コード・ページへの文字変換が行われます。アプリケーション・マシンがデータを受信する場合は、データベース・コード・ページからアプリケーション・コード・ページへの変換が行われます。
- ファイルをインポートまたはロードするクライアントまたはアプリケーションが、インポートまたはロードされるファイルと異なったコード・ページで実行されるとき。

以下のオブジェクトについては、文字変換は行われません。

- ファイル名。
- FOR BIT DATA 属性を割り当てた列に指定したり、そこから派生したデータ、または SQL 操作によって FOR BIT または BLOB データに変換されるデータ。
- サポートされている EUC または UCS-2 との互換機能がインストールされていない DB2® 製品またはプラットフォーム。この場合、アプリケーションは SQLCODE -332 (SQLSTATE 57017) エラーを受け取ります。

データベース・マネージャーがオペレーティング・システム環境に基づいてマルチバイト・コード・ページを変換するときに使用する変換機能と変換表、あるいは DBCS 変換 API。

注: EUC を用いる DBCS などのマルチバイト・コード・ページの間で文字ストリング変換を行うと、ストリングの長さが大きくなったり、小さくなったりすることがあります。さらに、PC DBCS、EUC、および UCS-2 コード・セット中の別の文字に割り当てられたコード・ポイントは、同じ文字をソートしたときに、別の結果になることがあります。

拡張 UNIX® コード (EUC) のコード・ページ・サポート

C または C++ アプリケーションで GRAPHIC データ (2 バイト文字) を使用するホスト変数には、特殊な考慮事項が必要です。それには、特殊なプリコンパイラー、アプリケーションのパフォーマンス、およびアプリケーション設計に関する事柄が含まれます。

日本語と中国語 (繁体字) の両方の EUC コード・ページの多くの文字には、データベースとクライアント・アプリケーションで GRAPHIC データ (2 バイト文字を必要とする) をサポートするための特別な方法が必要です。それらの EUC コード・ページの GRAPHIC データの保管および操作には、UCS-2 コード・セットを使用します。

関連概念:

- 204 ページの『フェデレーテッド照会を評価する場合の分析のガイドライン』

関連資料:

- 「*管理ガイド: プランニング*」の『コード・ページ 923 および 924 の変換表』
- 「*管理ガイド: プランニング*」の『ユーロを使用可能なコード・ページ遷移表ファイル』

ストアード・プロシージャーのガイドライン

ストアード・プロシージャーでは、データベース・アプリケーション環境において事前プログラムされたプロシージャーを、リモート・データベースに呼び出しを一度出すだけで実行することができます。データベース・アプリケーション環境では、多くの操作は反復的です。たとえば、一定のデータを受信したり、データベースに対して複数の同じ要求の実行したり、一定のデータを戻したりするために、データベースに何度もアクセスしていることがあります。

リモート・データベースのための単一 SQL ステートメントを処理するには、 2 回の送信 (1 回の要求と 1 回の受信) が必要になります。アプリケーションには多数の SQL ステートメントが含まれているので、その作業を完了するには、伝送が何度も必要です。

しかし、データベース・クライアントで多数の SQL ステートメントがカプセル化されたストアード・プロシージャーを使用するなら、プロセス全体で送信は 2 回だけです。

通常、これらのストアード・プロシージャーはデータベース・エージェントとは別のプロセスで実行されます。このように分けているため、ストアード・プロシージャーとエージェント・プロセスとはルーターを介して通信する必要があります。しかし、エージェント・プロセスで実行する特殊なストアード・プロシージャーのパフォーマンスは向上できます。ただし、これにはデータやデータベースが破壊されるという大きなリスクが伴います。

これらのリスクを伴うストアード・プロシージャーは、 *NOT FENCED* として作成されたストアード・プロシージャーです。 NOT FENCED で作成されたストアード・プロシージャーの場合、ストアード・プロシージャーは、データベース・エージェントが使用するデータベース制御構造から分離されていません。 DBA が、データベース制御構造が、偶然であれ故意であれ、このストアード・プロシージャーによって壊されることがないようにしたい場合は、 *NOT FENCED* オプションを省略します。

データベースを破壊する危険があるため、 *NOT FENCED* で作成されたストアード・プロシージャーは、最高のパフォーマンスが必要な場合に**のみ**使用してください。さらに、NOT FENCED で作成されたストアード・プロシージャーとして実行するためには、プロシージャーが正しくコーディングされ、完全にテストされていることを完全に確認する必要があります。これらの、NOT FENCED ストアード・プロシージャーのいずれかを実行中に致命的エラーが発生した場合は、データベース・マネージャーは、このエラーがアプリケーションで発生したのかデータベース・マネージャーのコードで発生したのかを調べ、適切なリカバリー処置を実行します。

NOT FENCED で作成されたストアード・プロシージャーがデータベース・マネージャーを壊してリカバリー不能にし、結果としてデータの消失やデータベースの破壊を招く可能性もあります。 NOT FENCED で作成された信頼できるストアード・プロシージャーを実行するときは、細心の注意を払ってください。たいていの場合は、アプリケーションのパフォーマンス分析を適切に実行すれば、 NOT FENCED で作成されたストアード・プロシージャーを使用しなくても望ましいパフォーマンスを得ることができます。たとえば、トリガーでパフォーマンスを向上させることも可能です。

関連概念:

- 92 ページの『照会チューニングのガイドライン』

アプリケーションの並列処理

DB2® は、主に対象マルチプロセッサー (SMP) マシン上で並列環境をサポートしますが、単一プロセッサー・マシン上でも限られた範囲でサポートします。 SMP マシンでは、データベースに複数のプロセッサーがアクセスでき、複雑な SQL 要求の実行を複数のプロセッサー間で分け合って、並列に実行することができます。

アプリケーションのコンパイル時にインプリメントする並列処理の多重度を指定するには、 CURRENT DEGREE 特殊レジスターまたは DEGREE BIND オプションを使用します。*程度* とは、並行して実行する照会のパーツの数を指します。プロセッサーの数と並列処理の多重度として選択された値との間には、厳密な関係はありません。マシン上のプロセッサーよりも多い数または少ない数を指定することもで

きます。単一プロセッサー・マシンの場合も、 1 より高い程度を設定して何らかの方法でパフォーマンスを改善できます。しかし、並列処理の多重度が高くなれば、システム・メモリーと CPU のオーバーヘッドが増えるという点にご注意ください。

照会の並列実行の使用時にパフォーマンスを最適化するには、いくつかの構成パラメーターを変更する必要があります。特に並列処理の多重度の高い環境では、共用メモリーとプリフェッチの量を制御する構成パラメーターを検討して変更する必要があります。

以下の 3 つの構成パラメーターは、パーティション内並列処理を制御および管理します。

- *intra_parallel* データベース・マネージャー構成パラメーターは、並列サポートをオンにしたりオフにしたりします。
- *max_querydegree* データベース・マネージャー構成パラメーターは、データベースでの照会における並列処理の多重度の上限を設定します。この値は、CURRENT DEGREE 特殊レジスターおよび DEGREE BIND オプションをオーバーライドします。
- *dft_degree* データベース構成パラメーターは、 CURRENT DEGREE 特殊レジスターおよび DEGREE BIND オプションのデフォルト値を設定します。

照会で DEGREE = ANY を指定してコンパイルすると、データベース・マネージャーによってパーティション内並列処理の多重度が選ばれます。この程度は、プロセッサーの数や照会の特性などを示すいくつかの係数に基づいて選ばれます。ですから、係数の値およびシステム上のアクティビティーの量によっては、実際に実行時に使用される多重度の値がプロセッサーの数よりも少ない場合があります。システムが酷使されている場合は、照会の実行のまえに並列処理が低められる可能性があります。なぜなら、パーティション内並列処理は照会にかかる時間を節約するためにシステム・リソースを酷使し、それが他のデータベース・ユーザーのパフォーマンスに悪影響を及ぼすからです。

SQL オプティマイザーによって選択された並列処理の多重度を表示するには、 SQL Explain 機能を使用してアクセス・プランを表示します。実行時に実際に使用されている並列処理の多重度に関する情報を表示するには、データベース・システムのモニターを使用します。

非 SMP 環境での並列処理

SMP マシンがなくても、並列処理の多重度を指定できます。たとえば、単一プロセッサー・マシンで入出力制約の照会を実行する場合でも、 2 度以上を宣言しておいた方が有利です。この場合、プロセッサーは入力または出力タスクの完了を待たずに、次の照会の処理を開始できます。しかし、2 度以上を宣言しても、単一プロセッサー・マシン上の入出力並列処理は制御されません。 Load などのユーティリティーでは、このような宣言とは関係なく、入出力並列処理を制御することができます。キーワード ANY は、 *dft_degree* データベース・マネージャー構成パラメーターを設定するためにも使用できます。オプティマイザーは、ANY キーワードによって、パーティション内並列処理の多重度を判別することができます。

関連概念:

- 214 ページの『Explain ツール』
- 193 ページの『パーティション内並列処理の最適化ストラテジー』

関連資料:

- 525 ページの『max_querydegree - 照会の最大並列処理多重度』
- 525 ページの『intra_parallel - パーティション内並列処理機能の使用可能化』
- 503 ページの『dft_degree - デフォルト多重度』

REOPT を指定してバインディングすることによりパフォーマンスを向上させる

パラメーター・マーカー、ホスト変数、および特殊レジスターなどの入力変数に使用される値が、デフォルトのフィルター係数の推定値の予測範囲外の場合、SQL 照会の実行中のパフォーマンスが不十分になることがあります。デフォルトのフィルター係数は、実データ値が不明なシナリオで使用されるものであり、実行時 (実データ値を使用する) に実際に戻される行数の推定値です。

BIND オプションは、DB2® がホスト変数、パラメーター・マーカー、および特殊レジスターの値を使用して実行時にアクセス・パスを最適化するようにするかどうかを指定します。 REOPT 値は、BIND、PREP、または REBIND コマンドの以下の引き数によって指定されます。

REOPT NONE
ホスト変数、パラメーター・マーカー、または特殊レジスターを含んだ所定の SQL ステートメントのアクセス・パスを、これらの変数の実際の値を使って最適化しません。代わりにこれらの変数のデフォルトの推定値を使用します。このプランをキャッシュに入れて、それ以降使用します。これがデフォルトの動作です。

REOPT ONCE
所定の SQL ステートメントのアクセス・パスを、初めて照会を実行するときに、ホスト変数、パラメーター・マーカー、または特殊レジスターの実際の値を使って最適化します。このプランをキャッシュに入れて、それ以降使用します。

REOPT ALWAYS
所定の SQL ステートメントのアクセス・パスを、毎回の実行時に、ホスト変数、パラメーター・マーカー、または特殊レジスターの値を使って、いつもコンパイルおよび再最適化します。

関連概念:

- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『静的 SQL における REOPT の影響』
- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『動的 SQL における REOPT の影響』

第 4 章 環境についての考慮事項

アプリケーションを設計し、コーディングする際に考慮する要因 (43 ページの『第 3 章 アプリケーションについての考慮事項』で説明されています) に加えて、特定の環境要因も、アプリケーションについて選択されたアクセス・プランに影響を与える可能性があります。

SQL オプティマイザーに直接影響を与える要因については、 109 ページの『第 5 章 システム・カタログ統計』を参照してください。

アプリケーションを調整し、環境を変更する場合は、そのアプリケーションを再バインドして、最適のアクセス・プランが確実に使用されるようにしてください。

照会の最適化に影響を与えるデータベース・パーティション・グループ

パーティション・データベースでは、オプティマイザーは表のコロケーションを認識し、照会に対する最適のアクセス・プランを判別する際にそのコロケーションを使用します。表が頻繁に結合照会に関係する場合、それらの表は、結合される各表にある行が同じデータベース・パーティションにあるように、パーティション・データベース内のパーティション間で分割する必要があります。結合操作を実行するときに、結合される両方の表にあるデータのコロケーションによって、データをあるパーティションから別のパーティションに移動されなくなります。同じデータベース・パーティション・グループに両方の表を置き、表のデータが確実に同じパーティションに入れられるようにしてください。

パーティション・データベースでは、表のサイズによって、データをより多くのパーティションに分散すると、照会の実行にかかる見積時間 (つまりコスト) が減少します。表の数、表のサイズ、それらの表のデータがある場所、および結合が必要かどうかといった照会のタイプはすべて照会のコストに影響を与えます。

関連概念:

- 183 ページの『パーティション・データベースでの結合ストラテジー』
- 185 ページの『パーティション・データベースでの結合方式』
- 322 ページの『パーティション・データベースのパーティション』

表スペースが照会の最適化に与える影響

表スペースの特性のうちのあるものは、 SQL コンパイラーによって選択されるアクセス・プランに影響を与える可能性があります。

- コンテナー特性

  コンテナー特性は、照会の実行時に関連付けられた入出力のコストに重大な影響を与える可能性があります。アクセス・プランを選択するとき、SQL オプティマイザーは、異なる複数の表スペースのデータにアクセスする場合のコストの相違も含めて、それらの入出力コストを考慮に入れます。オプティマイザーが表スペ

ースのデータにアクセスするための入出力コストを見積もるときに、SYSCAT.TABLESPACES システム・カタログの 2 つの列が使用されます。

– OVERHEAD。データがメモリーに読み込まれるまでにコンテナーで必要な時間の見積値 (ミリ秒)。 このオーバーヘッド活動には、ディスク待ち時間以外にも、コンテナーの入出力コントローラーのオーバーヘッド (ディスクのシーク時間を含む) が含まれます。

  以下の公式を使って、オーバーヘッドのコストを見積もることができます。

```
OVERHEAD = ミリ秒単位の平均シーク時間
           + (0.5 * 回転待ち時間)
```

  ここで、
  - 0.5 は半回転した場合の平均オーバーヘッドを示します。
  - 1 回転ごとの回転待ち時間はミリ秒単位で以下のように計算されます。

```
(1 / RPM) * 60 * 1000
```

    ここで、
    • 1 分当たりの回転数で除算し、1 回転当たりの分数を求めます。
    • 60 (1 分間の秒数) で乗算します。
    • 1000 (1 秒間のミリ秒数) で乗算します。

  たとえば、ディスクの 1 分当たりの回転数が 7 200 であるとします。回転待ち時間の公式を使用すると、次のようになります。

```
(1 / 7200) * 60 * 1000 = 8.328 ミリ秒
```

  この値は、想定される 11 ミリ秒の平均シーク時間と共に、次のように OVERHEAD 見積もりの計算に使用することができます。

```
OVERHEAD = 11 + (0.5 * 8.328)
         = 15.164
```

  見積もられた OVERHEAD 値が約 15 ミリ秒となります。

– TRANSFERRATE。1 ページのデータをメモリーに読み込むのに必要な時間の見積値 (ミリ秒)。

  それぞれの表スペース・コンテナーが単一の物理ディスクである場合は、以下の公式を使って、転送コストを 1 ページ当たりのミリ秒数で見積もることができます。

```
TRANSFERRATE = (1 / spec_rate) * 1000 / 1 024 000 * page_size
```

  ここで、
  - spec_rate は、転送速度に関するディスクの仕様を、 1 秒当たりの MB 数で表します。
  - spec_rate で除算し、MB 当たりの秒数を求めます。
  - 1000 (1 秒間のミリ秒数) で乗算します。
  - MB 当たり 1 024 000 バイトで除算します。
  - ページ・サイズ (バイト単位) で乗算します (たとえば、4 KB のページの場合は 4 096 バイト)。

  たとえば、ディスクの指定率が 1 秒当たり 3 MB であるとします。この場合、次の計算が行われます。

```
TRANSFERRATE = (1 / 3) * 1000 / 1024000 * 4096
             = 1.333248
```

見積もられた TRANSFERRATE 値は、ページ当たり約 1.3 ミリ秒になります。

表スペース・コンテナーが単一の物理ディスクではなく、 (RAID などの) ディスク・アレイである場合、使用する TRANSFERRATE を決定しようとするときに追加の考慮事項を考慮する必要があります。アレイが比較的小さい場合は、障害はディスク・レベルにあると想定して、 spec_rate にディスクの数を乗算することができます。

しかし、コンテナーを構成しているアレイ内のディスクの数が多ければ、障害はディスク・レベルではなく、ディスク・コントローラー、入出力バス、またはシステム・バスなどのその他の入出力サブシステム構成装置の 1 つに存在する可能性もあります。この場合、入出力スループット能力は、 spec_rate とディスクの数の積であるとは想定できません。順次スキャン中に、実際の入出力率 MB 単位を測定する必要があります。たとえば、順次スキャンは select count(*) from big_table となり、サイズは MB 単位となります。この数を、big_table が存在している表スペースを構成するコンテナーの数で除算します。上記の公式の spec_rate をこの計算結果で置き換えます。たとえば、 4 つのコンテナー表スペースの中の表をスキャンしている間に測定された 100 MB の順次入出力率は、コンテナー当たり 25 MB となるか、TRANSFERRATE がページ当たり (1/25) * 1000 / 1024000 * 4096 = 0.16 ミリ秒となります。

表スペースに割り当てられたコンテナーは、それぞれ異なる物理ディスク上に存在しています。最適の結果を得るためには、所定の表スペースに使用されるすべての物理ディスクの OVERHEAD および TRANSFERRATE の特性が同じでなければなりません。これらの特性が同じでない場合には、 OVERHEAD および TRANSFERRATE の値を設定するときには平均値を使用する必要があります。

これらの列のメディア特有の値は、ハードウェア仕様から、または実験によって得ることができます。 これらの値は、CREATE TABLESPACE および ALTER TABLESPACE ステートメントで指定できます。

コンテナーとしてディスク・アレイを使用している上記のような環境では、実験は特に重要です。データを移動させる単純な照会を作成して、その照会をプラットフォーム固有の測定ユーティリティーと一緒に使用する必要があります。その後、表スペース内の異なるコンテナー構成についてその照会を再実行します。CREATE および ALTER TABLESPACE ステートメントを使用して、現在の環境でデータが転送された方法を変更することができます。

これら 2 つの値によって提供される入出力コスト情報は、いくつかの方法でオプティマイザーに影響を与える可能性があります。データにアクセスするのに索引を使用するかどうか、または 1 つの結合の中で内部と外部にどの表を選択するか、などがこれに含まれます。

- プリフェッチ

表スペースのデータにアクセスするための入出力コストを考慮するとき、オプティマイザーは、ディスクからデータおよび索引ページをプリフェッチすることによって、照会のパフォーマンスに与える潜在的な影響も考慮します。 データおよび索引ページのプリフェッチを行うと、データをバッファー・プールに読み込むことに関連するオーバーヘッドと待ち時間が少なくなります。

オプティマイザーは SYSCAT.TABLESPACES の PREFETCHSIZE 列および EXTENTSIZE 列の情報を使用して、表スペースに対して生じる取り出しの量を見積もります。

- EXTENTSIZE を設定できるのは、(たとえば CREATE TABLESPACE ステートメントを使って) 表スペースを作成するときだけです。デフォルトのエクステント・サイズは 32 ページ (それぞれが 4 KB) で、普通はこれで十分です。
- PREFETCHSIZE は、表スペースを作成するとき、および/または ALTER TABLESPACE ステートメントを使用するときに設定できます。デフォルトのプリフェッチ・サイズは DFT_PREFETCH_SZ データベース構成パラメーターの値によって決まります。このパラメーターはオペレーティング・システムによって異なります。このパラメーターのサイズ変更に関する推奨を検討して、データの移動を改善するために必要な変更を行ってください。

次に、RESOURCE 表スペースの特性を変更するための構文の例を示します。

```
ALTER TABLESPACE RESOURCE
  PREFETCHSIZE 64
  OVERHEAD     19.3
  TRANSFERRATE 0.9
```

表スペースに対して変更を行った後は、アプリケーションを再バインドすること、および、最適なアクセス・プランが使用されるように、 RUNSTATS ユーティリティーを用いて索引に関する最新の統計を収集することを考慮してください。

関連概念:

- 121 ページの『カタログ統計の表』
- 149 ページの『SQL コンパイラーの処理』
- 18 ページの『DMS 表スペース・アドレス・マップの図』

フェデレーテッド・データベースに影響を与えるサーバー・オプション

フェデレーテッド・システムは、DB2® DBMS (フェデレーテッド・データベース) と 1 つまたは複数のデータ・ソースから構成されています。データ・ソースは、CREATE SERVER ステートメントを発行したときにフェデレーテッド・データベースに識別されます。このステートメント発行の際に、サーバー・オプションを含めることもできます。このサーバー・オプションにより DB2 と指定したデータ・ソースに関係するフェデレーテッド・システムのさまざまな動作を細かく調整、制御することができます。後でサーバー・オプションを変更するには、 ALTER SERVER ステートメントを使用します。

注: サーバーを作成してサーバー・オプションを指定する前に、分散結合のインストール・オプションをインストールし、データベース・マネージャー・パラメーター *federated* を YES に設定する必要があります。

指定するサーバー・オプションの値は、照会のプッシュダウン分析、グローバル最適化、およびフェデレーテッド・データベース操作のその他の局面に影響します。たとえば、 CREATE SERVER ステートメントで、 *cpu_ratio* オプションのように、サーバー・オプションの値としてパフォーマンス統計を指定することができます。これは、データ・ソースおよびフェデレーテッド・サーバーにおける CPU の相対速度を指定します。また、ソースとフェデレーテッド・サーバーのデータ入出力装置の相対速度を表す値を *io_ratio* として設定できます。 CREATE SERVER ステートメントを実行すると、このデータはカタログ・ビュー SYSCAT.SERVEROPTIONS に追加され、オプティマイザーはデータ・ソースのアクセス・プランを立てるときにそのデータを使用します。統計情報が変更された場合 (たとえばデータ・ソース CPU がアップグレードされた場合) などには ALTER SERVER ステートメントを使用して、 SYSCAT.SERVEROPTIONS の設定に変更を反映してください。次回以降、オプティマイザーはデータ・ソースへのアクセス・プランを選択する際に、その新規の情報を使用します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER SERVER ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.SERVEROPTIONS カタログ・ビュー』
- 536 ページの『federated - フェデレーテッド・データベース・システム・サポート』
- 「*SQL リファレンス 第 2 巻*」の『CREATE SEQUENCE ステートメント』

第 5 章 システム・カタログ統計

システム・カタログに保管されている統計データは、オプティマイザーが照会に最適なアクセス・プランを選択する際に役立ちます。次のように RUNSTATS を実行して、この統計データを必ず更新するようにしてください。

- 内容が常に変更されている表の場合は、頻度の高い決まったインターバルで実行します。
- 相当数の表行のデータを追加または変更する場合、毎回その操作の後に実行します。そのような操作には、バッチ更新や、行を追加するデータ・ロードがあります。

カタログ統計

SQL コンパイラーで SQL 照会プランが最適化されるとき、その判断は、データベースの表および索引のサイズについての統計情報の影響を大きく受けます。行を選択したり表を結合するために表および索引の特定の列が使用される場合、オプティマイザーでは、これらの列におけるデータ分散についての情報も使用されます。この情報は、照会ごとの代替アクセス・プランのコストを見積もるために使用されます。

表サイズおよびデータ分散情報に加えて、索引のクラスター比率、索引内のリーフ・ページ数、元のページからオーバーフローする表の行数、および表内の入力されているページ数と空のページ数についての統計情報も収集できます。この情報は、表および索引を再編成する時期を決定するために使用します。

RUNSTATS ユーティリティーを実行すると、ローカル・データベース内の特定の表および索引に関する統計情報が収集されます。収集された統計は、システム・カタログ表に保管されます。

収集される統計は、ユーティリティーを実行するパーティションにある表パーティション、またはデータベース・パーティション内で表が入っている最初のパーティションにある表パーティションに関するものだけです。

注: RUNSTATS ユーティリティーではニックネームの使用がサポートされていないので、フェデレーテッド・データベース照会の場合は別の方法で統計を更新します。照会でフェデレーテッド・データベースにアクセスする場合は、すべてのデータベース内の表に対して RUNSTATS を実行し、次いでリモート・データベースにアクセスするニックネームをドロップして再作成することにより、新しい統計をオプティマイザーで使用できるようにします。

RUNSTATS の効率と収集された統計の有用性を改善するには、以下のヒントを考慮してください。

- 表を結合するために使用される列、または WHERE、GROUP BY、および照会の類似の文節で使用される列に関する統計のみを収集します。索引に含まれていれば、これらの列は RUNSTATS コマンドで ONLY ON KEY COLUMNS 文節を使用して指定できます。
- 特定の表および表内の特定の列について *num_freqvalues* および *num_quantiles* の値をカスタマイズします。
- 詳細な索引統計のために実行されるバックグラウンド計算量を減らすために、SAMPLE DETAILED 文節を指定して DETAILED 索引統計を収集します。SAMPLE DETAILED 文節を指定すると、統計を収集するのに必要な時間が短縮され、ほとんどの場合に十分な精度が得られます。
- データが入っている表の索引を作成する場合は、索引の作成時に統計が作成されるように、COLLECT STATISTICS 文節を追加します。
- 表の行が大量に追加または削除された場合、または統計を収集する列のデータが更新された場合は、 RUNSTATS を再実行して統計を更新してください。
- RUNSTATS は単一パーティションの統計しか収集しないので、常にデータがすべてのパーティションに分散していない場合には、統計はあまり正確になりません。データ分散がひずんでいると疑われる場合は、 RUNSTATS を実行する前に REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用して、パーティションにデータを再分散させることもできます。

以下の場合、分散統計は収集されません。
- 構成パラメーター *num_freqvalues* および *num_quantiles* をゼロ (0) に設定している場合。
- データの分散が分かっている場合。たとえば、各データ値がユニークである場合など。
- 列が、統計が収集されないデータ・タイプである場合。該当するデータ・タイプは、LONG、ラージ・オブジェクト (LOB)、または構造化列です。
- 副表の行タイプの場合、表レベルの統計 NPAGES、FPAGES、および OVERFLOW は収集されません。
- 変位値分散が要求されたが、列の中に非 NULL 値が 1 つしかない場合。
- 拡張索引または宣言済み一時表の場合。

注: 宣言済み一時表に対して RUNSTATS を実行することはできますが、宣言済み一時表にはカタログ項目がないため、結果の統計はシステム・カタログに保管されません。ただし、宣言済み一時表のカタログ情報を表すメモリー構造に保管されます。したがって、ときには、これらの表に対して RUNSTATS を実行することが役立つ場合もあります。

関連概念:

- 121 ページの『カタログ統計の表』
- 126 ページの『収集される統計情報』
- 139 ページの『モデル化および what-if の計画のカタログ統計』
- 141 ページの『実動データベースのモデル化の統計』
- 143 ページの『手動でのカタログ統計更新の一般規則』

関連タスク:

- 112 ページの『カタログ統計の収集』

関連資料:

- 507 ページの『num_freqvalues - 保存される頻度値の数』
- 508 ページの『num_quantiles - 列の変位値の数』
- 「コマンド・リファレンス」の『RUNSTATS コマンド』

カタログ統計の収集と分析

このセクションでは、カタログ統計の収集についてのガイドラインと手順を示します。また、データ分散やクラスタリングなどをさらに理解するために、収集したデータを分析することについてのヒントも示します。

統計の収集および更新のガイドライン

RUNSTATS コマンドは、表と索引データの両方に関する統計を収集し、アクセス・プランに関する正確な情報を、オプティマイザーに提供します。

注: RUNSTATS は、これを実行したパーティションにある表の統計しか収集しません。このパーティションの RUNSTATS 結果は、他のパーティションに外挿されます。 RUNSTATS を実行したデータベース・パーティションに表パーティションが含まれていない場合、発行した要求は、その表のパーティションを保持するデータベース・パーティション・グループの中の先頭のデータベース・パーティションに送られます。

以下のような状態のときに、 RUNSTATS ユーティリティーを使用して統計を収集してください。

- 表にデータがロードされており、該当する索引が作成済みの場合。
- 表に新しい索引を作成する場合。最後に RUNSTATS を実行してからその表を変更していない場合、新しい索引に関してのみ RUNSTATS を実行する必要があります。
- 表が REORG ユーティリティーによって再編成されている場合。
- データの変更、削除、および挿入によって、表とその索引が大幅に変更される場合。 (この場合「大量」とは、表データと索引データの 10 ～ 20 % 程度が影響を受けたことを意味します。)
- パフォーマンスが重要なアプリケーション・プログラムをバインドする前。
- 現在の表と前の表を比較する場合。決まったインターバルで統計を更新すると、パフォーマンス上の問題を容易に発見できます。
- プリフェッチ数量が変更される場合。
- REDISTRIBUTE DATABASE PARTITION GROUP ユーティリティーを使用している場合。

  注: DB2® の以前のバージョンでは、このコマンドは DATABASE PARTITION GROUP キーワードではなく NODEGROUP キーワードを使用していました。

RUNSTATS のパフォーマンスを向上させ、統計の保管に使用されるディスク・スペースを節約するために、データ分散統計を収集しなければならない列だけを指定するように意識してください。

理想的には、統計を実行した後でアプリケーション・プログラムを再バインドするようにします。新しい統計の場合、照会オプティマイザーによって別のアクセス・プランを選択される場合があります。

一度にすべての統計を収集する時間がない場合には、一度に更新する表および索引の統計を少しだけにして、表のセットに対して順番に RUNSTATS を実行してください。表で行われた活動の結果として、 RUNSTATS を実行して選択的に部分更新を行ったある期間と次の期間の間に不整合が見つかった場合は、照会の最適化中に警告メッセージ (SQL0437W、理由コード 6) が出されます。たとえば、最初は RUNSTATS を使用して表分散統計を収集したとし、次に、RUNSTATS を使用して索引統計を収集したとします。照会の最適化中に、表で行われた活動の結果として不整合が検出され、削除された場合、警告メッセージが出されます。このような場合、 RUNSTATS を再度実行し、分散統計を更新しなければなりません。

確実に索引統計を表と同期化させるために、 RUNSTATS を実行して、表と索引の両方の統計を同時に収集してください。索引統計では、最後に実行された RUNSTATS で収集された表統計および列統計のほとんどを保持しています。表統計を最後に収集した時点以降にその表が広範囲に変更された場合には、その表の索引統計のみを収集すると、すべてのノードでそれらの 2 つの統計のセットが同期していないことになります。

実稼働中のシステムで RUNSTATS を呼び出すと、稼働中処理のパフォーマンスに悪影響を与える可能性があります。 RUNSTATS ユーティリティーはスロットル・オプションをサポートするようになりました。これは、ハイレベルのデータベース・アクティビティー中の RUNSTATS 実行のパフォーマンス影響を制限するのに使用できます。

関連概念:

- 109 ページの『カタログ統計』
- 130 ページの『分散統計のオプティマイザーの使用』
- 119 ページの『自動統計収集』

関連タスク:

- 112 ページの『カタログ統計の収集』
- 113 ページの『特定の列に関する分散統計の収集』
- 115 ページの『詳細索引統計の収集』

カタログ統計の収集

表および索引のカタログ統計を収集することにより、オプティマイザーが照会に最も適したアクセス・プランを選択するために使用する情報を提供できます。

前提条件:

表および索引を含むデータベースに接続し、以下の許可レベルのいずれかを持っていなければなりません。

- sysadm
- sysctrl
- sysmaint

- dbadm
- その表に対する CONTROL 特権

手順:

カタログ統計を収集するには、以下を行います。

1. 統計情報を収集する表および索引を含んでいるデータベースに接続します。
2. DB2 コマンド行から、適切なオプションを指定して RUNSTATS コマンドを実行します。これらのオプションを使用することにより、表および索引に対して実行される照会で収集される統計を調整できます。

   注: RUNSTATS は、これを実行したパーティションにある表の統計しか収集しません。このパーティションの RUNSTATS 結果は、他のパーティションに外挿されます。 RUNSTATS を実行したデータベース・パーティションに表パーティションが含まれていない場合、発行した要求は、その表のパーティションを保持するデータベース・パーティション・グループの中の先頭のデータベース・パーティションに送られます。
3. RUNSTATS が完了したら、 COMMIT ステートメントを発行してロックを解放します。
4. 統計情報を再生成した表および索引にアクセスするパッケージを再バインドします。

グラフィカル・ユーザー・インターフェースを使用して、オプションの指定および統計の収集を行うには、コントロール・センターを使用します。

関連概念:

- 109 ページの『カタログ統計』
- 111 ページの『統計の収集および更新のガイドライン』

関連タスク:

- 113 ページの『特定の列に関する分散統計の収集』
- 115 ページの『詳細索引統計の収集』
- 272 ページの『表の再編成時の決定』

関連資料:

- 「*コマンド・リファレンス*」の『RUNSTATS コマンド』

特定の列に関する分散統計の収集

RUNSTATS とそれ以降の照会計画分析の両方を効果的に行うために、 WHERE、GROUP BY、および同様の文節で照会が使用する表列だけで、分散統計を収集することもできます。列の結合グループで、カーディナリティー統計を収集することもできます。グループ内の列を参照する照会の選択を確立するときに、オプティマイザーはそのような情報を使用して、列の相関を検出します。

以下のステップでは、データベースが **sales** であり、それに索引 **custidx1** および **custidx2** を含む、表 **customers** が入っているということを想定しています。

前提条件:

表および索引を含むデータベースに接続し、以下の許可レベルのいずれかを持っていなければなりません。

- sysadm
- sysctrl
- sysmaint
- dbadm
- その表に対する CONTROL 特権

注: RUNSTATS は、これを実行したパーティションにある表の統計しか収集しません。このパーティションの RUNSTATS 結果は、他のパーティションに外挿されます。 RUNSTATS を実行したデータベース・パーティションに表パーティションが含まれていない場合、発行した要求は、その表のパーティションを保持するデータベース・パーティション・グループの中の先頭のデータベース・パーティションに送られます。

手順:

特定の列に関する統計を収集するには、以下を行います。

1. **sales** データベースに接続します。
2. DB2 コマンド行で以下のコマンドのいずれかを実行します。どれを実行するかは、要件によって異なります。
   - 列 **zip** および **ytdtotal** に関する分散統計を収集するには、以下を行います。

     ```
     RUNSTATS ON TABLE sales.customers
          WITH DISTRIBUTION ON COLUMNS (zip, ytdtotal)
     ```

   - 同じ列に関する分散統計を収集するものの、分散のデフォルトを調整するには、以下を行います。

     ```
     RUNSTATS ON TABLE sales.customers
          WITH DISTRIBUTION ON
          COLUMNS (zip, ytdtotal NUM_FREQVALUES 50 NUM_QUANTILES 75)
     ```

   - **custidx1** および **custidx2** で索引付けされている分散統計を収集するには、以下を行います。

     ```
     RUNSTATS ON TABLE sales.customer
             ON KEY COLUMNS
     ```

   - 特定の列 **zip** と **ytdtotal**、および **region** と **territory** を含む列グループだけに関する列統計を収集するには、以下を行います。

     ```
     RUNSTATS ON TABLE sales.customers
              ON COLUMNS (zip, (region, territory), ytdtotal)
     ```

コントロール・センターを使用して、分散統計を収集することもできます。

関連概念:

- 121 ページの『カタログ統計の表』

関連タスク:

- 112 ページの『カタログ統計の収集』
- 115 ページの『詳細索引統計の収集』

詳細索引統計の収集

索引統計を収集することにより、オプティマイザーで、照会を解決するために索引を使用すべきかどうかを評価できます。

以下のステップでは、データベースが **sales** であり、それに索引 **custidx1** および **custidx2** を含む、表 **customers** が入っているということを想定しています。

前提条件:

表および索引を含むデータベースに接続し、以下の許可レベルのいずれかを持っていなければなりません。

- sysadm
- sysctrl
- sysmaint
- dbadm
- その表に対する CONTROL 特権

SAMPLED DETAILED オプションを指定して RUNSTATS を実行するには、統計ヒープが 2MB 必要です。この追加メモリー要件用に、追加の 488 4K ページを *stat_heap_sz* データベース構成パラメーターの設定に割り振ってください。ヒープが小さすぎた場合、 RUNSTATS は統計の収集を試行する前にエラーを戻します。

手順:

索引の詳細な統計を収集するには、以下を行います。

1. **sales** データベースに接続します。
2. DB2 コマンド行で以下のコマンドのいずれかを実行します。どれを実行するかは、要件によって異なります。
   - **custidx1** と **custidx2** の両方に関する詳細な統計を作成するには、以下を行います。

     ```
     RUNSTATS ON TABLE sales.customers AND DETAILED INDEXES ALL
     ```

   - 両方の索引に関する詳細な統計を作成するものの、索引入力ごとに詳細な統計を実行するのではなく、サンプリングを使用するには、以下を行います。

     ```
     RUNSTATS ON TABLE sales.customers AND SAMPLED DETAILED INDEXES ALL
     ```

   - 索引および表の統計に整合性を持たせるために、表の分散統計のほかに詳細なサンプリング済み統計を作成するには、以下を行います。

     ```
     RUNSTATS ON TABLE sales.customers
              WITH DISTRIBUTION ON KEY COLUMNS
              AND SAMPLED DETAILED INDEXES ALL
     ```

コントロール・センターを使用して、索引および表統計を収集することもできます。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』
- 126 ページの『収集される統計情報』

- 135 ページの『詳細索引統計』

関連タスク:

- 112 ページの『カタログ統計の収集』

表データのサンプルでの統計の収集

表統計は、特定の照会に最適なアクセス・プランを選択するときにオプティマイザーによって使用されるので、統計が特定時点の表の状態を正確に反映するものであることは重要です。表に対する活動が増えれば増えるほど、統計収集の頻度も増やす必要があります。データベースのサイズが増えるにつれて、効率的に統計を収集することが一層重要になります。統計を収集する表データからランダムにサンプリングすると、 RUNSTATS のパフォーマンスは向上します。 I/O 性能や CPU 制約が限られたシステムの場合、こうしたパフォーマンス上の利点は計り知れません。サンプルが小さいほど、RUNSTATS は速く終了します。

バージョン 8.2 以降、RUNSTATS コマンドには TABLESAMPLE オプションが用意されており、表のデータのサンプルから統計を収集できるようになっています。サンプリングはデータの一部だけを使用するので、この機能を使用すると統計収集の効率がよくなります。同時に、複数のサンプリング方式によって、正確性が保証されます。

サンプルを収集する方法は 2 通りから指定できます。ベルヌーイ方式は、行レベルでデータをサンプリングします。データ・ページの完全表スキャンを実行中に、各行を順番に検討し、数値パラメーターで指定した確率 P に基づいて行を選択します。こうして選択した行からのみ、統計が収集されます。同様にして、SYSTEM 方式はページ・レベルでデータをサンプリングします。このとき、各ページを確率 P に基づいて選択し、確率 1-P/100 で除外します。

サンプルに組み込まれるページごとに 1 つの I/O だけが必要であるため、ページ・レベルのサンプリングは非常に高いパフォーマンスを実現します。行レベルのサンプリングを使用した場合、各表ページを完全表スキャンに取り込むため、I/O コストは減りません。しかしながら、I/O の量が減ることはなくても、統計情報の収集処理は CPU での処理を主としたものなので、行レベルのサンプリングでもパフォーマンスはかなり向上します。

データ値が高クラスター化されている状況では、ページ・レベル・サンプリングよりも行レベル・サンプリングのほうが良いサンプルを提供できます。ページ・サンプリングと比べると、行レベルのサンプルでは各データ・ページから P パーセントの行をサンプリングするので、全データの統計的傾向をより反映することができます。ページ・レベルのサンプリングでは、P パーセントのページに含まれる行すべてがサンプリング対象になります。行が表の中でランダムに分散している場合は、行サンプル統計とページ・サンプル統計の正確さはほぼ同じです。

RUNSTATS コマンドに REPEATABLE オプションを指定せずに何回も実行すると、各サンプルがランダムに生成されます。 REPEATABLE 文節を使用すると、TABLESAMPLE オプションを指定した RUNSTATS コマンドを最後に実行したときと同じサンプルが生成されます。変化のすくないデータをもつ表に対して一貫性のある統計の生成が必要な場合に、この機能は便利です。

関連概念:

- 92 ページの『SQL 照会でのデータ・サンプリング』

関連資料:

- 「コマンド・リファレンス」の『RUNSTATS コマンド』

統計プロファイルを使用した統計の収集

RUNSTATS ユーティリティーには、統計プロファイルを登録して使用するオプションがあります。このプロファイルは、特定の表にどの統計を収集するかを指定したオプションの集まりです (たとえば、表統計、索引統計、または分散統計)。

この機能によって、RUNSTATS コマンド発行時に指定するオプションの保管が可能になり、コマンド・オプションをタイプし直さなくても繰り返し同じ統計を表に収集できるので、統計の収集が容易になります。

統計の収集中であってもなくても、統計プロファイルを登録または更新できます。たとえば、プロファイルの登録と統計の収集を同時に行うには、 RUNSTATS コマンドに SET PROFILE オプションを指定して実行します。統計の収集は行わずに、プロファイルの登録だけを行う場合は、 RUNSTATS コマンドに SET PROFILE ONLY オプションを指定して実行します。

すでに登録してある統計プロファイルを使用して統計を収集するには、 RUNSTATS コマンドに表の名前と USE PROFILE オプションだけを指定して実行します。

統計プロファイル内で特定の表について現在指定されているオプションを表示するには、次の SELECT ステートメントを使用してカタログ表を照会します。 tablename はプロファイルの指定を確認する表の名前です。

```
SELECT STATISTICS_PROFILE FROM SYSIBM.SYSTABLES WHERE NAME = tablename
```

ユーザー統計プロファイルを登録すると、そのプロファイルに対応する RUNSTATS コマンド・ストリングは、作成されると同時にカタログ表 SYSIBM.SYSTABLES の STATISTICS_PROFILE 列に保管されます。登録した統計プロファイルのコマンド・ストリング・サイズが STATISTICS_PROFILE 列より大きい場合には、RUNSTATS ユーティリティーは列のサイズにあわせて、コマンド・ストリングを切り捨て SYSTABLES.STATISTICS_PROFILE 列に保管します。プロファイルの内部バージョンとしては切り捨てられずにシステム・カタログに保持されます。

自動統計プロファイル作成

DB2® 自動統計プロファイル作成機能を使用することで、統計プロファイルを自動的に生成することもできます。この機能を使用可能にすると、データベース活動に関する情報が収集され、照会フィードバック・ウェアハウスに保管されます。このデータに基づいて、統計プロファイルが生成されます。この機能を使用可能にすると、特定のクエリー処理とどの統計が関係し合っているのかが分からないために生じる問題が軽減され、最小限の統計情報を収集することで最適な処理パフォーマンス実現できます。

この機能は自動統計収集機能とともに使用できます。自動統計収集機能は、自動的に生成された統計プロファイル内に含まれる情報に基づいて統計保守を自動的にスケジュールに入れます。

この機能を使用可能にするには、該当する構成パラメーターを設定して、自動表保守がすでに使用可能になっていることが必要です。 AUTO_STATS_PROF 構成パラメーターは照会フィードバック・データの収集を活動化し、 AUTO_PROF_UPD 構成パラメーターは統計プロファイルの生成を活動化し、自動統計収集で使用できるようにします。

注: 自動統計プロファイル生成は DB2 シリアル・モードでのみ活動化でき、フェデレーテッド intra_pararell 有効環境、またはデータベース・パーティショニング環境での照会には使用できません。

統計プロファイル生成が最適なのは、多数の述部を適用し大きく複雑な照会を実行する環境、述部列のデータどうしの相関関係が密な環境、複数の表を結合およびグループ化する環境です。一方、トランザクション処理を主に行う環境には、適していません。

この機能は、いくつかの異なる方法で使用できます。

- テスト環境。テスト・システムでは、AUTO_STATS_PROF と AUTO_PROF_UPD を ON に設定します。テスト・システムでは、ランタイム・モニターのパフォーマンス・オーバーヘッドを大目に見ることができます。テスト・システムで実際のデータや照会を使用する場合には、この環境によって RUNSTATS の統計パラメーターに適正な相関や設定を調べることができ、その結果を統計プロファイルに保管しておきます。このプロファイルを実動システムに転送することも可能で、その場合照会にモニター・オーバーヘッドを取らずに済みます。
- 実稼働環境で、特定の照会に関するパフォーマンスの問題を扱う。特定の照会セットにパフォーマンス上の問題が検出され、その原因が統計または相関の不良である可能性がある場合に、 AUTO_STATS_PROF をオンにして一定時間ターゲット・ワークロードを実行します。自動統計プロファイルは照会のフィードバックを分析して、 SYSTOOLS.OPT_FEEDBACK_RANKING* 表に勧告を作成します。これらの勧告をよく読み、それに基づいて手動で統計プロファイルを改良できます。この勧告に基づいて DB2 が統計プロファイルを自動で更新するためには、AUTO_STATS_PROF と AUTO_PROF_UPD を両方オンにします。

注: 照会のモニター、および照会フィードバック・データのフィードバック・ウェアハウスへの保管に関連して、いくらかのパフォーマンス・オーバーヘッドがあります。

照会フィードバック・ウェアハウスの作成: フィードバック・ウェアハウスは SYSTOOLS スキーマの 3 つの表からなっており、これらの表には照会実行中に検出された述部に関する情報が保管されます。 3 つの表とは、OPT_FEEDBACK_QUERY、OPT_FEEDBACK_PREDICATE、および OPT_FEEDBACK_PREDICATE_COLUMN です。

自動統計プロファイル作成を使用するには、SYSINSTALLOBJECTS ストアード・プロシージャーを使用してまず照会フィードバック・ウェアハウスを作成する必要があります。このストアード・プロシージャーは、オブジェクトを作成およびドロップする SYSTOOLS スキーマの共通ストアード・プロシージャーです。

次のようにして、SYSINSTALLOBJECTS ストアード・プロシージャーを呼び出します。

```
call SYSINSTALLOBJECTS ( toolname, action, tablespacename, schemaname)
```

説明:

toolname
オブジェクトを作成またはドロップするツールの名前を指定します。ここでは、″ASP″ または ″AUTO STATS PROFILING″ です。

action 行う処置を指定します。'C' は作成、'D' はドロップ。

tablespacename
フィードバック・ウェアハウス表を作成する表スペースの名前。この入力パラメーターはオプションです。このパラメーターを指定しない場合、デフォルトのユーザー・スペースが使用されます。

schemaname
オブジェクトを作成またはドロップするときに使用するスキーマの名前。このパラメーターはここでは使用しません。

たとえば、フィードバック・ウェアハウスを表スペース ″A″ に作成するには、次のように入力します。 `call SYSINSTALLOBJECTS ("ASP", 'C', "A", "")`

関連概念:

- 119 ページの『自動統計収集』

関連タスク:

- 120 ページの『自動統計収集の使用』

自動統計収集

DB2® オプティマイザーは、あらゆる照会に関し、最も効果的なアクセス・プランを判別するためにカタログ統計を使用します。ある表または索引に関する統計が古い、あるいは不完全であるなら、オプティマイザーが最適でないプランを選択したり、照会の実行がスローダウンしたりする可能性があります。しかし、特定のワークロードのために収集する統計を決めて、それらの統計を最新に保つことには、時間がかかります。

DB2 の Automated Table Maintenance 機能の一部である自動統計収集を使えば、ワークロードに必要な統計と更新の必要な統計を DB2 に判断させることができます。自動統計収集を使用可能にすると、DB2 は自動的にバックグラウンドで RUNSTATS ユーティリティーを実行して、正しい統計を収集および保守します。

自動統計収集によるパフォーマンスへの影響は、以下のいくつかの方法によって最低限に抑えられています。

- スロットル調整した RUNSTATS を使って統計収集を実行する。スロットル調整することにより、現在のデータベースのアクティビティーに基づいて、RUNSTATS ユーティリティーの消費するリソースの量が制御されます。データベースのアクティビティーが増加すると RUNSTATS ユーティリティーの実行速度が低下し、リソースの要求が小さくなります。
- パフォーマンスの最適化に必要な最低限の統計のセットだけを収集する。これは統計プロファイルを使用して実現されます。統計プロファイルでは、以前のデータベース・アクティビティーに関する情報を使用し、データベースのアクティビティーのタイプを考えて、データベースのワークロードに必要な統計と、それらの統計が無効になるまでの時間を見極めます。

- アクティビティーのレベルが高い表 (更新、削除、および挿入の数で測る) だけを、統計収集の対象として考慮する。大きい表 (構成ページが 4000 を超える表) の場合は、表に対する大量のアクティビティーによって確かに統計が変化したかどうかを判断するためにサンプリングします。変化が確実な場合にだけ、このような大きな表の統計は収集されます。
- メンテナンス・ポリシー定義で決定した最適なメンテナンス時間帯に実行されるように、 RUNSTATS ユーティリティーは自動でスケジュールリングされます。このポリシーはまた自動統計収集を有効にする表のセットも指定するので、不必要なリソース消費をさらにおさえます。
- 自動統計収集の実行中も、影響を受ける表に対し、あたかも RUNSTATS がその表で実行されていないかのように、引き続き通常のデータベース・アクティビティー (更新、挿入、削除) が可能。

関連概念:

- 117 ページの『統計プロファイルを使用した統計の収集』

関連タスク:

- 120 ページの『自動統計収集の使用』

関連資料:

- 541 ページの『util_impact_lim - インスタンス影響ポリシー』
- 510 ページの『autonomic_switches - 自動保守スイッチ』

自動統計収集の使用

正確で完全なデータベースの統計を得ることは、効率的なデータ・アクセスと最適のワークロード・パフォーマンスにとって重要です。関係のあるデータベース統計を更新および保守するには、自動表保守機能の自動統計収集機能を使用してください。シリアル環境では、照会データを収集し、統計プロファイルを生成することにより、オプションでこの機能を拡張できます。これは、自分のワークロードにまさに必要な統計のセットを DB2 が自動で収集する助けになります。このオプションは、intra_parallel 有効環境、データベース・パーティショニング、またはフェデレーテッド環境では使用できません。

手順:

この機能は、グラフィカル・ユーザー・インターフェース・ツールか、コマンド行インターフェースのいずれかを使用してオンにすることができます。

- グラフィカル・ユーザー・インターフェース・ツールを使って、データベースを自動統計収集するようセットアップするには、次のようにします。
  1. コントロール・センターでデータベース・オブジェクトを右クリックするか、ヘルス・センターで自動統計収集を構成するデータベース・インスタンスを右クリックすることにより、「自動保守の構成」ウィザードを開く。ポップアップ・ウィンドウで**「自動保守の構成」**を選択してください。
  2. このウィザードで、自動統計収集を使用可能にし、自動的に統計を収集する表を指定し、 RUNSTATS ユーティリティーを実行するメンテナンス時間帯を指定することができる。

3. オプション: 自動統計プロファイルの生成を使用可能にするには、コマンド行インターフェースを使って次の 2 つの構成パラメーターを「ON」に設定する。
   - AUTO_STATS_PROF
   - AUTO_PROF_UPD

- コマンド行インターフェースを使って、データベースを自動統計収集するようセットアップするには、次のようにします。
  1. 次の各構成パラメーターを「ON」に設置する。
     - AUTO_MAINT
     - AUTO_TBL_MAINT
     - AUTO_RUNSTATS
  2. オプション: 自動統計プロファイルの生成を使用可能にするには、次の 2 つの構成パラメーターを「ON」に設定する。
     - AUTO_STATS_PROF
     - AUTO_PROF_UPD

関連概念:

- 117 ページの『統計プロファイルを使用した統計の収集』
- 119 ページの『自動統計収集』

関連資料:

- 510 ページの『autonomic_switches - 自動保守スイッチ』

収集される統計

このセクションでは、カタログ統計表をリストし、これらの表のフィールドの使用について説明します。統計表の説明の後のセクションでは、収集して表に保管することができるデータの種類について説明します。

カタログ統計の表

以下の表では、カタログ統計を含むシステム・カタログ表と、特定の統計を収集する RUNSTATS オプションに関する情報が提供されています。

表 18. 表統計 (SYSCAT.TABLES と SYSSTAT.TABLES)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
FPAGES	表が使用しているページの数	○	○
NPAGES	行が含まれているページの数	○	○
OVERFLOW	オーバーフローしている行の数	○	×
CARD	表内の行の数 (カーディナリティー)	○	○ (注 1)

表 18. 表統計 (SYSCAT.TABLES と SYSSTAT.TABLES) (続き)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
ACTIVE_BLOCKS	MDC 表の場合、占有されているブロックの合計数	○	×

注:
1. 表に定義された索引がないときに、索引の統計を要求した場合には、 CARD 統計が新たに更新されることはありません。 前の CARD 統計は引き続き保持されます。

表 19. 列統計 (SYSCAT.COLUMNS と SYSSTAT.COLUMNS)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
COLCARD	列カーディナリティー	○	○ (注 1)
AVGCOLLEN	列の平均長	○	○ (注 1)
HIGH2KEY	列内で 2 番目に高い値	○	○ (注 1)
LOW2KEY	列内で 2 番目に低い値	○	○ (注 1)
NUMNULLS	列内の NULL の数	○	○ (注 1)
SUB_COUNT	サブエレメントの平均数	○	× (注 2)
SUB_DELIM_LENGTH	各サブエレメントを分ける各区切り文字の平均長	○	× (注 2)

注:
1. 列統計は、索引キーの中の最初の列に関して収集されます。
2. これらの統計では、ブランクで区切られている一連のサブフィールドまたはサブエレメントを含む列の、データに関する情報が提供されています。 SUB_COUNT および SUB_DELIM_LENGTH 統計が収集されるのは、タイプ CHAR、VARCHAR、GRAPHIC、および VARGRAPHIC の、 1 バイト文字セットのストリング列の場合だけです。

表 20. 複数列統計 (SYSCAT.COLGROUPS および SYSSTAT.COLGROUPS)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
COLGROUPCARD	列グループのカーディナリティー	○	×

注: 以下の 2 つの表でリストされている複数列分布統計は、 RUNSTATS では収集されません。ただし、手動で更新することはできます。

表 21. 複数列分布統計 (SYSCAT.COLGROUPDIST および SYSSTAT.COLGROUPDIST)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
TYPE	F = 頻出値 Q = 変位値	○	×
ORDINAL	グループ内の列の序数	○	×
SEQNO	*n* 番目の TYPE 値を表す、シーケンス番号 *n*。	○	×

表 21. 複数列分布統計 (SYSCAT.COLGROUPDIST および SYSSTAT.COLGROUPDIST) (続き)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
COLVALUE	文字リテラルまたは NULL 値としてのデータ値	○	×

表 22. 複数列分布統計 2 (SYSCAT.COLGROUPDISTCOUNTS および SYSSTAT.COLGROUPDISTCOUNTS)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
TYPE	F = 頻出値 Q = 変位値	○	×
SEQNO	*n* 番目の TYPE 値を表す、シーケンス番号 *n*。	○	×
VALCOUNT	TYPE = F の場合、VALCOUNT は、この SEQNO によって識別される列グループでの、COLVALUE のオカレンスの数です。 TYPE = Q の場合、VALCOUNT は、この SEQNO の列グループで、値が COLVALUE 以下である行の数です。	○	×
DISTCOUNT	TYPE = Q の場合、この列には、この SEQNO の列グループでの、 COLVALUE 以下の固有値の数が含まれています。利用不能の場合は NULL です。	○	×

表 23. 索引統計 (SYSCAT.INDEXES と SYSSTAT.INDEXES)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
NLEAF	索引リーフ・ページの数	×	○
NLEVELS	索引レベルの数	×	○
CLUSTERRATIO	表データのクラスタリングの程度	×	○ (注 2)
CLUSTERFACTOR	クラスタリングの詳細の程度	×	詳細説明 (注 1、2)
DENSITY	索引の対象となるページ範囲にあるページ数に対する SEQUENTIAL_PAGES の比率 (パーセンテージ) (注 3)	×	○
FIRSTKEYCARD	索引の最初の列の固有値の数	×	○

表 23. 索引統計 (SYSCAT.INDEXES と SYSSTAT.INDEXES) (続き)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
FIRST2KEYCARD	索引の最初の 2 つの列の固有値の数	×	○
FIRST3KEYCARD	索引の最初の 3 つの列の固有値の数	×	○
FIRST4KEYCARD	索引の最初の 4 つの列の固有値の数	×	○
FULLKEYCARD	索引のすべての列の固有値の数 (ただし、すべての RID が削除対象としてマークされているタイプ 2 索引のキー値はすべて除く)	×	○
PAGE_FETCH_PAIRS	異なるバッファー・サイズでのページ取り出し見積もり	×	詳細説明 (注 1、2)
SEQUENTIAL_PAGES	索引キーの順序で、間をあまり空けずにディスクに位置づけられたリーフ・ページの数	×	○
AVERAGE_SEQUENCE_PAGES	順次にアクセスできる索引ページの平均数これは、プリフェッチャーが順次として検出できる索引ページの数です。	×	○
AVERAGE_RANDOM_PAGES	順次ページ・アクセスの間のランダム索引ページの平均数	×	○
AVERAGE_SEQUENCE_GAP	シーケンス間のギャップ	×	○
AVERAGE_SEQUENCE_FETCH_PAGES	順次にアクセスできる表ページの平均数これは、索引を使用して表の行をフェッチするときに、プリフェッチャーが順次として検出できる表ページの数です。	×	Yes (注 4)
AVERAGE_RANDOM_FETCH_PAGES	索引を使用して表の行をフェッチするときの、順次ページ・アクセスの間のランダム表ページの平均数	×	Yes (注 4)
AVERAGE_SEQUENCE_FETCH_GAP	索引を使用して表の行をフェッチするときの、シーケンス間のギャップ	×	Yes (注 4)
NUMRIDS	索引内のレコード ID (RID) の数 (タイプ 2 索引の削除対象の RID を含む)。	×	○
NUMRIDS_DELETED	索引内の削除対象としてマークされている RID の合計数 (すべてのレコード ID が削除対象としてマークされている、リーフ・ページの RID は除く)	×	○

表 23. 索引統計 (SYSCAT.INDEXES と SYSSTAT.INDEXES) (続き)

統計	説明	RUNSTATS オプション	
		TABLE	INDEXES
NUM_EMPTY_LEAFS	すべてのレコード ID が削除対象としてマークされている、リーフ・ページの合計数	×	○

注:
1. 詳細索引統計は、RUNSTATS コマンドに DETAILED 文節を指定します。
2. 表のサイズが相当大きいものでない限り、 DETAILED 文節を指定しても CLUSTERFACTOR および PAGE_FETCH_PAIRS は収集されません。 表のページ数が約 25 より大きいと、 CLUSTERFACTOR または PAGE_FETCH_PAIRS 統計が収集されます。この場合、CLUSTERRATIO は -1 です (収集されません)。表が比較的小さいと、RUNSTATS は CLUSTERRATIO だけを記入し、 CLUSTERFACTOR および PAGE_FETCH_PAIRS は記入しません。 DETAILED 文節を指定しないと、CLUSTERRATIO 統計だけが収集されます。
3. この統計は、その表に属する索引を含むページがファイルに対してどのくらいの比率 (パーセンテージ) を占めるかを測定します。表に定義された索引が 1 つしかない表の場合には、通常、DENSITY は 100 になります。DENSITY は、索引ページがプリフェッチされたときに、他の索引から不適切なページが平均してどのくらい読み取られたのかについて、オプティマイザーが見積もるのに使用されます。
4. これらの統計は、この表が DMS 表スペースにある場合は計算できません。
5. プリフェッチ統計は、統計収集が LOAD または CREATE INDEX コマンドの呼び出し時に指定されている場合でも、それらのコマンドの実行中には収集されません。プリフェッチ統計は、順次検出フラグ構成パラメーター (seqdetect) がオフになっている場合も収集されません。

表 24. 列分散統計 (SYSCAT.COLDIST と SYSSTAT.COLDIST)

統計	説明	RUNSTATS オプション	
		表	索引
DISTCOUNT	TYPE が Q の場合は、COLVALUE 統計以下の固有値の数	DISTRIBUTION (注 2)	No
TYPE	行の統計が頻出値統計または変位値統計かの標識	DISTRIBUTION	No
SEQNO	表の行を固有に識別するのに役立つシーケンス番号の頻度のランク	DISTRIBUTION	No
COLVALUE	頻出値統計または変位値統計を収集する際のデータ値	DISTRIBUTION	No
VALCOUNT	列内でデータ値が発生する頻度、または変位数の場合は、データ値 (COLVALUE) 以下の数値	DISTRIBUTION	No

注:
1. 列分散統計は、RUNSTATS コマンドに WITH DISTRIBUTION 文節を指定します。列の値が十分に不均一でないかぎり、分散統計は**収集されません**。
2. DISTCOUNT は、索引の最初のキー列である列でのみ収集されます。

関連概念:

- 109 ページの『カタログ統計』
- 126 ページの『収集される統計情報』

- 138 ページの『ユーザー定義関数の統計』
- 141 ページの『実動データベースのモデル化の統計』

収集される統計情報

表とそれに関連する索引で RUNSTATS ユーティリティーを実行する場合、以下の種類の統計情報が常にシステム・カタログ表に保管されます。

表および索引の場合
- 使用中のページの数
- 行を含んでいるページの数
- オーバーフローしている行の数
- 表内の行の数 (カーディナリティー)
- MDC 表の場合、データを含んでいるブロックの数

表内の各列、および索引キーの中の最初の場合
- 列のカーディナリティー
- 列の平均の長さ
- 列内で 2 番目に高い値
- 列内で 2 番目に低い値
- 列内の NULL の数

指定した列のグループの場合
- 列グループのタイム・スタンプに基づいた名前
- 列グループのカーディナリティー

索引のみの場合
- リーフ・ページの数
- 索引レベルの数
- この索引への表データのクラスタリングの程度。
- 索引によって占有されるページの範囲内のページ数に対する、索引キーの順序のディスク上のリーフ・ページの数の比率
- 索引の最初の列の固有値の数
- 索引の最初、2 番目、3 番目、4 番目の列の固有値の数
- 索引のすべての列の固有値の数
- 索引キーの順序で、間をあまり空けずにディスクに位置づけられたリーフ・ページの数
- すべての RID が削除対象としてマークされているページの数
- 一部の RID が削除対象としてマークされているページで、削除対象としてマークされている RID の数

索引に関する詳細な統計を要求した場合、索引に対する表のクラスタリングの程度、およびさまざまなバッファー・サイズに関するページ・フェッチの見積もりに関する、より優れた情報も保管します。

表および索引に関する以下の種類の統計も収集できます。
- データ分散統計

オプティマイザーはデータ分散統計を使用して、データが均等に分布していない表、および列に膨大な数の重複値がある表に関して効果的なアクセス・プランを見積もります。

- 詳細索引統計

  オプティマイザーは詳細な索引統計を使用して、索引を介した表へのアクセスの効果性を判別します。

- サブエレメント統計

  オプティマイザーは、 LIKE 述部でのサブエレメント統計、特にストリング内に組み込まれたパターンを検索するもの (`LIKE %disk%` など) を使用します。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』

関連タスク:

- 112 ページの『カタログ統計の収集』
- 113 ページの『特定の列に関する分散統計の収集』
- 115 ページの『詳細索引統計の収集』

分散統計

以下の 2 つの種類のデータ分散統計を収集できます。

- 頻度統計

  これらの統計は、重複の数が最も多い列およびデータ値、その次に重複値が多い列およびデータ値というように、 *num_freqvalues* 構成パラメーターで指定したレベルまで、それらの情報を提供します。頻度統計の収集を使用不可にするには、 *num_freqvalues* を 0 に設定します。

  num_freqvalues を行ごと、および特定の列の RUNSTATS オプションとして設定することもできます。

- 変位値統計

  これらの統計は、データ値がほかの値との関連でどのように分布しているかに関する情報を提供します。これらの K 変位値と呼ばれる統計の値 V とは、 K 個以上の値がその値 V 以下であることを示すものです。 K 変位値は、値の昇順をソートすることで計算できます。 K 変位値は、その範囲の最後から K 番目の位置の値です。

  列データ値がグループ化されるセクションの数を指定するには、 *num_quantiles* データベース構成パラメーターを、 2 ～ 32,767 の間の値に設定します。デフォルト値である 20 では、オプティマイザー見積エラーは、等価、より小、またはより大の述部で最大プラス・マイナス 2.5%、 BETWEEN 述部で最大エラーでプラス・マイナス 5% です。変位値統計の収集を使用不可にするには、 num_quantiles を 0 または 1 に設定します。

  num_quantiles を行ごと、または特定の列に設定することもできます。

注: より大きな *num_freqvalues* および *num_quantiles* 値を指定すると、RUNSTATS の実行時に、 *stat_heap_sz* データベース構成パラメーターで指定されているメモリーでは不足することがあります。また多くの CPU リソースも必要になることがあります。

分散統計を収集する場合

所定の表に関して分散統計を作成および更新すべきかどうかを判断するには、次の 2 つの要因を考慮してください。

- アプリケーションが静的または動的 SQL を使用するか。

  分散統計は、ホスト変数を使用しない動的 SQL および静的 SQL で最も便利です。ホスト変数と一緒に SQL を使用すると、オプティマイザーは分散統計を限定された方法で使用することになります。

- 列内のデータの分布が均一か。

  表内の少なくとも 1 つの列に、かなり 「不均一」なデータ分散があり、その列が下記のような等号述部または範囲述部に頻繁に現れる場合、分散統計を作成することをお勧めします。

```
WHERE C1 = KEY;
WHERE C1 IN (KEY1, KEY2, KEY3);
WHERE (C1 = KEY1) OR (C1 = KEY2) OR (C1 = KEY3);
WHERE C1 <= KEY;
WHERE C1 BETWEEN KEY1 AND KEY2;
```

データ分散における不均一性には、次の 2 種類がありますが、これらは一緒に発生する可能性があります。

- データが最高データ値と最低データ値の間に均等に分布しておらず、 1 つ以上の副次的なデータ範囲の中にクラスター化されている場合。データが範囲 (5、10) の中にクラスター化されている、以下の例を考慮してください。

C1
0.0
5.1
6.3
7.1
8.2
8.4
8.5
9.1
93.6
100.0

  変位値統計は、オプティマイザーがこのようなデータ分散を扱う際に役立ちます。

  列データが均一に分布していないかどうかを判別するには、以下の例のような照会を実行してください。

```
SELECT C1, COUNT(*) AS OCCURRENCES
  FROM T1
GROUP BY C1
ORDER BY OCCURRENCES DESC;
```

- 重複データ値が頻繁に発生する場合。データが以下の頻度で分布している列について考慮してください。

データ値	頻度
20	5
30	10
40	10
50	25
60	25
70	20
80	5

  オプティマイザーが重複値を処理するのに役立つように、変位値と頻出値の両方の統計を作成してください。

索引統計のみを収集する場合

次の状況では、索引データにのみ基づく統計を収集したいと思われるでしょう。
- RUNSTATS ユーティリティーが実行されてから新たに索引が作成されたため、表データの統計を再収集する必要がなくなった場合。
- 索引の最初の列に影響を与える大量のデータ変更がなされた場合。

指定する統計制度のレベル

分散統計を収集する精度を決定するには、データベース構成パラメーター *num_quantiles* および *num_freqvalues* を指定します。表または列の統計を収集するときに、これらのパラメーターを RUNSTATS オプションとして指定することもできます。大きな値を設定するほど、分散統計の作成および更新時に RUNSTATS が使用する精度が高くなります。ただし精度を高くすると、 RUNSTATS の実行とカタログ表で必要なストレージの両方で、より多くのリソースを使用しなければなりません。

ほとんどのデータベースでは、 *num_freqvalues* データベース構成パラメーターに 10 ～ 100 を指定します。頻出値の頻度とそれ以外の値の頻度が互いにほぼ等しいか、または頻出値以外の値の頻度が最頻出値の頻度に比べて無視できる程度になるよう、頻出値統計を作成するのが理想です。データベース・マネージャーはこの数より小さい数を収集することがあります。その理由は、複数個あるデータ値に限りこの統計は収集されるからです。変位値統計のみを収集する必要がある場合、 *num_freqvalues* をゼロに設定します。

変位値の数を設定する場合、 *num_quantiles* データベース構成パラメーターの設定として 20 ～ 50 を指定します。変位値の数を決めるときの経験則は、次のとおりです。
- 範囲照会の行数の見積もりで許容できる最大エラー P をパーセント単位で求めます。
- 述部が BETWEEN 述部である場合は変位値の数を約 100/P にし、述部がその他の種類の範囲述部 (<、<=、>、または >=) である場合には約 50/P にします。

たとえば、変位値の数が 25 であれば、最大見積エラーは、 BETWEEN 述部の場合は 4%、″>″ 述部の場合は、2% という結果になる必要があります。一般に、変位値は 10 以上を指定します。 50 以上の変位値が必要になるのは、極端に不均一なデ

ータの場合です。頻出値統計のみを必要とする場合、 *num_quantiles* をゼロに設定してください。値の範囲全体が 1 つの変位値内に収まるため、このパラメーターを「1」に設定する場合、変位値統計が収集されます。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』
- 130 ページの『分散統計のオプティマイザーの使用』
- 131 ページの『分散統計の使用の拡張例』

関連タスク:

- 112 ページの『カタログ統計の収集』
- 113 ページの『特定の列に関する分散統計の収集』

関連資料:

- 507 ページの『num_freqvalues - 保存される頻度値の数』
- 508 ページの『num_quantiles - 列の変位値の数』

分散統計のオプティマイザーの使用

オプティマイザーは分散統計を使用することにより、照会を満たす、可能なさまざまなアクセス・プランのコストをより正確に見積もることができます。

WITH DISTRIBUTION 文節を指定して RUNSTATS を実行しなかった場合、カタログ統計表には、表のサイズ、表内の最大値と最小値、その索引のいずれかに対する表のクラスタリングの多重度、索引付けされた列内の固有値の数に関する情報のみが含まれます。

最大値と最小値の間の値の分散に関して追加情報がなければ、オプティマイザーはデータ値が均等に分布していると想定します。データ値がそれぞれ大きく異なる場合、範囲内のいくつかの部分でクラスター化されている場合、または多くの重複値を含んでいる場合、オプティマイザーは最適なアクセス・プランよりも小さく選択します。

以下の例を考慮してください。

コストが最低のアクセス・プランを選択するために、等号述部または範囲述部を満たす列を含む行の数を、オプティマイザーが見積もる必要があるためです。見積もりが正確になるほど、オプティマイザーが最適のアクセス・プランを選択する可能性が大きくなります。たとえば、次の照会を考えてみます。

```
SELECT C1, C2
  FROM TABLE1
  WHERE C1 = 'NEW YORK'
  AND C2 <= 10
```

C1 と C2 の両方に索引が 1 つあるとします。この場合に可能なアクセス・プランの一つとしては、 C1 の索引を使用して `C1 = 'NEW YORK'` の行をすべて検索してから、取り出された行ごとに `C2 <= 10` かどうかを調べるというものが考えられます。別の計画としては、 C2 の索引を使用して `C2 <= 10` の行をすべて検索してから、取り出された行ごとに `C1 = 'NEW YORK'` であるかどうか調べる、という方法が

あります。通常、照会を実行するために主なコストは行を検索するコストであるため、最適なプランとは、必要な検索が最も少ないプランです。このようなプランを選択するには、各述部を満たす行の数を見積もる必要があります。

分散統計が使用可能でなく、 RUNSTATS が表に対して実行されている場合、オプティマイザー使用できる情報は、ある列について、2 番目に高いデータ値 (HIGH2KEY)、 2 番目に低いデータ値 (LOW2KEY)、固有値の数 (COLCARD)、および行数 (CARD) だけです。これにより、等号または範囲の述部を満たす行の数は、列内の各データ値の頻度はすべて等しく、データ値が区間 (LOW2KEY、HIGH2KEY) にわたって均等に分布する、という仮定の下に見積もられます。具体的には、等号述部 C1 = KEY を満たす行の数は、 CARD/COLCARD として見積もられ、また範囲述部 C1 BETWEEN KEY1 AND KEY2 を満たす行の数は、次のように見積もられます。

```
      KEY2 - KEY1
------------------  x CARD       (1)
 HIGH2KEY - LOW2KEY
```

以上の見積もりが正確な見積もりとなるのは、列内のデータ値の真の分散が、十分に均一である場合だけです。使用できる分散統計がなく、データ値の頻度が相互に大きく異なっているか、またはデータ値が幅の狭い区間 (LOW_KEY、HIGH_KEY) の中にクラスター化が行われている場合は、見積もりはけた違いのものになり、オプティマイザーが最適でないアクセス・プランを選択する可能性があります。

分散統計が使用できるときは、等価述部を満たす行数を計算するのに頻出値統計を使用し、範囲述部を満たす行数を計算するのに頻出値統計と変位数とを使用することによって、前述のエラーを大幅に小さくすることができます。

関連概念:

- 109 ページの『カタログ統計』
- 127 ページの『分散統計』
- 131 ページの『分散統計の使用の拡張例』

関連タスク:

- 113 ページの『特定の列に関する分散統計の収集』

分散統計の使用の拡張例

オプティマイザーが分散統計を使用する仕方を理解するには、フォーム C1 = KEY の等号述部を含む照会についてまず考慮してください。

頻出値の統計の例

頻出値が使用可能であれば、以下のように、オプティマイザーはそれらの統計を使用して、適切なアクセス・プランのコストを選択できます。

- KEY が、N 最大頻出値のいずれかである場合、オプティマイザーは、カタログ内に保管されている KEY の頻度を使用します。
- KEY が N 最大頻出値のどれでもない場合、オプティマイザーは、 (COLCARD - N) 個の非頻出値が均一に分散しているという仮定の下に、述部を満たす行の数を見積もります。つまり、行の数は、次のように見積もられます。

```
CARD - NUM_FREQ_ROWS
--------------------          (2)
   COLCARD - N
```

CARD は表内の行の数、COLCARD は列のカーディナリティー、
NUM_FREQ_ROWS は N 最大頻出値のいずれかと値の等しい行の合計数です。

たとえば、ある列 (C1) のデータ値の頻度が以下のようになっているとします。

データ値	頻度
1	2
2	3
3	40
4	4
5	1

この列に関して、最頻出値 (つまり N = 1) にのみ基づいた頻出値統計が使用可能な場合、表内の行の数は 50、列のカーディナリティーは 5 です。述部 C1 = 3 の場合、ちょうど 40 個の行がそれを満たしています。データが均等に分布しているとオプティマイザーが想定している場合、述部を満たす行の数は 50/5 = 10 として見積もられ、 -75% のエラーになります。オプティマイザーが頻出値統計を使用できる場合、行の数は 40 と見積もられ、エラーなしになります。

2 つの行が述部 C1 = 1 を満たす、別の例について考慮します。頻出値統計を使用しないと、述部を満たす行の数は 10 で 400% のエラーと見積もられます。以下の公式を使って、見積エラーを (パーセンテージで) 計算できます。

```
見積もられた行数 - 実際の行数
------------------------------  X 100
         実際の行数
```

頻出値統計 (N = 1) を使用すると、オプティマイザーは、たとえば以下のように前述の公式 (2) を使用して、この値の行の数を見積もります。

```
(50 - 40)
--------- = 3
 (5 - 1)
```

また、エラー率は、次のように 1 桁減ります。

```
 3 - 2
------- = 50%
   2
```

変位値統計の例

以下の変位値統計の説明では、「K 変位値」という用語が使用されます。列の *K 変位値* とは、さまざまなデータ値のうち、データ値が V 以下である行が「K」個以上あるような最小のデータ値 V のことです。 K 変位値は、データ値の昇順により列内の行をソートすることで計算できます。 K 変位値は、そのソートされた列の K 番目の行にあるデータ値です。

変位値統計を使用できる場合、以下の例で示されているように、オプティマイザーが範囲述部を満たす行の数をより正確に見積もることができます。以下の値を含む列 (C) があるとします。

C
0.0
5.1
6.3
7.1
8.2
8.4
8.5
9.1
93.6
100.0

次のように、K 変位値は、K = 1、4、7、および 10 のものが使用可能であるとします。

K	K 変位値
1	0.0
4	7.1
7	8.5
10	100.0

まず最初に述部 C <= 8.5 について考えます。前述のデータでは、正確には 7 つの行がこの述部を満たしています。データ分散が均一であると仮定し、前述の公式 (1) で KEY1 を LOW2KEY に置き換えると、述部を満たす行の数は、次のように見積もられます。

```
 8.5 - 5.1
---------- x 10 *= 0
93.6 - 5.1
```

*= は 「ほぼ等しい」という意味です。この見積もりのエラーは、約 -100% です。

変位値統計を使用できる場合、オプティマイザーは、変位値の 1 つから最も大きい値である 8.5 を見つけ、それに対応する K の値 7 を使用して行の数を見積もることにより、同じ述部 (C <= 8.5) を満たす行の数を見積もります。この場合、エラーは 0 になります。

次に、述部 C <= 10 について考えます。正確には 8 行がこの述部を満たしています。オプティマイザーが、データ分散が均一であると想定し、公式 (1) を使うと、述部を満たす行の数は 1 として見積もられ、エラーは -87.5% になります。

前述の例とは異なり、値 10 は、保管された K 変位値のどれでもありません。ただし、オプティマイザーは変位値を使用して、述部を満たす行の数を r_1 + r_2 と見積もります。ここで、r_1 は、述部 C <= 8.5 を満たす行の数、 r_2 は、述部 C <= 8.5 かつ C <= 10 を満たす行の数です。 前述の例のとおり、r_1 = 7 です。r_2 を見積もるため、オプティマイザーは線形補完を使用します。

```
        10 - 8.5
r_2 *= ---------- x (値を持つ行の数 > 8.5 かつ <= 100.0)
        100 - 8.5
        10 - 8.5
r_2 *= ---------- x (10 - 7)
        100 - 8.5
```

```
       1.5
r_2 *= ---- x (3)
       91.5
r_2 *= 0
```

最終的な見積もりは、r_1 + r_2 *= 7 であり、エラーは -12.5% のみです。

前述の例で変位値により見積もりの正確さが増したのは、実際のデータ値は範囲 5〜10 でクラスター化されているのに、標準の見積公式ではそのデータ値が 0〜100 の間で均一に分布していると想定されているためです。

このほかにも、変位値を使うと、データ値ごとの頻度の差が非常に大きい場合に正確さを向上させることができます。ある列のデータ値の頻度が次のようになっているものとします。

データ値	頻度
20	5
30	5
40	15
50	50
60	15
70	5
80	5

K 変位値は、K = 5、25、75、95、および 100 のものが使用可能であるとします。

K	K 変位値
5	20
25	40
75	50
95	70
100	80

さらに、3 個の最頻出値に基づく頻出値統計が使用可能であるとします。

述部 C BETWEEN 20 AND 30 を考えてみましょう。データ値の分散状況からは、正確には 10 個の行がこの述部を満たしていることが分かります。データ分散が均一であると仮定し、公式 (1) を使うと、述部を満たす行の数は次のように見積もられます。

```
30 - 20
-------  x 100 = 25
70 - 30
```

エラーは 150% です。

頻出値統計と変位値を使用すると、述部を満たす行の数は r_1 + r_2 と見積もられます。ここで、r_1 は述部 (C = 20) を満たす行数、 r_2 は述部 C > 20 AND C <= 30 を満たす行数となります。 公式 (2) を使用すると、r_1 は次のように見積もられます。

```
100 - 80
-------- = 5
  7 - 3
```

線形補完を使用すると、r_2 は次のように見積もられます。

```
  30 - 20
  ------- x (値を持つ行の数 > 20 かつ <= 40)
  40 - 20
  30 - 20
= ------- x (25 - 5)
  40 - 20
= 10,
```

最終見積もりは 15 になり、エラーは 3 分の 1 になりました。

関連概念:

- 109 ページの『カタログ統計』
- 127 ページの『分散統計』
- 130 ページの『分散統計のオプティマイザーの使用』
- 145 ページの『手動での分散統計の更新の規則』

関連タスク:

- 113 ページの『特定の列に関する分散統計の収集』

詳細索引統計

DETAILED 文節を指定して索引に RUNSTATS を実行する場合、必要なデータ・ページ・フェッチの数を、さまざまなバッファー・プール・サイズに基づいてオプティマイザーが見積もることのできる、統計情報を収集します。この追加情報は、索引を介して表にアクセスするコストを、オプティマイザーがより正確に見積もる助けになります。

注: 詳細な索引統計を収集する場合、 RUNSTATS はより多くのメモリーを必要とし、 CPU の処理にかかる時間も長くなります。統計的に有効な数の項目のみに関して計算された情報で、 SAMPLED DETAILED オプションは、統計ヒープとして 2MB が必要です。この追加メモリー要件用に、追加の 488 4K ページを *stat_heap_sz* データベース構成パラメーターの設定に割り振ってください。ヒープが小さすぎた場合、 RUNSTATS は統計の収集を試行する前にエラーを戻します。

DETAILED 統計の PAGE_FETCH_PAIRS と CLUSTERFACTOR は、表のサイズが十分に大きい (約 25 ページ以上) 場合にのみ収集されます。この場合には、CLUSTERFACTOR の値は 0 から 1 の間になり、 CLUSTERRATIO の値は -1 (収集されないことを示す) になります。 25 ページより小さい表の場合には、CLUSTERFACTOR の値は -1 (収集されないことを示す) になり、その表の索引に対して DETAILED 文節を指定した場合であっても、 CLUSTERRATIO の値は 0 から 100 になります。

DETAILED 統計は、完全な索引スキャンがさまざまなバッファー・サイズの下で行われるとした場合に、表のデータ・ページにアクセスするのに必要な物理入出力の数に関する、簡潔な情報を提供します。 RUNSTATS は索引のページをスキャンして、さまざまなバッファー・サイズのモデルを作り、ページ不在が起こる頻度の見積もりを収集します。たとえば、使用可能なバッファー・ページが 1 つだけの場合、索引によって参照される新しいページごとに、ページ不在になります。さらに悪い状況として、各行が別のページを参照する場合、入出力の数が、多くても索引

付けされた表内の行と同じになります。他方の極端な例をあげると、バッファーが表全体を入れられるくらい大きい (ただし最大バッファー・サイズ以下) 場合には、すべての表ページの読み取りが一度だけ行われます。結果として、物理入出力の数はバッファー・サイズの単調で非増加の関数になります。

また統計情報では、索引順序に対する表の行のクラスタリングの程度に関する、より優れた見積もりも提供されます。索引との関連でクラスター化が行われている表の行が少ないほど、索引を介して表の行にアクセスしなければならない入出力が多数あります。オプティマイザーは索引を介した表へのアクセスのコストを見積もる際に、バッファー・サイズとクラスタリングの程度の両方を考慮します。

照会において参照する列が索引内に収まっているわけではない場合には、DETAILED 索引統計を収集しなければなりません。また、DETAILED 索引統計は以下のような状況で使用する必要があります。

- 表にクラスタリングの程度が異なる複数の非クラスタリング索引が存在する場合
- クラスタリングの程度がキー値間で一様ではない場合
- 索引の値が不均一に更新される場合

予備知識なしでは、またはさまざまなバッファー・サイズでの索引スキャンを強制的に行って、その結果の物理入出力をモニターするのでなければ、これらの条件を評価するのは困難です。これらの状況が生じるか否かを最も簡単に判別する方法は、索引に関する DETAILED 統計を収集し、検査して、その結果の PAGE_FETCH_PAIRS が非線形であった場合にはそれらを保存することです。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』

関連タスク:

- 112 ページの『カタログ統計の収集』
- 115 ページの『詳細索引統計の収集』

サブエレメント統計

ブランクで区切られているサブフィールドまたはサブエレメントを含む列が表に含まれており、照会が WHERE 文節でそれらの列を参照している場合、アクセス・プランを最適なものにするために、サブエレメント統計を収集しなければなりません。

たとえば、データベースに、行ごとに文書の記述がある表 DOCUMENTS が含まれ、 DOCUMENTS には KEYWORDS という列があり、この列には、テキスト検索用に文書に関連するキーワードのリストが含まれているものとします。KEYWORDS の値としては次のものがある可能性があります。

```
'database simulation analytical business intelligence'
'simulation model fruit fly reproduction temperature'
'forestry spruce soil erosion rainfall'
'forest temperature soil precipitation fire'
```

この例では、各列の値は 5 個のサブエレメントから構成され、それぞれのエレメントにブランクで区切られたワード (キーワード) があります。

% match_all 文字を使用した列で LIKE 述部を指定する照会の場合、次のようになります。

```
SELECT .... FROM DOCUMENTS WHERE KEYWORDS LIKE '%simulation%'
```

列のサブエレメント構造に関して、オプティマイザーが基本統計を認識することが有効な場合がよくあります。

以下の統計は、 LIKE STATISTICS 文節を指定した RUNSTATS を実行した場合に収集されます。

SUB_COUNT
サブエレメントの平均数。

SUB_DELIM_LENGTH
サブエレメントを区切っている区切り文字の平均長。区切り文字は、ここでは 1 つ以上の連続するブランク文字です。

KEYWORDS 列の例では、SUB_COUNT は 5 で、SUB_DELIM_LENGTH は 1 となります。これは、区切り文字が 1 個のブランク文字であるためです。

DB2®_LIKE_VARCHAR レジストリー変数は、次の形式の述部をオプティマイザーが処理する方法に影響します。

```
COLUMN LIKE '%xxxxxx'
```

xxxxxx には任意のストリングが入ります。つまり、この場合は、LIKE 述部の検索値は % 文字で始まります。 (% 文字で終了しないこともあります。) 「wildcard LIKE predicates」として述べられます。 すべての述部に対して、オプティマイザーが述部に一致する行の数を見積もる必要があります。ワイルドカード LIKE 述部の場合、オプティマイザーは、一致する COLUMN が連結されている一連のエレメントの構造を含んでいると推定し、前後の % 文字を含まないストリングの長さに基づいて、各エレメントの長さを見積もります。

サブエレメント統計の値を調べるには、 SYSIBM.SYSCOLUMNS を照会します。たとえば、以下のようにします。

```
select substr(NAME,1,16), SUB_COUNT, SUB_DELIM_LENGTH
   from sysibm.syscolumns where tbname = 'DOCUMENTS'
```

注: LIKE STATISTICS 文節を使用する場合、 RUNSTATS は時間がかかることがあります。たとえば、DETAILED および DISTRIBUTION オプションを使用していないと、RUNSTATS は 5 文字の列のある表では 15% から 40% 長く時間がかかる可能性があります。 DETAILED または DISTRIBUTION オプションを指定していると、オーバーヘッドの絶対量が同じでも、オーバーヘッドのパーセンテージは少なくなります。このオプションの使用を考える場合、照会パフォーマンスの改善に対して、このオーバーヘッドを評価する必要があります。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』

関連タスク:

- 112 ページの『カタログ統計の収集』
- 113 ページの『特定の列に関する分散統計の収集』

ユーザーが更新可能なカタログ統計

このセクションでは、ユーザーが手動で更新できるカタログ統計データについて説明し、そのような手動変更についてのガイドラインを示します。ある場合には、この統計データは RUNSTATS によって収集されないために手動で追加する必要のあるデータです。また、他にも、収集された統計データをテスト・データベースにインポートし、実験のために実動データベースをモデル化するといった特定の目的のために、収集されたデータを変更したい場合があります。

警告: 実動データベースでは、 RUNSTATS によって収集されたデータを手動で更新しないでください。重大なパフォーマンス上の問題が生じる可能性があります。

ユーザー定義関数の統計

ユーザー定義関数 (UDF) に関する統計情報を作成する場合、SYSSTAT.FUNCTIONS カタログ・ビューを編集します。 UDF 統計が使用可能であれば、オプティマイザーは各種アクセス・プランのコストを見積もる際にそれらを使用できます。 RUNSTATS ユーティリティーでは、 UDF の統計は収集されません。使用できる統計がないなら、統計の列の値は -1 になり、オプティマイザーは単純な UDF を前提とするデフォルト値を使用します。

パフォーマンスを向上させるための見積もりを提供できる、統計列についての情報を、次の表にまとめます。

表 25. 関数統計 (SYSCAT.FUNCTIONS と SYSSTAT.FUNCTIONS)

統計	説明
IOS_PER_INVOC	関数が実行されるたびに実行される読み取り/書き込み要求の数の見積値。
INSTS_PER_INVOC	関数が実行されるたびに実行されるマシン語命令の数の見積値。
IOS_PER_ARGBYTE	入力引き数バイトごとに実行される読み取り/書き込み要求の数の見積値。
INSTS_PER_ARGBYTES	入力引き数バイトごとに実行されるマシン語命令数の見積もり。
PERCENT_ARGBYTES	関数が実際に処理する入力引き数バイトの平均比率 (%) の見積値。
INITIAL_IOS	関数が最初または最後に呼び出されるときにだけ実行される読み取り/書き込み要求の数の見積値。
INITIAL_INSTS	関数が最初または最後に呼び出されるときにだけ実行されるマシン語命令の数の見積値。
CARDINALITY	表関数によって生成される行数の見積値。

たとえば、米国製の靴のサイズを、それに対応する欧州製の靴のサイズに変換する UDF (EU_SHOE) を考えてみましょう。 (これらの 2 つの靴のサイズは、UDT になります。) この UDF の場合、統計列を次のように設定します。

- INSTS_PER_INVOC: 次のことを行うのに必要なマシン語命令数の見積もりに設定します。
  - EU_SHOE の呼び出し
  - 出力ストリングの初期化
  - 結果の戻り
- INSTS_PER_ARGBYTE: 入力ストリングを欧州製靴サイズに変換するのに必要な、マシン語命令数の見積もりに設定します。
- PERCENT_ARGBYTES: 100 に設定すると、入力ストリング全体を変換することになります。
- UDF は計算しか実行しないため、INITIAL_INSTS、 IOS_PER_INVOC、IOS_PER_ARGBYTE、および INITIAL_IOS は 0 に設定します。

PERCENT_ARGBYTES は、必ずしも入力ストリング全体を処理するとは限らない関数によって使用されます。たとえば、2 つの引き数を入力とし、第 2 引き数の中で、第 1 引き数が現れる最初の出現の開始位置を返す UDF (LOCATE) を考えてみましょう。第 1 引き数の長さが第 2 引き数と比べると十分に小さく、第 2 引き数の平均 75 % が探索されるものとします。この情報に基づくと、PERCENT_ARGBYTES は 75 に設定されます。この平均 75 % の見積もりは、以下に示す追加の前提事項に基づいています。

- 2 回に 1 回は、最初の引き数は検出されず、2 番目の引き数全体が探索されることになります。
- 最初の引き数は 2 番目の引き数内のどの場所にも現れる可能性があるため、最初の引き数が検出されるとき、平均して 2 番目の引き数の半分が探索される結果となります。

INITIAL_INSTS または INITIAL_IOS を使用すると、最初または最後に関数が呼び出されるときにだけ実行されるマシン語命令または読み取り/書き込み要求の数の見積もりを記録することができます。

ユーザー定義関数によって使用される入出力と命令についての情報を得るには、プログラム言語コンパイラーによって、またはオペレーティング・システムで使用可能なモニター・ツールによって提供される出力を使用することができます。

関連概念:

- 121 ページの『カタログ統計の表』
- 143 ページの『手動でのカタログ統計更新の一般規則』

モデル化および what-if の計画のカタログ統計

表および索引の実際の状態を反映しないものの、計画を目的としてデータベースに行うことのできるさまざまな変更を調べられるように、システム・カタログ内の統計情報を変更できます。選択されたシステム・カタログを更新できるという機能によって、次のことが可能になります。

- 開発システム上で、実動システム統計を使って照会パフォーマンスをモデル化します。
- 「what-if」照会パフォーマンス分析を実行します。

実動システムでの手動による統計の更新は行わないでください。そのようにすると、動的 SQL を含む実動照会で最適なアクセス・プランを、オプティマイザーが選択できません。

要件

表と索引およびそれらのコンポーネントの統計を変更するには、データベースに対する明示的な DBADM 権限が必要です。つまり、そのユーザー ID が DBADM 権限を持つものとして SYSCAT.DBAUTH 表に記録されている場合は、これらの統計を更新することができます。 DBADM グループに属していることは、明示的にこの権限を付与することにはなりません。 DBADM はすべてのユーザーの統計行を見ることができ、 SYSSTAT スキーマで定義されているビューに対して SQL UPDATE ステートメントを実行して、それらの統計列の値を更新できます。

DBADM 権限のないユーザーは、 CONTROL 特権を持つオブジェクトの統計が入った行しか表示できません。 DBADM 権限がない場合、各データベース・オブジェクトに対する以下の権限があれば、それぞれのオブジェクトの統計を変更できます。

- 表に対する明示的な CONTROL 特権。この表の列と索引に関する統計を更新することもできます。
- フェデレーテッド・データベース・システム内のニックネームに対する明示的な CONTROL 特権。これらのニックネームの列と索引に関する統計を更新することもできます。更新が影響を与えるのは、ローカル・メタデータだけであることに注意してください (データ・ソース表統計は変更されません)。 これらの変更が影響を与えるのは、 DB2® オプティマイザーが生成するグローバル・アクセス戦略だけです。
- ユーザー定義関数 (UDF) の所有権

以下に、EMPLOYEE 表の表統計を更新する例を示します。

```
UPDATE SYSSTAT.TABLES
SET  CARD   = 10000,
     NPAGES = 1000,
     FPAGES = 1000,
     OVERFLOW = 2
WHERE TABSCHEMA = 'userid'
  AND TABNAME   = 'EMPLOYEE'
```

カタログ統計を手動で更新するときは慎重に行ってください。軽率な変更により、以降の照会のパフォーマンスに対して重大な影響が及ぶ可能性があります。テストまたはモデル化のために使用する非実動データベースでも、以下の方法のいずれかを使用して、それらの表に適用された更新をリフレッシュし、統計を整合状態にすることができます。

- 変更が加えられた作業単位の ROLLBACK を実行します (その作業単位がコミットされていないことが前提です)。
- RUNSTATS ユーティリティーを使用すると、カタログ統計を計算し直し、最新表示します。
- カタログ統計を更新して、その統計が収集されていないことを示します。 (たとえば、列 NPAGES を -1 に設定すると、ページ数の統計は収集されなかったことが示されます。)
- カタログ統計を更新前のデータで置換します。この方法は、*db2look* ツールを使って、変更前の統計をキャプチャーする場合に限り使えます。

場合によっては、オプティマイザーが特定の統計値か値の組み合わせを無効と判断する場合があります。デフォルト値が使用されて、警告が出されます。しかし、統計の更新時には大部分の妥当性検査が行われるので、このような状況はまれです。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』
- 138 ページの『ユーザー定義関数の統計』
- 141 ページの『実動データベースのモデル化の統計』

実動データベースのモデル化の統計

テスト・システムに、実動システムのデータのサブセットを入れる必要が生じる場合があります。しかし、テスト・システムのカタログ統計と構成パラメーターを実動システムのカタログ統計と構成パラメーターと一致するように更新したのでない限り、この種のテスト・システムで選択されたアクセス・プランは、実動システムで選択されるアクセス・プランと必ずしも同じにはなりません。

生産性向上ツール *db2look* を実動データベースに対して実行すると、テスト・データベースのカタログ統計を実動のものと一致させるのに必要な更新ステートメントを生成することができます。これらの更新ステートメントを生成する場合は、模擬モード (`-m` オプション) で *db2look* を使用します。そのようにした場合、*db2look* は、実動データベースのカタログ統計を模造するのに必要なステートメントをすべて含むコマンド・プロセッサー・スクリプトを生成します。これは、テスト環境で Visual Explain を使用して SQL ステートメントを分析するときに便利です。

db2look -e を用いて DDL ステートメントを抽出すると、表、ビュー、索引およびデータベース内の他のオブジェクトを含むデータベース・データ・オブジェクトを再作成することができます。このコマンドから作成されたコマンド・プロセッサー・スクリプトを別のデータベースに対して実行すると、データベースを再作成することができます。データベースを再作成して統計を設定するスクリプトで、 `-e` オプションと `-m` オプションを一緒に使用できます。

`db2look` によって生成された更新ステートメントをテスト・システムに対して実行したなら、テスト・システムを使用して、実動で生成されるアクセス・プランを妥当性検査できるようになります。オプティマイザーが表スペースのタイプと構成を使用して入出力コストの見積もりを行うので、テスト・システムには、同じ表スペース形状つまり表スペース・レイアウトがなければなりません。すなわち、同じタイプ (SMS か DMS のいずれか) のコンテナーが同じ数だけなければなりません。

db2look ツールは、 *bin* サブディレクトリーにあります。

この生産性向上ツールの使用方法について知りたい場合は、コマンド行で次のように入力してください。

```
db2look -h
```

コントロール・センターにも、この *db2look* ユーティリティー (「SQL の生成 - オブジェクト名 (Generate SQL - Object Name)」と呼ばれる) へのインターフェースが備わっています。コントロール・センターを使用すると、このユーティリティー

からの結果ファイルをスクリプト・センターに統合することができます。 コントロール・センターから、 *db2look* コマンドをスケジュールすることもできます。コントロール・センターを使用する場合の違いは、 *db2look* コマンドを使用する場合は、 1 つの呼び出しで最大 30 の表を分析できるのに対して、 1 つの表の分析しか行えないということです。コントロール・センターからは、 LaTex および図形出力はサポートされていないことも知っておく必要があります。

db2look ユーティリティーは OS/390 または z/OS データベースに対して実行することもできます。 db2look ユーティリティーは、 OS/390 オブジェクトの DDL および UPDATE 統計ステートメントを抽出します。これは、OS/390 または z/OS オブジェクトを抽出して、 DB2® Universal Database (UDB) データベース内にそのオブジェクトを再作成したい場合に非常に役立ちます。

DB2 UDB の統計と OS/390 の統計の間にはいくらかの違いがあります。 db2look は、適切な場合に DB2 for OS/390 and z/OS から DB2 UDB への適切な変換を実行し、 DB2 for OS/390 で対応するものが存在しない DB2 UDB の統計についてはデフォルト値 (-1) を設定します。以下に、db2look ユーティリティーが、 DB2 for OS/390 and z/OS の統計を DB2 UDB の統計にマップする方法を示します。以下の説明で、「UDB_x」は DB2 UDB の統計列を、「S390_x」は DB2 for OS/390 and z/OS の統計列を表します。

1. 表レベル統計。

   UDB_CARD = S390_CARDF
   UDB_NPAGES = S390_NPAGES

   S390_FPAGES はありません。ただし、DB2 for OS/390 and z/OS には、PCTPAGES という別の統計があり、表の行を含んでいるアクティブな表スペース・ページのパーセンテージを表します。それで、以下のように、S390_NPAGES および S390_PCTPAGES に基づいて UDB_FPAGES を計算することができます。

   ```
   UDB_FPAGES=(S390_NPAGES * 100)/S390_PCTPAGES
   ```

   UDB_OVERFLOW にマップする S390_OVERFLOW はありません。そのため、db2look ユーティリティーは、この値をデフォルト値に設定します。

   ```
   UDB_OVERFLOW=-1
   ```

2. 列レベル統計。

   UDB_COLCARD = S390_COLCARDF
   UDB_HIGH2KEY = S390_HIGH2KEY
   UDB_LOW2KEY = S390_LOW2KEY

   UDB_AVGCOLLEN にマップする S390_AVGCOLLEN がないため、 db2look ユーティリティーは、この値をデフォルト値に設定します。

   ```
   UDB_AVGCOLLEN=-1
   ```

3. 索引レベル統計。

   UDB_NLEAF = S390_NLEAF
   UDB_NLEVELS = S390_NLEVELS
   UDB_FIRSTKEYCARD= S390_FIRSTKEYCARD

UDB_FULLKEYCARD = S390_FULLKEYCARD
UDB_CLUSTERRATIO= S390_CLUSTERRATIO

OS/390 または z/OS で対応するものがない他の統計は、デフォルトに設定されます。つまり、以下のようになります。

```
UDB_FIRST2KEYCARD = -1
UDB_FIRST3KEYCARD = -1
UDB_FIRST4KEYCARD  = -1
UDB_CLUSTERFACTOR = -1
UDB_SEQUENTIAL_PAGES = -1
UDB_DENSITY  = -1
```

4. 列分布統計。

   DB2 for OS/390 and z/OS の SYSIBM.SYSCOLUMNS には、 2 つのタイプの統計があります。頻出値を表す「F」と、カーディナリティーを表す「C」です。DB2 UDB に適用可能なのはタイプ「F」の項目だけです。考慮されるのは、これらの項目です。

   ```
   UDB_COLVALUE = S390_COLVALUE
   UDB_VALCOUNT = S390_FrequencyF * S390_CARD
   ```

   また、DB2 for OS/390 の SYSIBM.SYSCOLUMNS には列 SEQNO がありません。これは DB2 for UDB で必要なため、 db2look により自動的に生成されます。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』
- 139 ページの『モデル化および what-if の計画のカタログ統計』
- 143 ページの『手動でのカタログ統計更新の一般規則』

関連資料:

- 「コマンド・リファレンス」の『db2look - DB2 統計および DDL 抽出ツール・コマンド』

手動でのカタログ統計更新の一般規則

カタログ統計を更新する場合、一般的な規則のうち最も重要なのは、各種統計の有効な値、範囲、および形式を確実に統計ビューに保管するということです。さらに、各種統計相互間のリレーションシップの一貫性を保つことも重要です。

たとえば、SYSSTAT.COLUMNS 内の COLCARD は SYSSTAT.TABLES 内の CARD より小さくなければなりません (列内の固有値の数は、行の数より多くすることはできません)。ここで、COLCARD を 100 から 25 に減らし、CARD を 200 から 50 に減らしたいと仮定します。このとき最初に SYSCAT.TABLES を更新したとすると、エラーを受け取ることになります (CARD が COLCARD より小さくなってしまうため)。正しい順序は、最初に SYSCAT.COLUMNS 内の COLCARD を更新し、その後で SYSSTAT.TABLES 内の CARD を更新するという順序です。逆に COLCARD を 100 から 250 に増やし、CARD を 200 から 300 に増やしたい場合にも、同じことが言えます。その場合には、最初に CARD を更新してから、その後で COLCARD を更新しなければなりません。

更新された統計と別の統計との間に矛盾が検出された場合には、エラーが出されます。ただし、矛盾が生じたときに必ずエラーが出されるとは限りません。特に 2 つの関連する統計が別々のカタログにある場合など、状況によっては、矛盾を検出したりエラーを報告したりするのが難しいことがあります。したがって、こういった矛盾を起こさないように十分注意が必要です。

カタログ統計を更新する前に検査する必要のある規則のうち、最も一般的なのは次のものです。

1. 数値統計は -1 もしくは 0 以上でなければなりません。
2. パーセンテージを表す数値統計 (たとえば SYSSTAT.INDEXES 内の CLUSTERRATIO) は、 0 ～ 100 の範囲でなければなりません。

注: 行タイプの場合、表レベルの統計 NPAGES、FPAGES、および OVERFLOW は、副表に対して更新されません。

関連概念:

- 121 ページの『カタログ統計の表』
- 138 ページの『ユーザー定義関数の統計』
- 139 ページの『モデル化および what-if の計画のカタログ統計』
- 141 ページの『実動データベースのモデル化の統計』
- 144 ページの『手動での列統計の更新の規則』
- 145 ページの『手動での分散統計の更新の規則』
- 146 ページの『手動での表およびニックネーム統計の更新の規則』
- 147 ページの『手動での索引統計の更新の規則』

手動での列統計の更新の規則

SYSSTAT.COLUMNS 内の統計を更新する場合には、以下の指針に従ってください。

- SYSSTAT.COLUMNS の HIGH2KEY および LOW2KEY を手動で更新する場合、生成される値の性質は次のようになります。
  - HIGH2KEY、LOW2KEY の値は、対応するユーザー列のデータ・タイプの有効値でなければなりません。
  - HIGH2KEY、LOW2KEY 値の長さは、33 または目標列のデータ・タイプの最大長の、いずれか小さい方でなければなりません (引用符は含みません。これを含めるとストリング長は最大 68 になります)。つまり、HIGH2KEY、LOW2KEY 値の判別の際には、対応するユーザー列の値の最初の 33 文字だけが考慮されます。
  - HIGH2KEY/LOW2KEY 値は、UPDATE ステートメントの SET 文節でコスト計算の操作なしに使用できるような方法で格納されます。これは、文字ストリングの場合、単一引用符がストリングの先頭と最後に追加され、ストリング中の既存のすべての引用符に対しては余分の引用符が追加されるということです。 HIGH2KEY、LOW2KEY のユーザー列値および対応する値の例を以下の表に示します。

表 26. HIGH2KEY および LOW2KEY 値のデータ・タイプ

ユーザー列のデータ・タイプ	ユーザー・データ	対応する HIGH2KEY、LOW2KEY 値
INTEGER	-12	-12
CHAR	abc	'abc'
CHAR	ab'c	'ab''c'

  - HIGH2KEY は、対応する列に異なる値が 4 つ以上含まれるときは必ず LOW2KEY よりも大きい値にする必要があります。
- 列のカーディナリティー (SYSSTAT.COLUMNS 内の COLCARD 統計) は、対応する表のカーディナリティー (SYSSTAT.TABLES 内の CARD 統計) より大きくすることはできません。
- 列の NULL の数 (SYSSTAT.COLUMNS 内の NUMNULLS 統計) は、対応する表のカーディナリティー (SYSSTAT.TABLES 内の CARD 統計) より大きくすることはできません。
- データ・タイプが LONG VARCHAR、 LONG VARGRAPHIC、 BLOB、 CLOB、 DBCLOB の列の統計はサポートされていません。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』
- 139 ページの『モデル化および what-if の計画のカタログ統計』
- 143 ページの『手動でのカタログ統計更新の一般規則』

手動での分散統計の更新の規則

手動で分散統計を更新するのは、実動データベースをモデル化するときか、人為的に構成されたデータベースに対して what-if テストを実行する場合だけにしてください。実動データベースに対する分散統計の更新は行わないでください。

カタログ内のすべての統計が整合しているか確認してください。具体的には、各列ごとに、頻出データ統計と変位値のカタログ項目は、以下の制約を満たしていなければなりません。

- 頻出値統計 (SYSSTAT.COLDIST カタログ中)。この制約には次のものが含まれます。
  - 列 VALCOUNT の値は、SEQNO の値の増加に対して、不変または減少でなければなりません。
  - 列 COLVALUE の値の数は、その列の固有値の数 (この数は、カタログ・ビュー SYSSTAT.COLUMNS 内の列 COLCARD に保管されている) 以下でなければなりません。
  - 列 VALCOUNT の値の合計数は、その列の行の数 (カタログ表 SYSSTAT.TABLES の列 CARD に保管されているもの) 以下でなければなりません。
  - ほとんどの場合、列 COLVALUE の値は、その列の 2 番目に高いデータ値と 2 番目に低いデータ値 (それぞれカタログ表 SYSSTAT.COLUMNS の列

HIGH2KEY および LOW2KEY に保管されているもの) との間にあります。HIGH2KEY より大きい頻出値が 1 つと LOW2KEY より小さい頻出値が 1 つ存在してもかまいません。

- 変位数 (SYSSTAT.COLDIST カタログ内の)。この制約には次のものが含まれます。
  - 列 COLVALUE の値は、SEQNO の値の増加に対して、不変または減少でなければなりません。
  - 列 VALCOUNT の値は、SEQNO の値が増加するにつれて、単調増加でなければなりません。
  - 列 COLVALUE 内の最大値に対応する列 VALCOUNT の項目は、その列の行の数と等しいものでなければなりません。
  - ほとんどの場合、列 COLVALUE の値は、その列の 2 番目に高いデータ値と 2 番目に低いデータ値 (それぞれカタログ表 SYSSTAT.COLUMNS の列 HIGH2KEY および LOW2KEY に保管されているもの) との間にあります。

行の数が「R」個である列 C1 で分散統計が使用可能である場合に、データ値の相対比率を同じに保ったまま、行数を「(F × R)」にした列に対応するように統計を変更したいものとします。頻出値統計を F 倍するには、列 VALCOUNT の各項目を F 倍しなければなりません。同様に、変位値を F 倍するには、列 VALCOUNT 内の各項目を F 倍しなければなりません。これらの規則に従わないなら、照会の実行時にオプティマイザーが誤ったフィルター要因を使用することになり、予測不能なパフォーマンスになる可能性があります。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』
- 139 ページの『モデル化および what-if の計画のカタログ統計』
- 143 ページの『手動でのカタログ統計更新の一般規則』

手動での表およびニックネーム統計の更新の規則

SYSTAT.TABLES で更新できる統計値は、
CARD、FPAGES、NPAGES、OVERFLOW だけです。 MDC 表の場合は
ACTIVE_BLOCKS も含まれます。更新の際には、以下に留意してください。

1. CARD は、その表に対応する SYSSTAT.COLUMNS 内のすべての COLCARD 値以上でなければなりません。
2. CARD は、NPAGES より大きくなければなりません。
3. FPAGES は、NPAGES より大きくなければなりません。
4. NPAGES は、(この統計が索引に関連している場合) どの索引の PAGE_FETCH_PAIRS 列内のどの「取り出し (Fetch)」値よりも小さいか、等しくなければなりません。
5. CARD は、(この統計が索引に関連している場合) どの索引の PAGE_FETCH_PAIRS 列内のどの「取り出し (Fetch)」値よりも大きくなければなりません。

フェデレーテッド・データベース・システムで作業している場合、リモート・ビューに対するニックネームについての統計を手操作で提供または更新するときには、

注意が必要です。 ニックネームが戻す行数などの統計情報は、このリモート・ビューの評価にかかる実際のコストを反映していないことがあるので、 DB2® オプティマイザーが正しく動作しないことがあります。統計の更新が役立つ状況には、リモート・ビューが SELECT リスト上で列関数が適用されていない単一の基本表上に定義されている場合が含まれます。複合ビューでは、それぞれの照会の調整が必要な複合チューニング・プロセスが必要になることがあります。 DB2 オプティマイザーがビューのコストをより正確に導き出す方法を知ることができるように、ニックネームに対してローカル・ビューを作成することを考慮してください。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』
- 139 ページの『モデル化および what-if の計画のカタログ統計』
- 143 ページの『手動でのカタログ統計更新の一般規則』

手動での索引統計の更新の規則

SYSSTAT.INDEXES 内の統計を更新する場合には、以下で説明する規則に従ってください。

1. PAGE_FETCH_PAIRS (SYSSTAT.INDEXES 内にある) は、以下の規則に従う必要があります。
   - PAGE_FETCH_PAIRS 統計内の個々の値は、一連のブランク区切り文字によって区切る必要があります。
   - PAGE_FETCH_PAIRS 統計内の個々の値は 10 桁より長くすることはできず、かつ最大整数値 (MAXINT = 2147483647) より小さい値でなければなりません。
   - CLUSTERFACTOR が 0 より大きい場合は、必ず有効な PAGE_FETCH_PAIRS 値が存在していなければなりません。
   - 単一の PAGE_FETCH_PAIR 統計に含まれるのは、ちょうど 11 対でなければなりません。
   - PAGE_FETCH_PAIRS のバッファー・サイズ項目は、値の昇順で並んでいなければなりません。
   - PAGE_FETCH_PAIRS 項目のバッファー・サイズ値を、 32 ビット・オペレーティング・システムの場合は MIN(NPAGES, 524287)、 64 ビット・オペレーティング・システムの場合は MIN(NPAGES, 2147483647) より大きくすることはできません (NPAGES は、対応する表 (SYSSTAT.TABLES) のページ数)。
   - PAGE_FETCH_PAIRS の「取り出し」項目は、値が降順でなければなりません。この場合、個々の「取り出し」項目は NPAGES 以上です。PAGE_FETCH_PAIRS 項目内の「取り出し」サイズ値は、対応する表の CARD (カーディナリティー) 統計より大きくすることはできません。
   - バッファー・サイズの値が 2 つの連続した対の値と同じ場合は、ページ取り出しの値も両方の対 (SYSSTAT.TABLES 中) の値と同じでなければなりません。

   有効な PAGE_FETCH_UPDATE は次のとおりです。

```
PAGE_FETCH_PAIRS =
 '100 380 120 360 140 340 160 330 180 320 200 310 220 305 240 300
  260 300 280 300 300 300'
```

ここで、

```
NPAGES = 300
CARD   = 10000
CLUSTERRATIO  =  -1
CLUSTERFACTOR = 0.9
```

2. CLUSTERRATIO と CLUSTERFACTOR (SYSSTAT.INDEXES 内にある) は、以下の規則に従わなければなりません。
   - CLUSTERRATIO の有効な値は -1、または 0 と 100 の間です。
   - CLUSTERFACTOR の有効な値は -1、または 0 と 1 の間です。
   - CLUSTERRATIO 値と CLUSTERFACTOR 値のうち少なくとも 1 つは、常に -1 でなければなりません。
   - CLUSTERFACTOR が正の値の場合は、有効な PAGE_FETCH_PAIR 統計が伴わなければなりません。
3. 以下の規則は、 FIRSTKEYCARD、 FIRST2KEYCARD、 FIRST3KEYCARD、 FIRST4KEYCARD、および FULLKEYCARD に適用されます。
   - FIRSTKEYCARD は、単一列索引の場合には FULLKEYCARD と等しくなければなりません。
   - FIRSTKEYCARD は、対応する列の COLCARD (SYSSTAT.COLUMNS 内) と等しくなければなりません。
   - これらの索引統計のいずれかが必要ない場合には、それらを -1 に設定する必要があります。たとえば、索引に 3 行しか列がない場合には、FIRST4KEYCARD を -1 に設定します。
   - 複数の列索引の場合、すべての統計が必要ならば、それらの間の関係は以下のようにしなければなりません。

     ```
     FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
       <= FULLKEYCARD <= CARD
     ```
4. 以下の規則は、SEQUENTIAL_PAGES および DENSITY に適用されます。
   - SEQUENTIAL_PAGES の有効な値は -1、または 0 と NLEAF の間です。
   - DENSITY の有効な値は -1、または 0 と 100 の間です。

関連概念:

- 109 ページの『カタログ統計』
- 121 ページの『カタログ統計の表』
- 139 ページの『モデル化および what-if の計画のカタログ統計』
- 143 ページの『手動でのカタログ統計更新の一般規則』

第 6 章 SQL コンパイラーに関する解説

SQL 照会をコンパイルすると、選択されたアクセス・プランが実行されるか、またはそれがシステム・カタログに保管されるまでに、いくつかのステップが実行されます。

パーティション・データベース環境の場合には、SQL コンパイラーが SQL 照会上で実行する作業はすべて、接続しているデータベース・パーティションで行われます。コンパイルされた照会は、実行される前に、データベース内のすべてのデータベース・パーティションに送信されます。

この章のトピックでは、 SQL コンパイラーが SQL ステートメントをコンパイルして最適化する方法についての詳細を説明しています。

SQL コンパイラーの処理

SQL コンパイラーは、いくつかのステップを実行して、実行可能なアクセス・プランを作成します。このステップを以下の図に示し、図の下のセクションで説明します。一部のステップは、フェデレーテッド・データベースの照会にのみ実行されることにご注意ください。

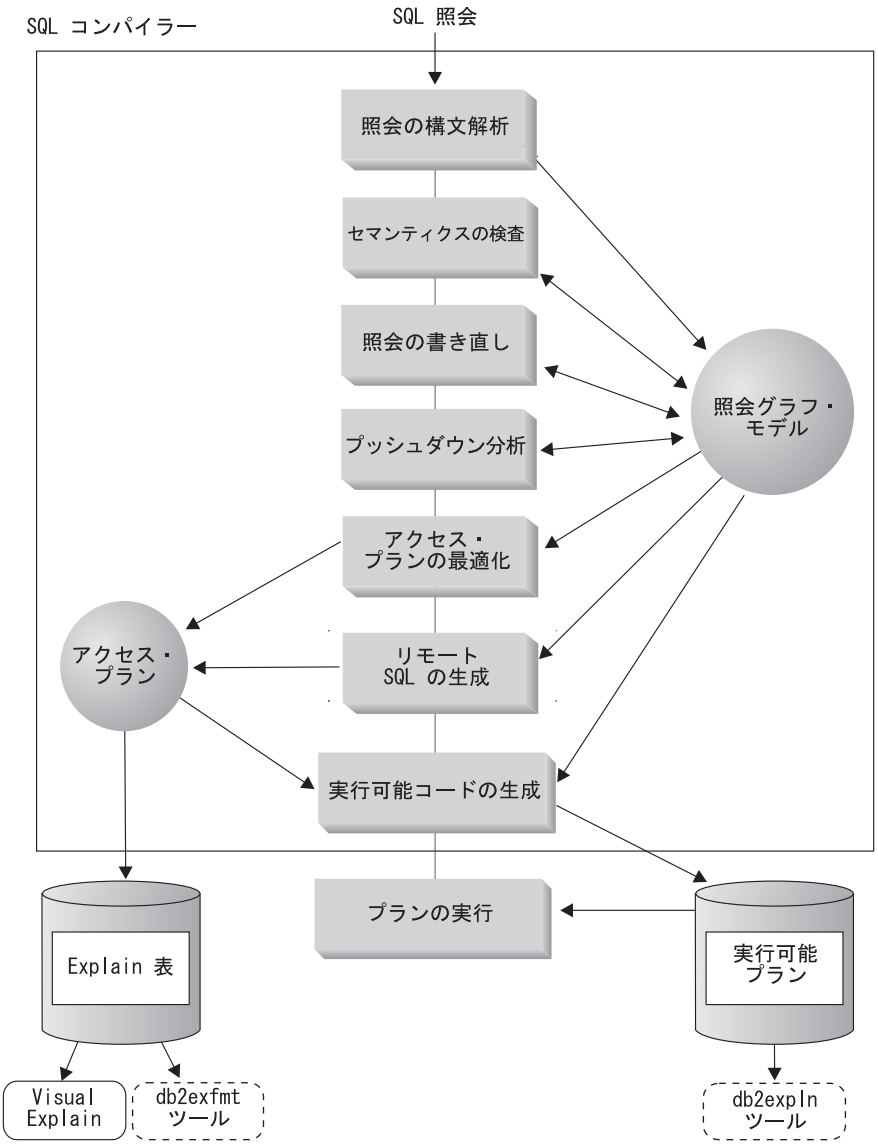


図 12. SQL コンパイラーによって実行されるステップ

照会グラフ・モデル

照会グラフ・モデル とは、以下で説明するステップで処理される照会を表示するメモリー内の内部的データベースです。

1. **照会の構文解析**

   SQL コンパイラーは、SQL 照会を分析して構文の妥当性検査を行います。構文エラーが検出されると、SQL コンパイラーは処理を停止し、照会をサブミットしたアプリケーションに対して該当する SQL エラーを戻します。構文解析が完了すると、照会の内部表示が作成され、照会グラフ・モデルに保管されます。

2. **セマンティクスの検査**

コンパイラーは、ステートメントのパーツ間に不整合がないことを確認します。セマンティクス検査の 1 つの簡単な例として、コンパイラーは、 YEAR スカラー関数用に指定した列のデータ・タイプが日時データ・タイプであるか検査します。

コンパイラーはまた、機能上のセマンティクスを照会グラフ・モデルに追加します。それには、参照制約、表チェック制約、トリガー、およびビューなどの結果が含まれます。照会グラフ・モデルには、照会ブロック、副照会、相関、派生表、式、データ・タイプ、データ・タイプ変換、コード・ページ変換、およびパーティション・キーを含む、照会のセマンティクスすべてが含まれます。

3. **照会の書き直し**

   コンパイラーは、照会グラフ・モデルに保管されているグローバルなセマンティクスを使用して、照会をもっと最適化しやすい形に変形し、その結果を照会グラフ・モデルに保管します。

   たとえば、コンパイラーは述部を移動することにより、適用されるレベルを変更し、照会のパフォーマンスを改善することがあります。このタイプの操作のことを、*一般述部プッシュダウン* といいます。パーティション・データベース環境では、以下の照会操作では、計算量が多くなります。

   - 集約
   - 行の再配分
   - 副照会の外側にある表の列への参照を含む副照会である相関副照会

   パーティション化された環境での一部の照会では、照会の書き直しの一環として非相関化が行われます。

4. **プッシュダウン分析 (フェデレーテッド・データベース)**

   このステップの主な作業は、データ・ソースにおいて、ある操作をリモートで評価 (プッシュダウン) できるかどうかを、オプティマイザーに通知することです。このタイプのプッシュダウン・アクティビティーは、データ・ソース照会に特有のものであり、一般の述部のプッシュダウン操作を拡張するものとなります。

   フェデレーテッド・データベース照会を実行するのでなければ、このステップは関係ありません。

5. **アクセス・プランの最適化**

   コンパイラーのうちのオプティマイザーの部分は、照会グラフ・モデルを入力データとして使用して、照会に答えるための多数の代替実行プランを生成します。各代替プランの実行コストを見積もるため、オプティマイザーは表、索引、列、および関数の統計を使用します。そして、見積実行コストの最も低いプランを選択します。オプティマイザーは、照会グラフ・モデルを使用して照会のセマンティクスを分析したり、索引、基本表、派生表、副照会、相関、および再帰を含め、広範囲にわたる要因に関する情報を獲得したりします。

オプティマイザーの部分では、別のタイプのプッシュダウン操作である、集約およびソート を考慮することもできます。この操作では、各操作の評価をデータ管理サービス・コンポーネントに知らせることにより、パフォーマンスを改善することができます。

さらにオプティマイザーでは、ページ・サイズの選択時に、別のサイズのバッファー・プールが存在するかどうかも考慮されます。環境にパーティション・データベースが含まれる場合には、そのことも、対称マルチプロセッサー (SMP) 環境で照会内並列処理の可能性のために選択されたプランを拡張する機能と同様に、考慮されます。この情報は、オプティマイザーが照会にとって最適のアクセス・プランを選択するのに使用されます。

コンパイラーのこのステップでの出力は、アクセス・プランです。このアクセス・プランは、Explain 表にキャプチャーされる情報となります。この情報は、アクセス・プランを生成するのに使われ、 Explain スナップショットによってキャプチャーできます。

6. **リモート SQL 生成 (フェデレーテッド・データベース)**

   オプティマイザーで選択した最終プランは、リモート・データ・ソースに対して実行される一連のステップで構成されています。リモート SQL 生成のステップでは、データ・ソースごとに実行される操作のために、データ・ソース固有の SQL ダイアレクトに基づく有効な SQL ステートメントが生成されます。

7. **「実行可能」コードの生成**

   最終ステップでは、コンパイラーはアクセス・プランと照会グラフ・モデルを使用して、照会の実行可能なアクセス・プラン、つまりセクションを作成します。このコード生成ステップでは、照会グラフ・モデルの情報を使用して、 1 つの照会で 1 回だけ計算するだけで済む式が繰り返し実行されないようにします。この最適化が行われる例としては、コード・ページ変換やホスト変数の使用を含むものがあげられます。

   ホスト変数、特殊レジスター、またはパラメーター・マーカーを持つ静的および動的 SQL ステートメントの (再) 最適化の照会を使用可能にするには、パッケージを REOPT BIND オプションを使ってバインドします。これを使うと、パッケージに属し、かつホスト変数、パラメーター・マーカー、または特殊レジスターを含んだ SQL ステートメントのアクセス・パスが、コンパイラーの選ぶデフォルトの推定値ではなく、これらの変数の値を使って最適化されます。照会の実行時に値が与えられてから、この最適化は実行されます。

   静的 SQL のアクセス・プランに関する情報は、システム・カタログ表に保管されます。パッケージが実行されると、データベース・マネージャーはシステム・カタログ表に保管されている情報を使用して、データのアクセス方法を決め、照会結果を提供します。この情報は、*db2expln* ツールによって使用されます。

注: たびたび変更されるテーブルでは、RUNSTATS を適切なインターバルで実行してください。オプティマイザーが最も効率的なアクセス・プランを作成するには、テーブルとそのデータに関する最新の統計情報が必要です。アプリケーションを再バインドして、更新済みの統計を利用してください。 RUNSTATS を実行しなかった (またはオプティマイザーが、 RUNSTATS が空かほぼ空の表に対して実行されたと想定している) 場合には、オプティマイザーは、デフォル

トを使用するか、あるいはディスク上に表を保管するのに使用されたファイル・ページ数 (FPAGES) に基づいて統計結果を引き出すように試みます。占有されたブロックの合計数が ACTIVE_BLOCKS 列に保管されます。

関連概念:

- 156 ページの『照会書き直しのメソッドとその例』
- 166 ページの『データ・アクセス方式』
- 173 ページの『述部の用語』
- 175 ページの『結合』
- 191 ページの『ソートとグループ化の影響』
- 193 ページの『パーティション内並列処理の最適化ストラテジー』
- 196 ページの『マテリアライズ照会表』
- 204 ページの『フェデレーテッド照会を評価する場合の分析のガイドライン』
- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『実行据え置きバインドの利点』
- 195 ページの『MDC 表の最適化ストラテジー』

照会の最適化に影響を与える構成パラメーター

構成パラメーターの中には、 SQL コンパイラーによって選択されるアクセス・プランに影響を与えるものがいくつかあります。それらのパラメーターの多くは単一パーティション・データベースに適用されるものですが、一部にはパーティション・データベースのみに適用されるものもあります。パーティション・データベースでは、各パラメーターに使用する値をすべてのパーティションで同じにする必要があります。

注: 構成パラメーターを動的に変更する場合、パッケージ・キャッシュにある以前のアクセス・プランのために、オプティマイザーが変更されたパラメーター値を即時には読み取れない可能性があります。パッケージ・キャッシュをリセットするには、 FLUSH PACKAGE CACHE コマンドを実行してください。

フェデレーテッド・システムで、照会の大部分がニックネームにアクセスする場合、環境を変更する前に、送信する照会のタイプを評価してください。たとえば、フェデレーテッド・データベースでは、 DBMS およびフェデレーテッド・システム内のデータといったデータ・ソースから、バッファー・プールがページをキャッシュに入れることはありません。このため、バッファーのサイズを増やしても、オプティマイザーがニックネームを含む照会のアクセス・プランを選択する際に、追加のアクセス・プランの選択肢を考慮するとは限りません。しかし、オプティマイザーは、データ・ソース表をローカルにマテリアライズすることが、最もコストを低くする手段、またはソート操作に必要なステップであると判断することがあります。この場合、 DB2® Universal Database に使用可能なリソースを増やすことで、パフォーマンスが向上します。

以下の構成パラメーターまたは要因は、 SQL コンパイラーによって選択されるアクセス・プランに影響を与えます。

- 作成時、または変更時に指定したバッファー・プールのサイズ。

オプティマイザーはアクセス・プランを選択するとき、ディスクからページをバッファー・プールに取り出すときの入出力コストを考慮し、照会を満たすために必要な入出力の数を見積もります。見積もりには、バッファー・プール使用率の予測も含まれています。すでにバッファー・プールの中にあるページに含まれている行を読み取るには、追加の物理的な入出力が必要ではないためです。

オプティマイザーは、 BUFFERPOOLS システム・カタログ表の、およびパーティション・データベースでは BUFFERPOOLDBPARTITION システム・カタログ表の *npages* 列の値を考慮します。

表を読み取るときの入出力コストは、以下のものに影響を与える可能性があります。
– 2 つの表の結合方法
– クラスター化されていない索引を使ってデータを読み取るかどうか

- デフォルトのパーティション内並行度 (dft_degree)

  `dft_degree` 構成パラメーターは、 CURRENT DEGREE 特殊レジスターおよび DEGREE BIND オプションのデフォルト値を指定します。値 1 は、パーティション内並列処理でないことを意味しています。値 -1 は、プロセッサー数と照会のタイプに基づいて、パーティション内並列処理の多重度をオプティマイザーが決定することを意味します。

- デフォルトの照会最適化クラス (dft_queryopt)

  照会最適化クラスは SQL 照会のコンパイル時に指定することができますが、デフォルトの最適化の多重度を設定することもできます。

  注: パーティション内並列処理は、 *intra_parallel* データベース構成パラメーターを設定して使用可能にしない限り行われません。

- アクティブ・アプリケーションの平均数 (avg_appls)

  SQL オプティマイザーは、選択されたアクセス・プランのランタイムにどれだけのバッファー・プールが使用可能かを見積もる助けとして、 *avg_appls* パラメーターを使用します。このパラメーターに高い値を指定すると、オプティマイザーがバッファー・プールをより控えめに使用するアクセス・プランを選択するという影響が出る可能性があります。値を 1 に指定すると、オプティマイザーはバッファー・プール全体がアプリケーションによって使用可能と見なします。

- ソート・ヒープ・サイズ (sortheap)

  ソートされる行が、ソート・ヒープ内で使用可能なスペースより多くのスペースを占める場合は、複数のソート・パスが実行され、各パスごとに行全体のうちの 1 つのサブセットがソートされることになります。各ソート・パスは、バッファー・プール内の一時表に保管され、ディスクに書き込むこともできます。すべてのソート・パスが完了すると、それらのソート済みサブセットはマージされ、ソート済みの単一の行集合になります。最終的にソートされたデータ・リストを保管するために一時表が必要ではない場合は、ソートは「パイプ処理」されるものと見なされます。つまり、ソートの結果を 1 つの順次アクセスで読み取ることができます。パイプ処理されたソートは、パイプ処理ではないソートの場合よりもパフォーマンスが向上するので、可能ならそれが使用されます。

アクセス・プランを選択するとき、オプティマイザーはソート操作のコストを見積もります。それには、ソートが次のものによってパイプ処理可能かどうかの評価も含まれます。
  - ソートするデータの量を見積もる。
  - *sortheap* パラメーターを見て、ソートのパイプ処理のために十分なスペースがあるかどうかを調べる。
- ロック・リスト用最大ストレージ (locklist) およびエスカレーション前のロック・リストの最大パーセント (maxlocks)

  分離レベルが**反復可能読み取り (RR)** である場合、 SQL オプティマイザーは *locklist* と *maxlocks* パラメーターの値を考慮して、行レベルのロックが表レベルのロックにエスカレーションされる可能性があるかどうかを判別します。オプティマイザーは、表アクセスに関してロック・エスカレーションが起きると見積もった場合、照会実行時のロック・エスカレーションのオーバーヘッドを生じさせる代わりに、そのアクセス・プランには表レベル・ロックを選択します。
- CPU 速度 (cpuspeed)

  SQL オプティマイザーは CPU 速度を使用して、特定の操作を実行するコストを見積もります。 CPU コストの見積もり、およびさまざまな入出力コストの見積もりは、照会に対して最適のアクセス・プランを選択するのに役立ちます。

  マシンの CPU 速度は、選択されるアクセス・プランに重大な影響を与える可能性があります。この構成パラメーターは、データベースをインストールまたは移行した時点で、自動的に適切な値に設定されます。このパラメーターは、テスト・システムにおいて実稼働環境のモデル化を行っている場合か、ハードウェア変更の影響を見積もっている場合でない限り、調整しないでください。このパラメーターを使用して異なるハードウェア環境のモデル化を行うと、その環境のために選択される可能性のあるアクセス・プランを検出することができます。 DB2 がこの自動構成パラメーターを再計算するようにするには、 -1 に設定してください。
- ステートメント・ヒープ・サイズ (stmtheap)

  ステートメント・ヒープのサイズは、オプティマイザーがさまざまなアクセス・パスを選択する際には影響がありませんが、複合 SQL ステートメントに関して実行される最適化の量には影響します。

  stmtheap パラメーターの設定値が十分大きくない場合は、使用可能なメモリーが足りないのでステートメントを処理できないことを示す SQL 警告を受け取ることがあります。 たとえば、 SQLCODE +437 (SQLSTATE 01602) により、ステートメントのコンパイルに使った最適化の量が、要求した量より少ないことを示す場合があります。
- 照会の最大並列処理多重度 (max_querydegree)

  max_querydegree パラメーターの値が ANY のときには、オプティマイザーが、使用する並列処理の多重度を選択します。 ANY 以外の値が示されている場合には、ユーザー指定の値がアプリケーションの並列処理の多重度を決定します。
- 通信スピード (comm_bandwidth)

通信スピードは、オプティマイザーがアクセス・プランを決めるのに使用されます。オプティマイザーはこのパラメーターの値を使用して、パーティション・データベースのデータベース・パーティション・サーバー間で一定の操作を実行する際のコストを見積もります。

関連概念:

- 149 ページの『SQL コンパイラーの処理』
- 166 ページの『データ・アクセス方式』
- 193 ページの『パーティション内並列処理の最適化ストラテジー』
- 195 ページの『MDC 表の最適化ストラテジー』

関連資料:

- 525 ページの『max_querydegree - 照会の最大並列処理多重度』
- 533 ページの『comm_bandwidth - 通信スピード』
- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
- 412 ページの『stmtheap - ステートメント・ヒープ・サイズ』
- 534 ページの『cpuspeed - CPU 速度』
- 438 ページの『avg_appls - アクティブ・アプリケーションの平均数』
- 503 ページの『dft_degree - デフォルト多重度』

照会の再書き込み

このセクションでは、オプティマイザーが照会を再書き込みしてパフォーマンスを向上させる方法について説明します。

照会書き直しのメソッドとその例

照会書き直し段階では、SQL コンパイラーは SQL ステートメントをより最適化しやすい形式に変換し、その結果として可能なアクセス・パスを改善することができます。照会書き直しは、多くの副照会または結合を伴う照会を含む、非常に複雑な照会の場合、特に重要です。照会生成プログラム・ツールはしばしばこの種の非常に複雑な照会を作成します。

SQL ステートメントに適用される照会書き直しの規則の数を変更するには、最適化クラスを変更します。照会書き直しの結果をいくつか表示させるには、 Explain 機能または Visual Explain を使用します。

照会の書き直しは、以下の 3 つの基本的な方法のいずれかで行えます。

- **操作のマージ**

  操作、特に SELECT 操作ができるだけ少ない照会を作成するため、 SQL コンパイラーは照会を書き直すことによって照会操作をマージします。以下の例に、マージ可能な操作をいくつか示します。

  – 例 - ビューのマージ

ビューを使用する SELECT ステートメントでは、表の結合順序が制限されたり、余分な表結合が生成されたりすることがあります。照会書き直し時にビューをマージすれば、これらの制限事項を除くことができます。

– 例 - 副照会から結合への変換

SELECT ステートメントに副照会が含まれている場合、表の配列処理の選択が制限される可能性があります。

– 例 - 余分な結合の除去

照会書き直し時には、余分な結合を除去して SELECT ステートメントを単純化することができます。

– 例 - 共用集約

照会がさまざまな関数を使用している場合は、書き直しによって、実行する必要のある計算の数を減らすことができます。

- **操作の移動**

照会を最小限の操作数と述部数で構成するため、コンパイラーは照会を書き直して照会操作を移動します。以下の例に、移動可能な操作をいくつか示します。

– 例 - DISTINCT の除去

照会書き直し時には、オプティマイザーで DISTINCT 操作の位置を移動して、その操作のコストを少なくすることができます。以降の例では、DISTINCT 操作は完全に除去されているケースを挙げています。

– 例 - 一般的な述部プッシュダウン

照会書き直し時に、オプティマイザーは述部を適用する順序を変更して、できるだけ早い機会により多くの選択述部が適用されるようにすることができます。

– 例 - 非相関化

パーティション・データベース環境にいる場合、データベース・パーティション間での結果セットの移動は高コストになります。他のデータベース・パーティションにブロードキャストする必要があるもののサイズ、またはブロードキャストの数 (あるいは、その両方) を減らすことは、照会の書き直し時の目標です。

- **述部の変換**

SQL コンパイラーは照会を書き直して、特定の照会に関して既存の述部をより最適化された述部に変換します。以下の例に、変換可能な述部をいくつか示します。

– 例 - 暗黙の述部の追加

照会の書き直し時に述部をその照会に追加することによって、オプティマイザーが照会の最適アクセス・プランを選択するときに、追加の表結合についても検討できるようになります。

– 例 - OR から IN への変換

照会書き直し時には、より効率的なアクセス・プランのために、OR 述部を IN 述部に変換することができます。また、IN 述部を OR 述部に変換すれば一層効率的なアクセス・プランが作成されるのであれば、 SQL コンパイラーでそのような変換をすることもできます。

関連概念:

- 158 ページの『コンパイラー書き直しの例: ビューのマージ』
- 160 ページの『コンパイラー書き直しの例: DISTINCT の除去』
- 162 ページの『コンパイラー書き直しの例: 暗黙の述部』
- 163 ページの『複数述部の列相関』

コンパイラー書き直しの例: ビューのマージ

EMPLOYEE 表に次の 2 つのビューでアクセスしているとしましょう。一方は高学歴の従業員を表示するビューで、他方は $35,000 以上の収入がある従業員を表示するビューです。

```
CREATE VIEW EMP_EDUCATION (EMPNO, FIRSTNME, LASTNAME, EDLEVEL) AS
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL
  FROM EMPLOYEE
 WHERE EDLEVEL > 17
CREATE VIEW EMP_SALARIES (EMPNO, FIRSTNAME, LASTNAME, SALARY) AS
SELECT EMPNO, FIRSTNME, LASTNAME, SALARY
  FROM EMPLOYEE
 WHERE SALARY > 35000
```

では、高学歴でかつ $35,000 以上の収入がある従業員をリストする照会を実行してみます。

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMP_EDUCATION E1,
       EMP_SALARIES  E2
 WHERE E1.EMPNO = E2.EMPNO
```

照会書き直し時に、これらの 2 つのビューをマージすると、次の照会が作成されます。

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

2 つのビューからの SELECT ステートメントをユーザー作成の SELECT ステートメントとマージすることによって、オプティマイザーはより多くのアクセス・プランの選択肢の中から選択できます。さらに、マージした 2 つのビューの使う基本表が同じである場合、追加の書き直しを実行できます。

例 - 副照会から結合への変換

SQL コンパイラーは、次のような副照会を含む照会がある場合、

```
SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO
FROM EMPLOYEE
WHERE WORKDEPT IN
      (SELECT DEPTNO
         FROM DEPARTMENT
        WHERE DEPTNAME = 'OPERATIONS')
```

これを次の形式の結合照会に変換します。

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME, PHONENO
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNAME = 'OPERATIONS'
```

一般に、結合は副照会を実行するよりはるかに効率的です。

例 - 余分な結合の除去

作成される、または生成される照会には、しばしば不要な結合が含まれています。照会書き直し段階で、次のような照会が生成される可能性もあります。

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

この照会では、SQL コンパイラーは結合を除去して、照会を次のように簡略化することができます。

```
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL, SALARY
  FROM EMPLOYEE
 WHERE EDLEVEL > 17
   AND SALARY  > 35000
```

次の例では、部門番号の EMPLOYEE サンプル表と DEPARTMENT サンプル表との間に、参照制約が存在すると想定しています。まずビューを作成します。

```
CREATE VIEW PEPLVIEW
   AS SELECT FIRSTNME, LASTNAME, SALARY, DEPTNO, DEPTNAME, MGRNO
         FROM EMPLOYEE E DEPARTMENT D
         WHERE E.WORKDEPT = D.DEPTNO
```

それから、次のような照会を作成します。

```
SELECT LASTNAME, SALARY
   FROM PEPLVIEW
```

この照会は、次のようになります。

```
SELECT LASTNAME, SALARY
   FROM EMPLOYEE
   WHERE WORKDEPT NOT NULL
```

この状態では、照会を書き直せることが分かっていても、基本表へのアクセス権がないため、書き直すことはできません。持っているのは、上記のビューへのアクセス権だけです。したがって、このタイプの最適化は、データベース・マネージャー内で実行する必要があります。

次のような場合、参照保全結合は冗長になる可能性があります。

- ビューが結合で定義される
- 照会が自動的に生成される

たとえば、ツールなどが自動生成したクエリーは最適でないこともあります。

例 - 共用集約

照会の中で複数の関数を使用すると、複数の計算が生成されますが、これは場合によってはかなり時間を要します。照会内で行う計算の数を減らすことによって、計画を改善することができます。たとえば、以下のような複数の関数を使用する照会があるとします。

```
SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  AVG(SALARY+BONUS+COMM) AS OAVG,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE;
```

SQL コンパイラーは、この照会を次のように変換します。

```
SELECT OSUM,
  OSUM/OCOUNT
  OCOUNT
FROM (SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE) AS SHARED_AGG;
```

この書き直しによって、照会の関数は、2 つの SUM と 2 つの COUNT から 1 つの SUM と 1 つの COUNT に減らされます。

関連概念:

- 149 ページの『SQL コンパイラーの処理』
- 156 ページの『照会書き直しのメソッドとその例』

コンパイラー書き直しの例: DISTINCT の除去

EMPNO 列を EMPLOYEE 表の主キーとして定義した場合、次の照会は、

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME
  FROM EMPLOYEE
```

DISTINCT 文節を除去することで書き直されます。

```
SELECT EMPNO, FIRSTNME, LASTNAME
  FROM EMPLOYEE
```

上記の例では、主キーが選択されているため、 SQL コンパイラーには返される各行がすでにユニークであることがわかっています。この場合、DISTINCT キーワードは不要です。照会を書き直さない場合は、オプティマイザーはソートなどの必要な処理を含む計画を作成して、列を明確にする必要があります。

例 - 一般的な述部プッシュダウン

述部が通常適用されるレベルを変更すると、パフォーマンスが向上する場合があります。たとえば、部署「D11」内の従業員全員のリストを示す以下のようなビューがあるとします。

```
CREATE VIEW D11_EMPLOYEE
 (EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM)
AS SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM
     FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
```

さらに、以下の照会があるとします。

```
SELECT FIRSTNME, PHONENO
  FROM D11_EMPLOYEE
 WHERE LASTNAME = 'BROWN'
```

コンパイラーの照会書き直し段階で、述部 LASTNAME = 'BROWN' がビュー D11_EMPLOYEE にプッシュされます。これにより、その述部が早い時期におそらく効率的に適用されるようになります。この例で実行可能な実際の照会は、次のようになります。

```
SELECT FIRSTNME, PHONENO
  FROM EMPLOYEE
 WHERE LASTNAME = 'BROWN'
   AND WORKDEPT = 'D11'
```

述部のプッシュダウンは、ビューに限定されません。述部がプッシュダウンされる可能性のあるこれ以外の状況には、 UNION、GROUP BY、派生表 (ネストされた表式や共通表式) があります。

例 - 非相関化

パーティション・データベース環境では、 SQL コンパイラーは次のような照会の書き直しを行う場合があります。

次の例では、プログラミング・プロジェクトに従事している従業員のうち給与が低い人をすべて検索します。

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
  FROM EMPLOYEE E, PROJECT P
WHERE P.EMPNO = E.EMPNO
  AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND E.SALARY+E.BONUS+E.COMM <
     (SELECT AVG(E1.SALARY+E1.BONUS+E1.COMM)
        FROM EMPLOYEE E1, PROJECT P1
        WHERE P1.PROJNAME LIKE '%PROGRAMMING%'
          AND P1.PROJNO = A.PROJNO
          AND E1.EMPNO = P1.EMPNO)
```

この照会は相関しており、また PROJECT と EMPLOYEE の両方が PROJNO 上にパーティション化されていないので、各プロジェクトが各データベース・パーティションにブロードキャストされます。さらに、この副照会の評価を何回も行わなければなりません。

SQL コンパイラーは、照会を以下に示すように書き直します。

- プログラミング・プロジェクトに従事している従業員の個別リストを決定して、それを DIST_PROJS とします。これが個別リストでなければならないのは、各プロジェクトに対して行われる集約が一度だけとなるようにするためです。

```
WITH DIST_PROJS(PROJNO, EMPNO) AS
(SELECT DISTINCT PROJNO, EMPNO
 FROM PROJECT P1
 WHERE P1.PROJNAME LIKE '%PROGRAMMING%')
```

- プログラミング・プロジェクトに従事している従業員の個別リストを使用して、これを従業員表に結合し、プロジェクトごとの平均報酬 AVG_PER_PROJ を求めます。

```
AVG_PER_PROJ(PROJNO, AVG_COMP) AS
(SELECT P2.PROJNO, AVG(E1.SALARY+E1.BONUS+E1.COMM)
 FROM EMPLOYEE E1, DIST_PROJS P2
 WHERE E1.EMPNO = P2.EMPNO
 GROUP BY P2.PROJNO)
```

- 最終的に、新しい照会は次のようになります。

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
 FROM PROJECT P, EMPLOYEE E, AVG_PER_PROG A
WHERE P.EMPNO = E.EMPNO
  AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND P.PROJNO = A.PROJNO
  AND E.SALARY+E.BONUS+E.COMM < A.AVG_COMP
```

書き直された SQL 照会では、プロジェクトごとの AVG_COMP (AVG_PRE_PROJ) を計算し、その結果を EMPLOYEE 表を含むデータベース・パーティションすべてにブロードキャストすることができます。

関連概念:

- 149 ページの『SQL コンパイラーの処理』
- 156 ページの『照会書き直しのメソッドとその例』

コンパイラー書き直しの例: 暗黙の述部

以下の照会は、「E01」に報告する部門のマネージャー、およびそれらのマネージャーが担当するプロジェクトのリストを生成するものです。

```
SELECT DEPT.DEPTNAME DEPT.MGRNO, EMP.LASTNAME, PROJ.PROJNAME
  FROM DEPARTMENT DEPT,
       EMPLOYEE   EMP,
       PROJECT    PROJ
 WHERE DEPT.ADMRDEPT = 'E01'
   AND DEPT.MGRNO = EMP.EMPNO
   AND EMP.EMPNO  = PROJ.RESPEMP
```

照会書き直しにより、以下の暗黙の述部が追加されます。

```
DEPT.MGRNO = PROJ.RESPEMP
```

この書き直しの結果、オプティマイザーがこの照会に最適のアクセス・プランを選択するとき、考慮できる結合の種類が増えることになります。

前述の述部移動のほかにも、照会書き直しでは、等号述部によって暗黙のうちに示される移動に基づいて、さらに別のローカル述部が導出されます。 たとえば、以下の照会は、部門 (部門番号が「E00」より大きいもの) の名前、およびその部門で働く従業員の名前のリストを作成するものです。

```
SELECT EMPNO, LASTNAME, FIRSTNAME, DEPTNO, DEPTNAME
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNO > 'E00'
```

この照会では、照会書き直し段階で、以下の暗黙の述部が追加されます。

```
EMP.WORKDEPT > 'E00'
```

この書き直しの結果として、オプティマイザーは結合する行の数を減らします。

例 - OR から IN への変換

OR 文節が、次の例のように同じ列にある 2 つ以上の単純等価述部を結合する場合を考えてみましょう。

```
SELECT *
  FROM EMPLOYEE
 WHERE DEPTNO = 'D11'
    OR DEPTNO = 'D21'
    OR DEPTNO = 'E21'
```

DEPTNO 列に索引が存在しない場合、OR 文節を次のような IN 述部に変換すると、照会をより効率的に処理できるようになります。

```
SELECT *
  FROM EMPLOYEE
 WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

注: 場合によっては、データベース・マネージャーは IN 述部を一連の OR 文節に変換して、索引 OR 処理を実行できるようにすることがあります。

関連概念:

- 149 ページの『SQL コンパイラーの処理』
- 156 ページの『照会書き直しのメソッドとその例』

複数述部の列相関

アプリケーションに 2 つ以上の結合述部が 2 つの表を結合するような結合で構成される照会がある場合があります。照会が別の表の似ていて関連した列の間のリレーションシップを決定する必要がある場合、これは珍しいことではありません。

たとえば、さまざまな色、伸縮性、品質の原料から製品を作るメーカーのことを考えてみましょう。完成品は、その原料と同じ色、伸縮性をしています。この場合、以下の照会を発行します。

```
SELECT PRODUCT.NAME, RAWMATERIAL.QUALITY FROM PRODUCT, RAWMATERIAL
   WHERE PRODUCT.COLOR       =  RAWMATERIAL.COLOR
     AND PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

この照会は、すべての製品の名前と原料の品質を戻します。以下の 2 つの結合述部があります。

```
PRODUCT.COLOR       =  RAWMATERIAL.COLOR
PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

オプティマイザーはこの照会を実行するプランを選択すると、上記の 2 つの述部の選択可能性をそれぞれ計算します。また 2 つの述部は独立していると想定します。つまり、それぞれの色ごとにすべてのレベルの伸縮性があり、また逆にそれぞれのレベルの伸縮性ごとにすべての色があると想定します。それから、表ごとに伸縮性のレベルの数と異なる色の数のカタログ統計情報を使用して、述部の対の全体的な選択可能性を見積もります。この見積もりに基づいて、たとえばネストしたループ結合とマージ結合のどちらを優先するかを選択します。

しかしながら、2 つの述部が独立していない場合もあります。たとえば、伸縮性の高い原料には 2、3 種類の色しかなく、伸縮性の非常に低い原料には伸縮性の高い原料の色とは別の 2、 3 色しかない場合も考えられます。この場合、2 つの述部を組み合わせた場合の選択可能性によって除去される行が少なくなるので、照会により戻される行は多くなります。極端なケースとしてそれぞれの色ごとに 1 つのレベルの伸縮性しかない場合やその逆の場合を考えてみてください。この場合、一方の述部は他方の述部によって暗黙指定されるので完全に省略することもできます。こうなるとオプティマイザーは最良のプランを選択することができない場合があります。たとえば、マージ結合の方が速いのに、ネスト・ループ結合プランを選択してしまうかもしれません。

他のデータベース製品の場合、データベース管理者はカタログ中の統計を更新して一方の述部の選択可能性を少なくすることによりこのパフォーマンス上の問題を解決していましたが、この方法は他の照会に思わぬ副次作用が及ぶことがありました。

DB2® UDB のオプティマイザーは、列に対して索引を定義する場合、あるいは適切な列のグループ列統計を収集して保守する場合、結合述部の相関の検出と補正を試みます。

たとえば、上記の伸縮性の例の場合、

```
PRODUCT.COLOR, PRODUCT.ELASTICITY
```

または

```
RAWMATERIAL.COLOR, RAWMATERIAL.ELASTICITY
```

またはこの両方のユニーク索引を定義することができます。

オプティマイザーが相関を検出できるように、この索引の非組み込み列は相関する列だけでなければなりません。索引だけのスキャンができるように、索引に組み込み列を含めることもできます。結合述部に 3 つ以上の相関列がある場合、そのすべての列を含むユニーク索引を定義してください。 1 つの表中の相関列がその表の主キーである場合がよくあります。主キーは常にユニークなので、それとは別にユニーク索引を定義する必要はありません。

正しく索引を作成した後、表の統計が最新のものになっていることを確認し、何らかの理由で (たとえば、オプティマイザーを制御しようとして) 手動で変更が加えられて本当の値でなくなっていたりしないか確認してください。

オプティマイザーはユニーク索引の統計表中の FIRSTnKEYCARD および FULLKEYCARD 列の情報を使用して相関事例を検出し、相関述部の組み合わせによる選択可能性を動的に調整するので、結合サイズとコストに関する見積もりの正確さが向上します。

別の方法としては、列のセットに関する列グループ統計を収集することができます。上記の伸縮性の例では、 PRODUCT.COLOR、PRODUCT.ELASTICITY または RAWMATERIAL.COLOR、RAWMATERIAL.ELASTCITY (あるいはその両方) 列に関する統計を収集できます。

列グループ統計は、RUNSTATS の「ON COLUMNS」オプションを使用して収集します。たとえば、PRODUCT.COLOR および PRODUCT.ELASTICITY に関する列グループ統計を収集するには、次の RUNSTATS コマンドを発行します。

```
RUNSTATS ON TABLE product ON COLUMNS ((color, elasticity))
```

単純な等価述部の相関

JOIN 述部相関に加えて、オプティマイザーはタイプ COL = *定数* の単純な等価述部に関する相関も管理します。たとえば、異なるタイプの車の表を考慮します。それぞれに、MAKE (製造メーカー)、MODEL、YEAR、COLOR、および STYLE (セダン、ステーション・ワゴン、スポーツ・カーなど) があります。 COLOR に関する述部は、 MAKE、MODEL、STYLE、または YEAR に関する述部から独立していると思われます。ほぼすべての製造メーカーは、毎年、それぞれのモデルおよびスタイルで標準色を使用できるようにするからです。しかし、特定の名前を持つモデルを製造するのは単一の車メーカーだけなので、述部 MAKE および MODEL が独立していないことは明らかです。複数の車メーカーによって同じモデル名が使用されることは非常にまれで、明らかに車メーカーもそれは望みません。

2 つの列 MAKE および MODEL の索引が存在するか、列グループ統計が収集される場合、オプティマイザーはその索引または列の統計情報を使用して、異なる値を結合した数値を判別し、2 つの列間の相関の選択可能性やカーディナリティーの見積もりを調整します。そうした述部が結合述部でない場合、オプティマイザーが調整を行うためのユニーク索引を持つ必要はありません。

関連概念:

- 149 ページの『SQL コンパイラーの処理』
- 156 ページの『照会書き直しのメソッドとその例』

REOPT BIND オプションを使用した照会最適化

ホスト変数、特殊レジスター、またはパラメーター・マーカーを持つ静的および動的 SQL ステートメントの照会最適化 (または再最適化) を使用可能にするには、REOPT BIND オプションを使ってパッケージをバインドしてください。このオプションを使用すると、パッケージに属し、かつホスト変数、パラメーター・マーカー、または特殊レジスターを含んだ SQL ステートメントのアクセス・パスが、コンパイラーの選ぶデフォルトの推定値ではなく、これらの変数の実際の値を使って最適化されます。照会の実行時に値が与えられてから、最適化は実行されます。

関連概念:

- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『静的 SQL における REOPT の影響』
- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『動的 SQL における REOPT の影響』

データ・アクセス方式

このセクションでは、照会が要求するデータにアクセスするためにオプティマイザーが選択できる方式について説明します。

データ・アクセス方式

SQL ステートメントをコンパイルするとき、 SQL オプティマイザーは照会を実現するさまざまな方法の実行コストを見積もります。見積もりに基づいて、オプティマイザーは最良のアクセス・プランを選択します。アクセス・プラン とは、 SQL ステートメントの解決に必要な操作の順序を指定するものです。アプリケーション・プログラムがバインドされると、パッケージ が作成されます。このパッケージには、そのアプリケーション・プログラムの静的 SQL ステートメント全部に関するアクセス・プランが入れられます。動的 SQL のアクセス・プランは、アプリケーション実行時に作成されます。

表のデータにアクセスするには 2 つの方法があります。

- 表全体を順番にスキャンする
- 最初に表の索引にアクセスすることにより、表の特定の行に位置付ける

照会が要求する結果を作り出すために、通常 WHERE 文節で記述される述部の条件にしたがって行を選択します。アクセスされた表の選択された行は結合されて結果セットを作り出します。この結果セットをさらにソートしたりグループ化して処理する場合もあります。

関連概念:

- 149 ページの『SQL コンパイラーの処理』
- 166 ページの『索引スキャンによるデータ・アクセス』
- 169 ページの『索引アクセスのタイプ』
- 172 ページの『索引アクセスとクラスター率』

索引スキャンによるデータ・アクセス

索引スキャン が行われるのは、データベース・マネージャーが以下のいずれかの理由で索引にアクセスするときです。

- 基本表にアクセスする前に、 (索引の特定の範囲内にある行をスキャンすることによって) 条件限定の行の集合の範囲を狭めるため。索引の*スキャン範囲* (スキャンの開始点と停止点) は、照会において索引列と比較する値によって判別されます。
- 出力を並べ替えるため。
- 要求した列データを直接取得するため。要求されたデータがすべて索引内にある場合、索引の対象である表にはアクセスする必要がありません。これは、*索引のみのアクセス* と呼ばれます。

索引が ALLOW REVERSE SCANS オプションで作成されていれば、スキャンは定義した方向とは逆方向に実行することもできます。

注: 適当な索引が作成されていない場合、または索引スキャンの方がコストがかかる場合、オプティマイザーは表スキャンを選択します。索引スキャンのコストが高くなり得るのは、表が小さい場合、索引クラスター率が低い場合、照会が表のほとんどの行を必要とする場合です。アクセス・プランが表スキャンと索引スキャンのどちらを使用するかを知るには、 SQL Explain 機能を使用してください。

範囲を区切るための索引スキャン

ある特定の照会に索引を使用できるかどうかを決定するとき、オプティマイザーは索引の各列を最初の列から順に評価し、 WHERE 文節の等式と他の述部を満足させるかどうかを調べます。*述部* は、WHERE 文節内で比較操作を明示する、または暗黙のうちに示す探索条件の 1 つのエレメントです。以下の場合に、述部は索引スキャンの範囲を区切るのに使用できます。

- 定数、ホスト変数、評価結果が定数になる式、またはキーワードに対して等しいかどうかがテストされます。
- 「IS NULL」または「IS NOT NULL」であるかどうかがテストされます。
- 基本的な副照会 (つまり、ANY、ALL、または SOME が含まれていない) についての等価性のテストであり、その副照会には即時親照会ブロック (つまり、この副照会が副選択となる SELECT) への相関列参照がない。
- 狭義および広義の不等比較をテストします。

以下の例で索引が範囲を制限するのに使用される場合を説明します。

- 索引が次のように定義されているとします。

  ```
  INDEX IX1:  NAME    ASC,
              DEPT    ASC,
              MGR     DESC,
              SALARY  DESC,
              YEARS   ASC
  ```

  この場合、以下の述部を使用して索引 IX1 のスキャンの範囲を制限することができます。

  ```
  WHERE  NAME  = :hv1
    AND  DEPT   = :hv2
  ```

  または

  ```
  WHERE  MGR  = :hv1
    AND  NAME = :hv2
    AND  DEPT = :hv3
  ```

  2 番目の WHERE 文節で、述部を索引内でキー列の現れる順序と同じ順序で指定する必要はありません。この例ではホスト変数を使用していますが、パラメーター・マーカー、式、または定数などの変数でも同じ効果があります。

- ALLOW REVERSE SCANS パラメーターを使用して作成された単一索引を考えてみましょう。こうした索引は、索引の作成時に定義した方向へのスキャンと、反対方向 (逆方向) へのスキャンをサポートします。ステートメントは、以下のようになります。

  ```
  CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
  ```

  この場合、索引 (iname) は、 cname の降順の (DESCending) 値に基づいて形成されます。逆方向スキャンを許可することにより、列に対する索引が降順でスキャンするように定義されているとしても、昇順でスキャンすることができます。索引を実際に両方向で使うことは、ユーザーが制御するのではなく、アクセス・プランの作成および考慮時にオプティマイザーによって制御されます。

以下の WHERE 文節では、NAME および DEPT の述部だけが索引スキャンの範囲を区切るのに使用され、 SALARY または YEARS の述部は使用されません。

```
WHERE  NAME  = :hv1
  AND  DEPT   = :hv2
  AND  SALARY = :hv4
  AND  YEARS  = :hv5
```

これは、これらの列を最初の 2 つの索引キー列から分離するキー列 (MGR) があるので、並べ替えがオフになるためです。しかし、いったん NAME = :hv1 および DEPT = :hv2 の述部によって範囲が決まったなら、残りの述部を残りの索引キー列に対して評価できるようになります。

不等比較をテストする索引スキャン

ある種の不等比較述部は索引スキャンの範囲を区切ることができます。不等比較述部には、以下の 2 つのタイプがあります。

- 狭義の不等比較述部

  範囲を区切る述部として使用できる狭義の不等比較オペレーターは、より大きい (>) とより小さい (<) です。

  索引スキャンの範囲を区切るのに考慮される狭義の不等比較述部は 1 つの列に対するものだけです。以下の例では、NAME 列と DEPT 列に対して述部を使用して範囲を区切ることはできますが、 MGR 列に対しては述部は使用できません。

  ```
  WHERE  NAME  = :hv1
    AND  DEPT  > :hv2
    AND  DEPT  < :hv3
    AND  MGR   < :hv4
  ```

- 広義の不等比較述部

  以下は、範囲を区切る述部として使用できる広義の不等比較オペレーターです。

  – >= と <=
  – BETWEEN
  – LIKE

  索引スキャンの範囲を区切る場合、広義不等比較述部については複数の列が考慮されます。次の例では、述部をすべて使用して索引スキャンの範囲を区切ることができます。

  ```
  WHERE  NAME  = :hv1
    AND  DEPT  >= :hv2
    AND  DEPT  <= :hv3
    AND  MGR   <= :hv4
  ```

  この例についてさらに考慮するために、:hv2 = 404、 :hv3 = 406、および :hv4 = 12345 を想定してください。データベース・マネージャーは部門 404 と 405 のすべての索引をスキャンしますが、 12345 より多い従業員数 (MGR 列) を持つ管理者を最初に検出した時点で、部門 406 のスキャンを停止します。

データ並び替えのための索引スキャン

照会がソートされた順序での出力を要求している場合、並び替える列が、最初の索引キー列から始まって連続して索引内に現れる場合は、データを並び替えるのに索引を使用することができます。並び替えまたはソートは、 ORDER BY、 DISTINCT、 GROUP BY、「= ANY」副照会、「> ALL」副照会、「< ALL」副照

会、 INTERSECT または EXCEPT、 UNION などの操作の結果得られます。例外は、索引キー列が「定数値」、つまり評価結果が定数の式に等しいかどうかを比較する場合です。この場合、並べ替えに使う列が最初の索引キー列以外のものである可能性があります。

次の照会を考えてみましょう。

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY MGR
```

この照会では、NAME も DEPT も必ず同じ値となり、結果としてその列についてはソース済みになるため、行を並び替えるのに索引を使用できます。つまり前述の WHERE 文節と ORDER BY 文節は次のものと同じです。

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY NAME, DEPT, MGR
```

このほかにもユニーク索引を使用することによって、ソート対象条件を省略することもできます。次の索引定義と ORDER BY 文節を考えてみましょう。

```
UNIQUE INDEX IX0:  PROJNO  ASC
SELECT PROJNO, PROJNAME, DEPTNO
   FROM PROJECT
ORDER BY PROJNO, PROJNAME
```

IX0 索引により PROJNO がユニークになるため、PROJNAME 列での順序付けは不要です。この固有性により、各 PROJNO 値には、PROJNAME 値が 1 つしかないことになります。

関連概念:

- 166 ページの『データ・アクセス方式』
- 26 ページの『索引の構造』
- 169 ページの『索引アクセスのタイプ』
- 172 ページの『索引アクセスとクラスター率』

索引アクセスのタイプ

照会が表から必要としているデータがすべて表の索引から取得できると、オプティマイザーが判断できる場合があります。他方、表にアクセスするのに複数の索引を使用する場合もあります。レンジ・クラスター表の場合は、データ・レコードの位置を計算する「仮想」索引を介してデータにアクセスできます。

索引のみのアクセス

場合によっては、表にアクセスせずに、索引からすべての必須データを検索できることがあります。これは、*索引のみの* アクセスと呼ばれます。

以下の索引定義を使って、索引のみのアクセスについて説明します。

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

基本表を読み取らずに、索引にアクセスするだけで、以下の照会を実行することができます。

```
SELECT NAME, DEPT, MGR, SALARY
   FROM EMPLOYEE
 WHERE NAME = 'SMITH'
```

しかしながら、しばしば必要な列が索引内にないことがあります。それらの列のデータを得るには、表の行を読み取らなければなりません。オプティマイザーが索引のみのアクセスを選べるようにするには、組み込み列付きでユニーク索引を作成します。たとえば、次の索引定義を考えてみてください。

```
CREATE UNIQUE INDEX IX1 ON EMPLOYEE
   (NAME ASC)
    INCLUDE (DEPT, MGR, SALARY, YEARS)
```

この索引は NAME 列を固有索引とし、しかも DEPT、 MGR、SALARY、および YEARS 列のデータも保管して維持します。これにより、次の照会は索引のみのアクセスで済んでしまいます。

```
SELECT NAME, DEPT, MGR, SALARY
   FROM EMPLOYEE
   WHERE NAME='SMITH'
```

しかし、索引に INCLUDE 列を追加することを考慮する場合、余分の記憶域スペースと保守コストに見合うメリットがあるかどうかを考えてください。こうした索引を読み取るだけで済む照会がめったに実行されないのなら、コストに見合うメリットがあるとは言えないかもしれません。

複数の索引アクセス

WHERE 文節の述部を実行するために、オプティマイザーが同じ表の複数の索引をスキャンすることを選択する場合があります。たとえば、以下の 2 つの索引定義を考えてみてください。

```
INDEX IX2:  DEPT    ASC
INDEX IX3:  JOB     ASC,
            YEARS   ASC
```

次の述部は上記の 2 つの索引を使用することにより実行できます。

```
WHERE DEPT = :hv1
  OR (JOB   = :hv2
  AND YEARS >= :hv3)
```

索引 IX2 をスキャンすることによって、 DEPT = :hv1 述部を満たす行 ID (RID) のリストが作成されます。また、索引 IX3 をスキャンすることによって、 JOB = :hv2 AND YEARS >= :hv3 述部を満たす RID のリストが作成されます。表にアクセスする前に、これらの 2 つの RID リストは組み合わせられて重複は除かれます。これは、*索引 OR 操作* と呼ばれます。

索引 OR 操作は、次の例のように IN 文節で指定されている述部でも使用される場合があります。

```
WHERE DEPT IN (:hv1, :hv2, :hv3)
```

索引 OR 操作の目的は、重複した RID を除去することですが、*索引 ANDing 結合* の目的は、共通 RID を検索することです。索引 ANDing 結合は、同じ表内の対応する列に複数の索引を作成し、さらに複数の AND 述部を使用する照会をこの表に

対して実行するアプリケーションにおいて行われます。 その種の照会において、索引付けされた列ごとに複数の索引スキャンを行うと、ビットマップを作成するためにハッシュされた値が生成されます。 2 番目のビットマップは 1 番目のビットマップをプローブするのに使用され、最終的に戻されるデータ・セットを作成するために取り出される修飾行を生成します。

たとえば、以下の 2 つの索引定義があるとします。

```
INDEX IX4: SALARY   ASC
INDEX IX5: COMM     ASC
```

この 2 つの索引を使って、以下の述部が解決されることになります。

```
WHERE SALARY BETWEEN 20000 AND 30000
   AND COMM BETWEEN 1000 AND 3000
```

この例では、索引 IX4 をスキャンすると、 SALARY BETWEEN 20000 AND 30000 述部を満たすビットマップが生成されます。 IX5 のスキャンおよび IX4 のビットマップのプローブを行うと、両方の述部を満たす修飾 RID のリストが生成されます。これは、「動的ビットマップ AND 結合」と呼ばれます。これは、表が十分なカーディナリティーを持っていて、さらに列の修飾範囲内に十分な値がある (または、等号述部が使用される場合は多数の重複がある) 場合にのみ行われます。

複数索引のスキャンを使った動的ビットマップのパフォーマンスを実現するためには、ソート・ヒープ・サイズ (*sortheap*) データベース構成パラメーターと、ソート・ヒープのしきい値 (*sheapthres*) データベース・マネージャー構成パラメーターとの値を変更する必要のある場合があります。

動的ビットマップがアクセス・プランの中で使用されている場合は、追加のソート・ヒープ・スペースが必要になります。 *sheapthres* が比較的、*sortheap* に近い (つまり、並行問い合わせの割合が 2、3 回の係数よりも少ない) 場合、重複索引アクセスに伴う動的ビットマップは、オプティマイザーが想定した値よりずっと少ないメモリーで作動しなくてはなりません。これを解決するには、*sheapthres* の値を *sortheap* と比較して増加させます。

注: オプティマイザーは、単一の表にアクセスするのに、索引 ANDing 操作と索引 OR 操作を組み合わせることはしません。

レンジ・クラスター表での索引アクセス

標準的な表とは異なり、従来の B ツリー索引のような行にキー値をマップする物理索引は、レンジ・クラスター表には不要です。その代わりに、列ドメインが持つ順次性を利用し、表内の行の位置に従い機能的にマッピングをします。このマッピングの単純な例では、範囲の最初のキー値は表の最初の行となり、範囲の 2 番目の値は表の 2 番目の行となります。

オプティマイザーは表のレンジ・クラスター・プロパティーを使用し、完全にクラスター化された索引に基づいてアクセス・プランを生成します。その際のコストはレンジ・クラスター関数の計算だけです。レンジ・クラスター表には元のキー値配列が保持されるので、表内の行のクラスター化は保証されます。

関連概念:

- 277 ページの『索引の利点と欠点』

- 166 ページの『データ・アクセス方式』
- 166 ページの『索引スキャンによるデータ・アクセス』
- 172 ページの『索引アクセスとクラスター率』

索引アクセスとクラスター率

アクセス・プランを選択するとき、オプティマイザーはディスクから必要なページをバッファー・プールに取り出すのに必要な入出力の数を見積もります。この見積もりには、バッファー・プール使用率の予測も含まれています。すでにバッファー・プールの中にあるページに含まれている行を読み取るには、追加の入出力が必要ないためです。

索引スキャンの場合、オプティマイザーは、システム・カタログ表 (SYSCAT.INDEXES) の情報を使用して、データ・ページをバッファー・プール内に読み込むための入出力コストを見積もります。 SYSCAT.INDEXES 表の以下の列の情報を使用します。

- CLUSTERRATIO 情報はこの索引に関連して表データのクラスター化の程度を示します。数が大きいほど、行は索引キーの順序に並んでいます。表の行がほとんど索引キーの順序に並んでいれば、いくつもの行を 1 つのデータ・ページがバッファーにある内にそのページから読み取ることができます。この列の値が -1 の場合、オプティマイザーは、使用可能なら PAGE_FETCH_PAIRS および CLUSTERFACTOR 情報を使用します。
- PAGE_FETCH_PAIRS には CLUSTERFACTOR 情報と共に、データ・ページをさまざまなサイズのバッファー・プールに読み込むのに必要な入出力の数をモデル化するための数値の対がいくつか含まれています。これらの列のデータは、索引に対して RUNSTATS を DETAILED 文節付きで実行した場合だけ集められます。

索引のクラスタリング統計が使用不可である場合、オプティマイザーはデフォルト値を使います。この値は索引に関するデータのクラスタリング率が低いことを想定しています。

索引のデータがどの程度クラスター化されているかによってパフォーマンスに重大な影響を与える可能性があるため、表の索引の 1 つを 100% クラスター化に近く維持してください。

一般的に、キーがクラスター索引のキーのスーパーセットである場合と 2 つの索引のキー列間に事実上の相関がある場合を除いて、 100% クラスタリングできるのは 1 つの索引だけです。

表を再編成するときに、行をクラスター化し挿入処理中この特性を保持するのに使用する索引を指定することができます。更新および挿入を行うと索引に対する表のクラスター化が低下するため、定期的に表を再編成することが必要な場合があります。 INSERT、UPDATE、および DELETE により頻繁に変更が加えられる表に対する再編成の頻度を少なくするには、ALTER TABLE で PCTFREE パラメーターを変更します。こうすると、追加の挿入データがあっても既存のデータとのクラスター化が維持されます。

関連概念:

- 282 ページの『索引のパフォーマンスのヒント』
- 169 ページの『索引アクセスのタイプ』

述部の用語

ユーザー・アプリケーションはデータベースから一連の行を要求するのに、結果セットとして戻す特定の行に対する修飾子を指定した SQL ステートメントを使います。通常この修飾子は照会の WHERE 文節に指定します。こうした修飾子を*述部*と言います。述部は評価プロセスでいつどのように使用されるかによって 4 種類に分類できます。それらのカテゴリーをパフォーマンスの点で高いものから順に示すと、次のようになります。

1. 範囲区切り述部
2. 索引 SARGable 述部
3. データ SARGable 述部
4. Residual の述部

注: *SARGable (SARGable 述部)* とは、各行を Fetch することにより、検索条件に該当するかどうかを確認できる条件節のことです。

次の表は述部カテゴリーの要約です。続くセクションでそれぞれのカテゴリーをもっと詳しく説明します。

表 27. 述部タイプの特性に関するサマリー

特性	述部タイプ			
	範囲区切り	索引 SARGable	データ SARGable	Residual
索引入出力の低減	○	×	×	×
データ・ページ入出力の低減	○	○	×	×
内部的に渡される行数の低減	○	○	○	×
修飾行の数の低減	○	○	○	○

範囲区切り述部および索引 SARGable 述部

範囲区切り述部は索引スキャンの範囲を限定します。それは索引探索の開始および停止キー値を提供します。索引 SARGable 述部は、探索の範囲を限定できませんが、述部が必要とする列は索引キーの一部であるため、索引を利用して評価する事ができます。たとえば、次の索引を考えてみてください。

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

次の WHERE 文節を含む照会についても考えてください。

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
 AND   YEARS > :hv5
```

最初の 2 つの述部 (NAME = :hv1, DEPT = :hv2) は索引 SARGable 述部であり、YEARS > :hv5 は範囲区切り述部です。

オプティマイザーはこれらの述部を評価するにあたり、基本表を読むのではなく、索引データを使用します。これらの*索引 SARGable* 述部によって、表から読み取る必要のある行が少なくなりますが、アクセスする索引ページの数は変わりません。

データ SARGable 述部

索引マネージャーによっては評価できないが、データ・マネージャー・サービスによって評価できる述部は、データ *SARGable* 述部と呼ばれます。通常このタイプの述部は表の個々の行にアクセスすることを必要とします。必要なら、データ・マネージャー・サービスは述部を評価するのに必要な列を取り出し、さらに SELECT リスト内の列を満たすもののうち索引から獲得できなかったものを取り出します。

たとえば、PROJECT 表に定義されている次の単一の索引を考えてみましょう。

```
INDEX IX0:  PROJNO ASC
```

この場合、次の照会では、DEPTNO = 'D11' 述部はデータ SARGable と見なされます。

```
SELECT PROJNO, PROJNAME, RESPEMP
  FROM PROJECT
WHERE DEPTNO = 'D11'
 ORDER BY PROJNO
```

Residual 述部

Residual 述部は表へのアクセス以上の入出力コストを必要とします。以下の特性を持っている場合があります。

- 相関する副照会を使用する
- ANY、ALL、SOME、または IN 文節を含む定量化された副照会を使用する
- 表とは別のファイルに保存されている LONG VARCHAR や LOB のデータを読み取る

こうした述部は、リレーショナル・データ・サービスによって評価されます。

索引のみに適用された述部が、データ・ページがアクセスされる際に再度適用しなければいけなくなる場合があります。たとえば、索引 OR 結合または索引 ANDing 結合を使用するアクセス・プランは、データ・ページにアクセスする際には、常に Residual 述部を再適用します。

関連概念:

- 149 ページの『SQL コンパイラーの処理』

結合方式およびストラテジー

このセクションでは、オプティマイザーが必要に応じて表を結合し、照会に結果を戻す方法について、また、オプティマイザーが使用する方式およびストラテジーについて説明します。

結合

結合とは、情報の何らかの共通の領域に基づいて複数の表からの情報を組み合わせるプロセスのことです。 1 つの表の行は別の表の行と、対応する行が結合基準に合致する場合に、組にされます。

たとえば、次の 2 つの表を考えてみてください。

Table1		Table2	
PROJ	**PROJ_ID**	**PROJ_ID**	**NAME**
A	1	1	Sam
B	2	3	Joe
C	3	4	Mary
D	4	1	Sue
		2	Mike

表の ID 列が同じ値である場合に Table1 と Table2 を結合するには、次に示した SQL ステートメントを使用します。

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABLE1 x, TABLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

この照会で次の結果行のセットが生成されます。

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

結合述部が存在するかどうか、また表と索引の統計によって判別される各種コストに応じて、オプティマイザーは以下の結合方式どれかを選択します。

- ネスト・ループ結合
- マージ結合
- ハッシュ結合

2 つの表を結合する場合、1 つの表は外部表として選択され、もう一方の表は内部表として選択されます。外部表は最初にアクセスされ、1 回だけスキャンされます。内部表が複数回スキャンされるかどうかは、結合の種類および存在する索引によります。照会によって 3 つ以上の表が結合されるとしても、オプティマイザーは 1 回に 2 つの表だけを結合します。必要なら中間結果を保持するために一時表が作成されます。

INNER または LEFT OUTER JOIN のような明示的な結合オペレーターを指定して、結合で表をどう使用するかを決定することができます。しかしながら、この方法で照会を変更する前に、オプティマイザーにどう表を結合するか決定させてみるべきです。それから、照会のパフォーマンスを分析して結合オペレーターを追加するかどうか決めてください。

関連概念:

- 176 ページの『結合方式』
- 183 ページの『パーティション・データベースでの結合ストラテジー』
- 185 ページの『パーティション・データベースでの結合方式』
- 660 ページの『結合の情報』

結合方式

照会が表の結合を要求する場合、オプティマイザーは以下の 3 つの基本結合方式の 1 つを選択することができます。

- ネスト・ループ結合
- マージ結合
- ハッシュ結合

これらの方式を以下のセクションで説明します。

ネスト・ループ結合

ネスト・ループ結合は、以下の 2 つの方法のいずれかで実行されます。

- 外部表のアクセスされる各行ごとに、内部表をスキャンする

  たとえば、表 T1 と T2 の列 A の値が次のようになっている場合を考えてみてください。

外部表 T1: 列 A
2
3
3

外部表 T2: 列 A
3
2
2
3
1

  ネスト・ループ結合を実行するのに、データベース・マネージャーは以下のステップを実行します。

  1. T1 から最初の行を読みます。 A の値は「2」です。
  2. 一致するもの (「2」) が見つかるまで T2 をスキャンしてから、その 2 つの行を結合します。
  3. 次に一致するもの (「2」) が見つかるまで T2 をスキャンしてから、その 2 つの行を結合します。
  4. 表の終わりまで T2 をスキャンします。
  5. T1 に戻って、次の行 (「3」) を読みます。

6. T2 を最初の行から始めて一致するもの (「3」) が見つかるまでスキャンし、その 2 つの行を結合します。
7. 次に一致するもの (「3」) が見つかるまで T2 をスキャンしてから、その 2 つの行を結合します。
8. 表の終わりまで T2 をスキャンします。
9. T1 に戻って、次の行 (「3」) を読みます。
10. 前と同じように T2 をスキャンし、一致する行 (「3」) をすべて結合します。

- 外部表のアクセスされる各行ごとに、内部表で索引検索を行う

  この方法は、次の形式の述部が存在している場合に、指定された述部に使用できます。

  ```
  expr(outer_table.column)  relop  inner_table.column
  ```

  ここで、relop は比較オペレーター (たとえば、=、>、>=、<、または <=) であり、expr は外部表で有効な式です。以下の例を考えてみましょう。

  ```
  OUTER.C1 + OUTER.C2 <= INNER.C1
  OUTER.C4 < INNER.C3
  ```

  この方法を使用すると、結合述部の選択可能性などいくつかの要因に依存するものの、外部表の各アクセスごとに内部表でアクセスされる行の数を大幅に減らせる場合があります。

ネスト・ループ結合を評価するとき、オプティマイザーは、結合を実行する前に外部表をソートするかどうかも決定します。結合列に基づいて外部表を並べ替えるなら、ページがすでにバッファー・プール内に存在する確率が高くなるため、内部表でディスクからページにアクセスするための読み取り操作の数が少なくなる場合があります。結合で高度にクラスター化された索引を使用して内部表にアクセスし、なおかつ外部表がソートされている場合、アクセスされる索引ページの数を最小限にとどめることができます。

さらに、オプティマイザーは、結合後にソートを行うとコストが高くなると予期した場合に、結合前にソートを実行するよう選択することがあります。 GROUP BY、DISTINCT、ORDER BY、またはマージ結合をサポートするには、結合後のソートが必要になる場合があります。

マージ結合

スキャン・マージ結合 またはソート・マージ結合 ということもあるマージ結合には、 table1.column = table2.column という形式の述部が必要です。これは、*等価結合述部* と呼ばれます。マージ結合には、索引アクセスによって、またはソートによって、結合する列での入力の並べ替えが必要になります。結合列が LONG フィールド列やラージ・オブジェクト (LOB) 列である場合には、マージ結合は使用できません。

マージ結合では結合される表は同時にスキャンされます。マージ結合の外部表は 1 回だけスキャンされます。外部表に繰り返される値がなければ、内部表も 1 回だけスキャンされます。繰り返される値がある場合には、内部表の中の行のグループが再度スキャンされることがあります。たとえば、表 T1 と T2 の列 A の値が次の

ようになっている場合、

外部表 T1: 列 A
2
3
3

外部表 T2: 列 A
1
2
2
3
3

マージ結合を実行するのに、データベース・マネージャーは以下のステップを実行します。

1. T1 から最初の行を読みます。 A の値は「2」です。
2. 一致するものが見つかるまで T2 をスキャンしてから、その 2 つの行を結合します。
3. 列が一致する間は T2 のスキャンを続け、列を結合します。
4. T2 内で「3」を読んだら、T1 に戻って次の行を読みます。
5. T1 内の次の値は「3」であり、これは T2 に一致するので、行を結合します。
6. 列が一致する間は T2 のスキャンを続け、列を結合します。
7. T2 の終わりに達します。
8. T1 に戻り、次の行を獲得します。 T1 の次の値は T1 の直前の値と同じなので、 T2 の最初の「3」から再度 T2 のスキャンが開始されます。データベース・マネージャーはこの位置を覚えておきます。

ハッシュ結合

ハッシュ結合には table1.columnX = table2.columnY の形式の述部が 1つ以上必要で、これらの列のタイプは同じでなければなりません。タイプが CHAR の列の場合は、長さが同じでなければなりません。タイプが DECIMAL の列の場合は、精度と位取りが同じでなければなりません。列のタイプを LONG フィールド列やラージ・オブジェクト (LOB) 列にすることはできません。

まず指定された INNER 表をスキャンし、 *sortheap* データベース構成パラメーターで指定されるソート・ヒープから引き出されるメモリー・バッファーに行をコピーします。このメモリー・バッファーは、結合述部の列から計算されたハッシュ値に基づくパーティションに分割されています。 INNER 表のサイズが使用可能なソート・ヒープのスペースより大きい場合は、選択したパーティションから一時表にバッファーが書き込まれます。

内部表が処理されたら、2 番目の表、つまり OUTER 表がスキャンされ、まず結合述部の列で計算されたハッシュ値を比較することにより、 INNER 表からの行と突き合わされます。 OUTER 行の列のハッシュ値が INNER 行の列のハッシュ値と一致したら、実際の結合述部列の値が比較されます。

一時表に書き込まれていないパーティションに対応した OUTER 表の行は、メモリー内の INNER 表の行と直ちに突き合わされます。対応する INNER 表のパーティションが一時表に書き込まれている場合は、 OUTER 行も一時表に書き込まれます。最後に、一致しているパーティションの組みが一時表から読み取られ、それらの行のハッシュ値が突き合わされ、結合述部が検査されます。

ハッシュ結合のパフォーマンス上の効果を十分に得るには、 *sortheap* データベース構成パラメーターの値、および *sheapthres* データベース・マネージャー構成パラメーターの値を変更する必要があるかもしれません。

ハッシュ結合のパフォーマンス上の効果を十分に得るには、 *sortheap* データベース構成パラメーターの値、および *sheapthres* データベース・マネージャー構成パラメーターの値を変更する必要があるかもしれません。

ハッシュ・ループとディスクへのオーバーフローを避けることができれば、ハッシュ結合のパフォーマンスが一番です。ハッシュ結合のパフォーマンスを調整するには、 *sheapthres* に使用可能なメモリーの最大量を見積もり、それから **sortheap** パラメーターを調整してください。可能な限りハッシュ・ループとディスク・オーバーフローを避けられるところまで設定値を大きくしてください。ただし *sheapthres* パラメーターで指定した制限に達しないようにします。

sortheap 値を増やすことも複数ソートのある照会のパフォーマンスを改善します。

関連概念:

- 175 ページの『結合』
- 183 ページの『パーティション・データベースでの結合ストラテジー』
- 185 ページの『パーティション・データベースでの結合方式』
- 660 ページの『結合の情報』

関連資料:

- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 408 ページの『sheapthres - ソート・ヒープしきい値』

最適の結合を選択する方法

照会のために最良の結合方法を選択するのに、オプティマイザーはさまざまな方式を使用します。こうした方式の中に、照会の最適化クラスによって決定される以下の検索方法があります。

- 最長一致の結合列挙
  - スペースおよび時間の観点から有効です。
  - 単一方向列挙です。つまり、2 つの表の結合方法が一度選択されると、それは以降の最適化においても変更されません。
  - 多くの表を結合するときは、最善のアクセス・プランではない場合があります。照会で 2 つまたは 3 つ程度の表しか結合しない場合、最長一致の結合列挙によって選択されるアクセス・プランは、動的プログラミング結合列挙によって選択されるアクセス・プランと同じになります。このことは、照会の同一列に、明示的に指定したものであれ述部変位枠で暗黙的に生成されたものであれ、多数の結合述部がある場合に特に当てはまります。
- 動的プログラミング結合列挙
  - 結合する表の数が増えると、スペースおよび時間の所要量も幾何級数的に増えます。
  - 最適なアクセス・プランのための効率的かつ完全な探索です。
  - DB2® for OS/390 and z/OS が使用する方法と似ています。

結合列挙アルゴリズムは、オプティマイザーが探索するプランの組み合わせの数の重要な決定要素です。

スタースキーマ結合

照会で参照される表はほとんど常に結合述部によって接続されています。結合述部を使わないで 2 つの表が結合すると、 2 つの表のカルテシアン積が形成されます。カルテシアン積では、最初の表の該当する各行が 2 番目の該当する各行に結合され、 2 つの表のサイズの乗積から成る、通常は非常に大きい結果表が作成されます。そのようなプランは効率よく実行するとは考えられないため、オプティマイザーはそうしたアクセス・プランのコストの判別でさえも行いません。

例外は、最適化クラスが 9 に設定されているか、特別な種類のスタースキーマである場合だけです。スタースキーマ には、ファクト表と呼ばれる中心的な表とディメンション表と呼ばれるほかのいくつかの表があります。照会に関係なく、ディメンション表にはすべてファクト表にアタッチする単一の結合だけがあります。それぞれのディメンション表には、ファクト表の特定の列についての情報を展開する追加の値が入っています。一般的な照会はディメンション表の値を参照する複数のローカル述部で成っており、ディメンション表をファクト表に接続する結合述部が含まれています。このような照会では、複数の小さいディメンション表のカルテシアン積を計算してから、大きいファクト表にアクセスするのが役に立ちます。この技法は、複数の結合述部を複数列索引に突き合わせる場合に役に立ちます。

DB2 は、少なくとも 2 つのディメンション表を持つスタースキーマによって設計されたデータベースに対する照会を認識したり、ディメンション表のカルテシアン積の計算を含む潜在的なプランを入れる探索スペースを大きくすることができます。カルテシアン積を計算するプランが見積もりで最低コストである場合は、そのプランがオプティマイザーによって選択されます。

上記で説明したスタースキーマ結合方式は、主キーの索引を結合に使用すると想定しています。それとは別に、外部キー索引に関係するシナリオもあります。ファクト表の外部キー列が単一列索引で、全ディメンション表をまたがって比較的高い選択可能性がある場合には、以下に示すようなスター型結合技法を使用することができます。

1. 各ディメンション表を次の方法で処理します。
   - ディメンション表とファクト表の外部キー索引との間で半結合を実行する方法
   - 行 ID (RID) 値をハッシュして動的にビットマップを作成する方法
2. ビットマップごとに直前のビットマップに対して AND 述部を使用します。
3. 最後のビットマップを処理した後に、残す RID を判別します。
4. それらの RID をソートする (オプション)。
5. 基本表の行を取り出す。
6. SELECT 文節で必要とされるディメンション表の列にアクセスして、ファクト表とそれらの各ディメンション表とを再結合します。
7. Residual 述部を再適用します。

この技法は複数列索引を必要としません。この技法を選択するために、ファクト表とディメンション表の間の明示的な参照保全制約は必要ではありませんが、実際にファクト表とディメンション表の間のリレーションシップはそうした関係である必要があります。

スター結合技法が作成し使用する動的ビットマップはソート・ヒープ・メモリーを必要としますが、そのサイズはソート・ヒープ・サイズ (*sortheap*) データベース構成パラメーターで指定します。

複合表

一対の表を結合した結果は複合表という新しい表になりますが、通常この表は別の内部表との別の結合の外部表になります。これを「複合外部」結合と言います。ある場合、特に最長一致の結合列挙方法を使用している場合には、 2 つの表を結合した結果を後の結合の内部表にすると役に立ちます。ある結合の内部表が 2 つ以上の表を結合した結果で成っていると、そのプランは「複合内部」結合です。たとえば次の照会を考えてみましょう。

```
SELECT COUNT(*)
FROM T1, T2, T3, T4
WHERE T1.A = T2.A AND
      T3.A = T4.A AND
      T2.Z = T3.Z
```

表 T1 と T2 を結合し (T1xT2)、T3 を T4 に結合し (T3xT4)、それから、最初の結合結果を外部表として選択し、 2 番目の結合結果を内部表として選択すると役に立つ場合があります。最後のプランでは ((T1xT2) x (T3xT4))、結合結果 (T3xT4) が複合内部表となります。照会最適化クラスに従って、オプティマイザーは結合の内部表となる表の最大数に異なる制約を課します。複合内部結合は最適化クラス 5、7、および 9 で使用できます。

関連概念:

- 175 ページの『結合』
- 176 ページの『結合方式』
- 183 ページの『パーティション・データベースでの結合ストラテジー』
- 185 ページの『パーティション・データベースでの結合方式』

パーティション・データベースの複製マテリアライズ照会表

複製マテリアライズ照会表は、データベースが表データの事前計算された値を管理できるようにすることによって、パーティション・データベース環境で頻繁に実行される結合のパフォーマンスを改善します。

照会および複製マテリアライズ表の例を考えてみてください。次のような前提事項があります。

- SALES 表は、複数パーティション表スペース REGIONTABLESPACE にあり、REGION 列でパーティション化されています。
- EMPLOYEE 表および DEPARTMENT 表が単一パーティション・データベース・パーティション・グループにあります。

EMPLOYEE 表の情報に基づき、複製マテリアライズ照会表を作成します。

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
      FROM EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

複製マテリアライズ照会表の内容を更新するには、次のステートメントを実行します。

```
REFRESH TABLE R_EMPLOYEE;
```

注: REFRESH ステートメントを使用した後は、他の表と同様に複製表で RUNSTATS を実行する必要があります。

次の例は、従業員ごとの売上、部署の合計、総合計を計算します。

```
SELECT d.mgrno, e.empno, SUM(s.sales)
FROM   department AS d, employee AS e, sales AS s
WHERE  s.sales_person = e.lastname
   AND e.workdept = d.deptno
GROUP BY ROLLUP(d.mgrno, e.empno)
ORDER BY d.mgrno, e.empno;
```

1 つのデータベース・パーティションにしか存在しない EMPLOYEE 表を使用するのではなく、データベース・マネージャーは SALES 表が保管されている各データベース・パーティションで複製される R_EMPLOYEE 表を使用します。結合を計算するために、従業員の情報をネットワークを超えてそれぞれのデータベース・パーティションに移動させる必要はないので、パフォーマンスが向上します。

コロケーテッド結合における複製マテリアライズ照会表

複製マテリアライズ照会表は結合のコロケーションでも助けになります。たとえば、スタースキーマに 20 を超えるノードにまたがる大規模なファクト表がある場合、ファクト表とディメンション表の結合はこれらの表が連結されていると最も効率的です。同一のデータベース・パーティション・グループにすべての表があれば、多い場合でも 1 つのディメンション表がコロケーテッド結合のために正しくパーティション化されます。他のディメンション表はすべてコロケーテッド結合では使用することができません。それは、ファクト表上の結合列がファクト表のパーティション・キーと対応していないためです。

C1 でパーティション化された FACT (C1、C2、C3、...) という表、 C1 でパーティション化された DIM1 (C1、dim1a、dim1b、...) という表、 C2 でパーティション化された DIM2 (C2, dim2a, dim2b, ...) という表などがある場合を考えてみてください。

この場合、述部 DIM1.C1 = FACT.C1 は連結できるので、 FACT と DIM1 の間の結合は完全であることが分かります。これらの表は両方とも C1 列でパーティション化されています。

しかし述部 WHERE DIM2.C2 = FACT.C2 による DIM2 の結合は連結できません。これは FACT が C1 列でパーティション化されており、C2 列ではないからです。この場合、DIM2 をファクト表のデータベース・パーティション・グループに複製して、パーティションごとにローカルに結合するようにできます。

注: この複製マテリアライズ照会表についての説明はデータベース内複製と関連しています。データベース間複製はサブスクリプション、コントロール表、異なったデータベース、および異なったオペレーティング・システムに配置されたデータと関連しています。

複製マテリアライズ照会表を作成する場合、ソース表はデータベース・パーティション・グループの単一ノード表でもマルチノード表でも構いません。ほとんどの場合、複製される表は小さく、単一ノード・データベース・パーティション・グループ内に配置することができます。表からの列のサブセットだけを指定することにより、または述部を使用して行数を指定することにより、さらには両方の方式を使用することにより、複製されるデータを制限することができます。複製マテリアライズ照会表を機能させるにはデータ・キャプチャー・オプションは必要ありません。

ソース表のコピーをすべてのパーティションに作成するため、マルチノード・データベース・パーティション・グループに複製マテリアライズ照会表を作成することもできます。すべてのパーティションに対しソース表をブロードキャストするよりも、大規模なファクト表とディメンション表間の結合は、この環境内でローカルで行う方が良いです。

複製された表の索引は、自動的に作成されません。ソース表のものとは異なる索引を作成することができます。しかし、ソース表になかった制約違反を防ぐため、ユニーク索引の作成や複製された表に制約を加えることはできません。制約はソース表に同じ制約があっても許可されません。

複製された表は照会内で直接参照できますが、特定のパーティション上の表データを見るために複製された表で NODENUMBER() 述部を使用することはできません。

EXPLAIN 機能を使用して、複製マテリアライズ照会表が照会のためのアクセス・プランで使用されたかどうかを調べてください。オプティマイザーが選択するアクセス・プランが複製マテリアライズ照会表を使用するかどうかは、結合される必要のある情報に依存します。オプティマイザーがオリジナル・ソース表をデータベース・パーティション・グループの他のパーティションにブロードキャストするほうがコストがかからないと判断した場合、オプティマイザーが複製マテリアライズ照会表を使用しない場合もあります。

関連概念:

- 175 ページの『結合』

パーティション・データベースでの結合ストラテジー

いくつかの面でパーティション・データベースの結合のための戦略はパーティションのないデータベースと違います。パフォーマンスを改善するために、標準の結合方式に加えて別の技法を適用することができます。

パーティション・データベース内で頻繁な結合が行われる表についての考慮事項の一つに、表の併置があります。表の併置は、パーティション・データベースにおいて、ある表のデータを、同じパーティション・キーに基づいて、同じパーティションにある別の表のデータを使用して見つけ出すための手段を提供します。併置が行

われると、照会の中で結合されるデータは、照会作業の一部として別のデータベース・パーティションに移動せずに処理されます。結合の応答セットのみがコーディネーター・ノードに移動されます。

表キュー

パーティション・データベースでの結合技法の説明は以下の用語を使用します。

- 表キュー

  データベース・パーティション間 (または、単一パーティション・データベースの場合はプロセッサー間) で行を転送するための機構。
- 指示表キュー

  行が受信データベース・パーティションの 1 つにハッシュされる表キュー。
- ブロードキャスト表キュー

  行がすべての受信データベース・パーティションに送信されるが、ハッシュは行われない表キュー。

表キューは、以下の環境で使用されます。

- パーティション間並列処理の使用時に、あるデータベース・パーティションの表データを別のデータベース・パーティションに渡す
- パーティション内並列処理の使用時に、1 つのデータベース・パーティション内で表データを渡す
- 単一パーティション・データベースの使用時に、1 つのデータベース・パーティション内で表データを渡す

各表キューは単一方向にデータを渡します。コンパイラーはどこで表キューが必要とされているかを判断し、それらをプランに組み込みます。プランが実行されると、データベース・パーティション間の接続を行うとその表キューが開始されます。表キューがクローズされるのは、処理が終了したときです。

表キューには、以下に示すようにいくつかの種類があります。

- *非同期表*キュー。 これらの表キューが非同期と呼ばれるのは、アプリケーションによって FETCH が出される前に、行の読み取りを行うためです。 FETCH が出されたときには、行はこの表キューから取り出されます。

  非同期表キューは、SELECT ステートメントに FOR FETCH ONLY 文節を指定した場合に使用されます。行の取り出しだけを行う場合には、非同期表キューが他よりも速い方法になります。
- *同期表*キュー。 これらの表キューが同期と呼ばれるのは、アプリケーションによって FETCH が出されるたびに行を 1 行読み取るためです。各データベース・パーティションでは、カーソルが、そのデータベース・パーティションから次に読み取られる行に位置づけられます。

  同期表キューは、 SELECT ステートメントに FOR FETCH ONLY 文節が指定されていない場合に使用されます。パーティション・データベース環境では、行の更新を行う場合には、データベース・マネージャーは同期表キューを使用します。

- マージ*表*キュー。

  これらの表キューは、順序を保存します。
- *非*マージ*表*キュー。

  これらの表キューは「正規」表キューとも呼ばれます。この表キューは、順序を保存しません。
- *listener 表*キュー。

  これらの表キューは、相関副照会とともに使用されます。相関値が副照会に渡された後、このタイプの表キューを使用して、結果が親照会ブロックに戻されます。

関連概念:

- 175 ページの『結合』
- 176 ページの『結合方式』
- 185 ページの『パーティション・データベースでの結合方式』

パーティション・データベースでの結合方式

以下の図はパーティション・データベースにおける結合方式を示します。

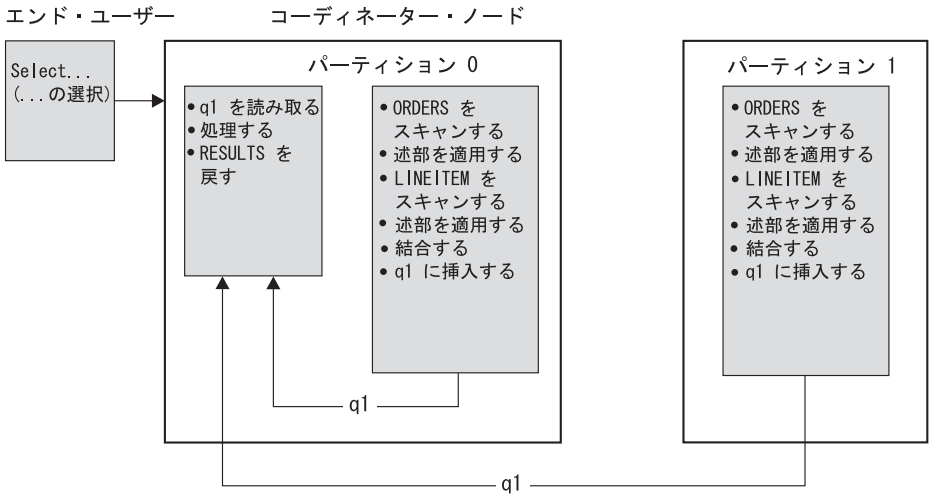
注: 図中の q1、q2、q3 は、例に出てくる表キューと対応しています。図に示されている表は、これらのシナリオの目的に合わせて、 2 つのデータベース・パーティションにまたがって分割されています。矢印は、表キューが送られる方向を示します。なお、コーディネーター・ノードはパーティション 0 です。

コロケーテッド結合

コロケーテッド結合はデータがあるパーティションでローカルに発生します。結合の完成後、そのパーティションはデータを他のパーティションに送信します。オプティマイザーがコロケーテッド結合を処理するためには、結合される表は連結され、対応するパーティション・キーのすべての対が等価結合述部に入れられなければなりません。

次の図に例を示します。

注: 複製マテリアライズ照会表はコロケーテッド結合の可能性を高めます。



LINEITEM 表と ORDERS 表は ORDERKEY 列上でパーティション化される。
結合は、各データベース・パーティションでローカルに行われる。
この例では、結合述部は次のように想定されている。
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

図 13. コロケーテッド結合の例

外部表のブロードキャスト結合

外部表のブロードキャスト結合は、結合される表の間に等価結合述部がない場合に使用できる並列結合方式です。また、この結合方式は、これが最も費用対効果が良い結合方式である状況でも使用されます。たとえば、外部表のブロードキャスト結合は、非常に大きな表が 1 つと非常に小さな表が 1 つあり、どちらの表も結合述部列上でパーティション化されていない場合に使用されます。両方の表をパーティションに分割するよりも、小さな表を大きな表にブロードキャストするほうがコストがかからない可能性があります。次の図に例を示します。

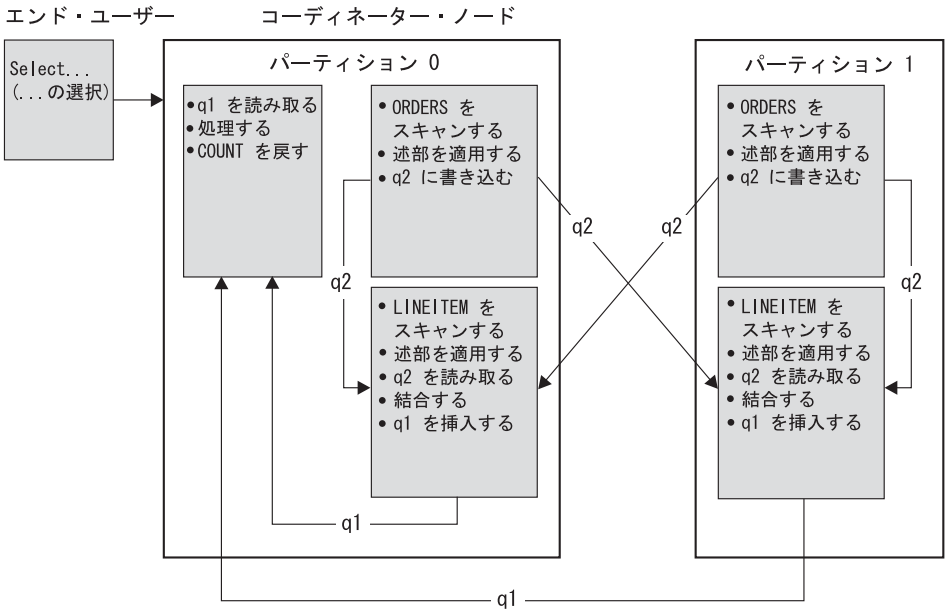
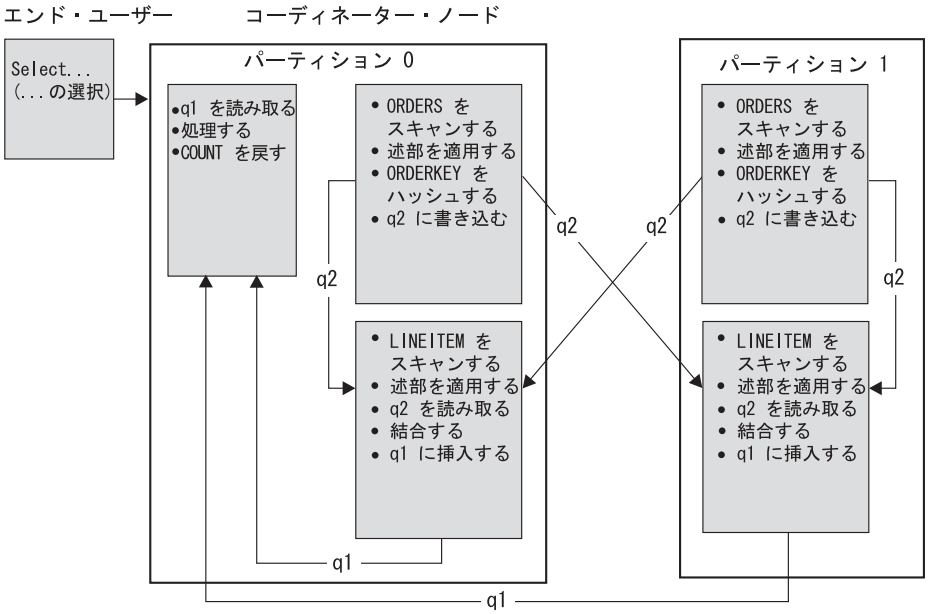


図 14. 外部表のブロードキャスト結合の例

外部表の指示結合

外部表の指示結合方式では、外部表の各行を内部表のパーティション属性に基づいて内部表のパーティションの 1 つに送ります。結合は、このデータベース・パーティション上で行われます。次の図に例を示します。

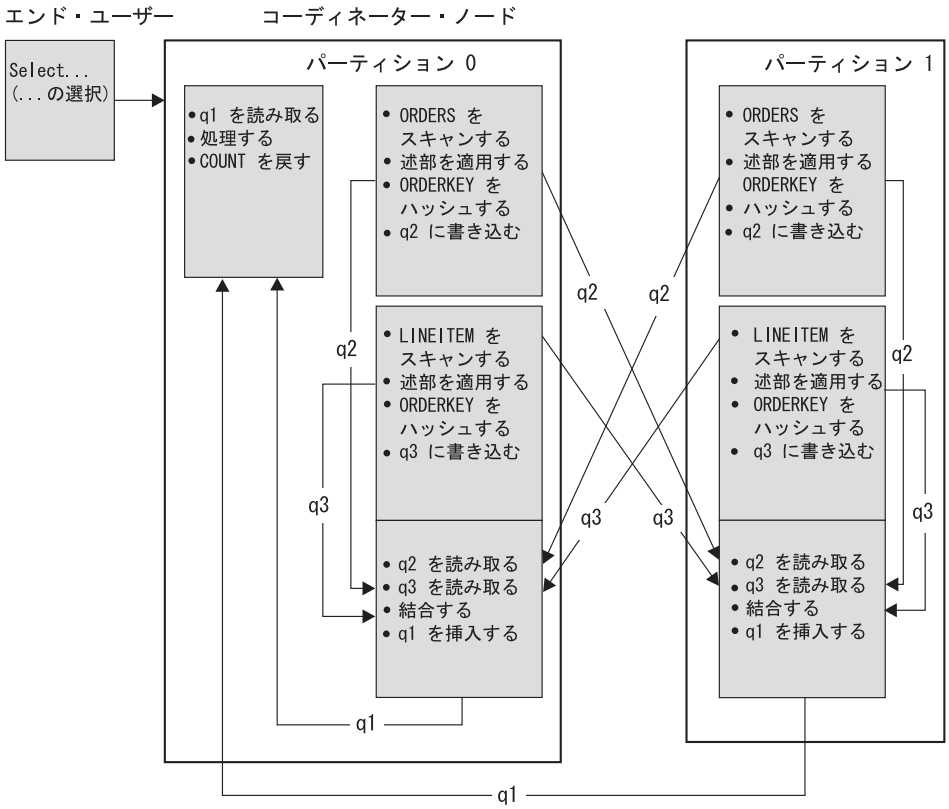


LINEITEM 表は ORDERKEY 列上でパーティション化される。
ORDERS 表は別の列でパーティション化される。
ORDERS 表がハッシュされて、適切な LINEITEM 表のデータベース・パーティションに送られる。
この例では、結合述部は次のように想定されている。
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

図 15. 外部表の指示結合の例

内部表および外部表の指示結合

内部表および外部表の指示結合方式では、結合を行う列の値に基づいて、外部表および内部表の行がデータベース・パーティションのセットに送られます。結合は、これらのデータベース・パーティション上で行われます。次の図に例を示します。例は次の図で示します。

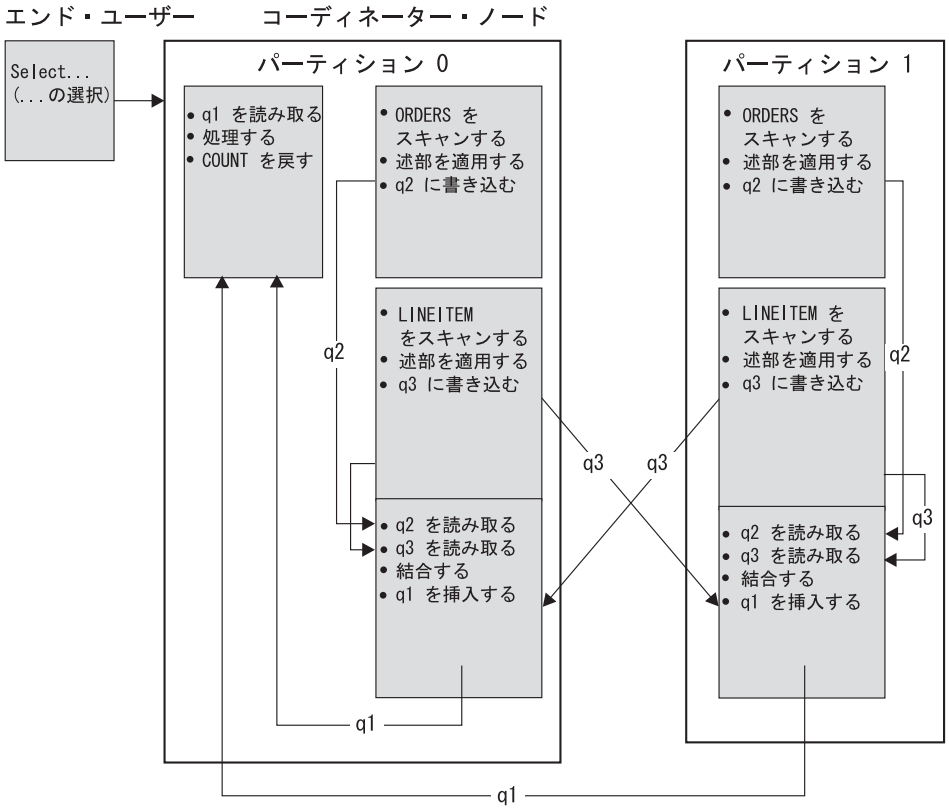


いずれの表も、ORDERKEY 列上ではパーティション化されない。
どちらの表もハッシュされ、新しいデータベースに送られて、そのパーティションで結合される。
両方の表キュー q2 と q3 が送られる。
この例では、結合述部は次のように想定されている。
ORDERS.ORDERKEY = LINEITEM.ORDERKEY

図 16. 内部表および外部表の指示結合の例

内部表のブロードキャスト結合

内部表のブロードキャスト結合方式では、内部表が外部結合表のすべてのデータベース・パーティションに対してブロードキャストされます。次の図に例を示します。

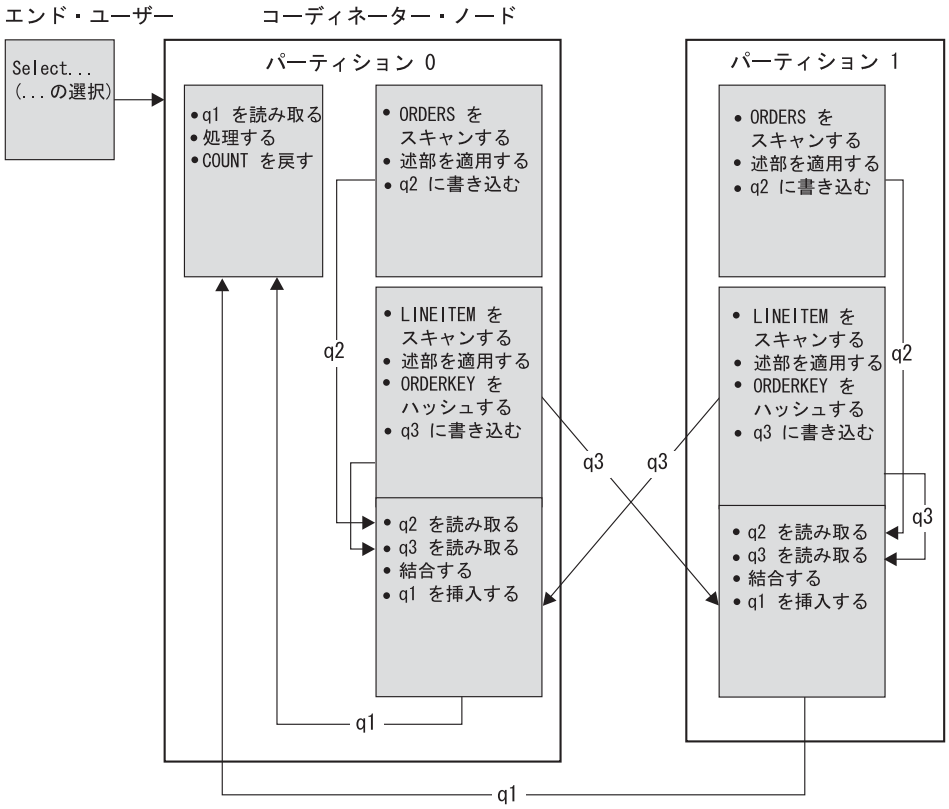


ORDERS 表は、LINEITEM 表を持つデータベース・パーティションすべてに送られる。
表キュー q3 は、外部表のデータベース・パーティションすべてにブロードキャストされる。

図 17. 内部表のブロードキャスト結合の例

内部表の指示結合

内部表の指示結合方式では、内部表の各行を、外部表のパーティション属性に基づいて外部結合表のデータベース・パーティションの 1 つに送ります。結合は、このデータベース・パーティション上で行われます。次の図に例を示します。



ORDERS 表は ORDERKEY 列上でパーティション化される。
LINEITEM 表は別の列でパーティション化される。
LINEITEM 表がハッシュされて、適切な ORDERS 表のデータベース・パーティションに送られる。
この例では、結合述部は次のように想定されている。
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

図 18. 内部表の指示結合の例

関連概念:

- 175 ページの『結合』
- 176 ページの『結合方式』
- 183 ページの『パーティション・データベースでの結合ストラテジー』

ソートとグループ化の影響

オプティマイザーは、アクセス・プランを選択する際に、データのソートによるパフォーマンスの影響を考慮します。ソートは、取り出した行を要求された順序で並び替えることができる索引が存在しない場合に行われます。ソートはオプティマイザーが索引スキャンよりソートの方がコストがかからないと判断した場合にも行われる場合があります。オプティマイザーは以下のいずれかの方法でデータをソートします。

- 照会の実行時に、ソートの結果をパイプ処理します。
- データベース・マネージャー内でソートを内部処理します。

パイプ・ソートと非パイプ・ソート

データの最終的なソートされたリストが 1 回の順次受け渡しで読み取り可能な場合には、結果は*パイプ処理* することができます。パイピングはソート結果を渡すのに非パイプの方法よりも高速に行えます。オプティマイザーは可能ならば、ソート結果をパイプ処理することを選択します。

ソートがパイプ処理されるかどうかには関係なく、ソート時間は、ソートする行の数、キー・サイズ、および行の幅を含め、いくつかの要因によって違ってきます。ソートされる行が、ソート・ヒープ内で使用可能なスペースより多くのスペースを占める場合は、複数のソート・パスが実行され、各パスごとに行全体のうちの 1 つのサブセットがソートされることになります。各ソート・パスはバッファー・プール内の一時表に記憶されます。バッファー・プールに十分のスペースがない場合、この一時表のページはディスクに書き込みます。すべてのソート・パスが完了したなら、それらのソート済みサブセットをマージして、ソート済みの単一の行集合にする必要があります。ソートをパイプ処理する場合、行をマージするときに、直接リレーショナル・データ・サービスに渡されます。

グループ化およびソート・プッシュダウン・オペレーター

場合によっては、オプティマイザーは、リレーショナル・データ・サービス・コンポーネントのデータ管理サービスに対して、ソート操作または集約操作のプッシュダウンを選択することができます。これらの操作をプッシュダウンにすると、データ管理サービス・コンポーネントがデータをソート・ルーチンまたは集約ルーチンに直接渡せるようになり、パフォーマンスが向上します。このプッシュダウンを行わない場合、データ管理サービスはまずこのデータをリレーショナル・データ・サービスに渡し、次いでソートまたは集約ルーチンとインターフェースを取ります。たとえば、次の照会にはこの最適化方法が適しています。

```
SELECT WORKDEPT, AVG(SALARY) AS AVG_DEPT_SALARY
  FROM EMPLOYEE
  GROUP BY WORKDEPT
```

ソートのグループ化操作

GROUP BY 操作で必要な順序をソートが生成する場合に、オプティマイザーは、ソートの実行中に GROUP BY の集約の一部または全部を実行することができます。これは、各グループにある行の数が多い場合は有利です。ソート中に行われる何らかのグループ化により、ソートをディスクにスピルさせる必要がなくなっているか少なくなっている場合はさらに有利です。

ソート中の集約には、正しい結果を戻すためには、集約の以下の最大 3 つのステージが必要です。

1. 最初の集約の段階である部分集約は、ソート・ヒープがいっぱいになるまで集約値を計算します。部分集約において、集約されていないデータが取り込まれて部分的な集約が作成されます。ソート・ヒープがいっぱいになったら、現在のソート・ヒープで計算された部分集約のすべてを含め、残りのデータをディスクに吐き出します。ソート・ヒープがリセットされた後、新しい集約が開始されます。
2. 2 番目の集約の段階である中間集約は、吐き出されたソート実行のすべてを取り込んで、さらにグループ化キーに対して集約を行います。グループ化キー列はパーティション・キー列のサブセットなので、集約はまだ完了しません。中間集約は既存の部分集約を使用して、新しい部分集約を作り出します。このステージは常に行われるわけではありません。これはパーティション内並列処理とパーティ

ション間並列処理の両方で使用されます。パーティション内並列処理では、グローバル・グループ化キーが使用可能になるときにグループ化は終了します。パーティション間並列処理では、グループ化キーが、複数のパーティションにわたってグループを分けているパーティション・キーのサブセットであって、集約を完了するために再パーティション分割が必要な場合に、このことが生じます。集約を完了するために単一のエージェントに減らされる前に、各エージェントが吐き出されたソート実行のマージを終了するときに、似たようなケースがパーティション内並列処理で存在します。

3. 最後の集約の段階である最終集約は、すべての部分集約を取り込んで最終集約を作り出します。このステップは、GROUP BY オペレーターで常に生じます。ソートが分割されないという保証はないので、ソートが集約を完了させることはありません。完全な集約は、非集約データを取り込んで、最終集約を作ります。パーティションがその使用を禁止していない場合、集約のこの方式はすでに正しい順序になっているデータをグループ化するのに通常使用されます。

関連概念:

- 268 ページの『ソート・パフォーマンスのガイドライン』

関連資料:

- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 408 ページの『sheapthres - ソート・ヒープしきい値』

最適化ストラテジー

このセクションでは、パーティション内並列処理およびマルチディメンション・クラスタリング (MDC) 表に対してオプティマイザーが使用する可能性のある、特定のストラテジーについて説明します。

パーティション内並列処理の最適化ストラテジー

SQL ステートメントのコンパイル時に並列処理の多重度が指定された場合、オプティマイザーは、シングル・データベース・パーティション内で並列して照会を実行するアクセス・プランを選択します。

実行時には、サブエージェントと呼ばれる複数のデータベース・エージェントが作成されて、照会を実行します。サブエージェントの数は、SQL ステートメントのコンパイル時に指定された並列処理の多重度以下になります。

オプティマイザーは、アクセス・プランを並列化するため、プランを各サブエージェントによって実行される部分とコーディネーター・エージェントによって実行される部分とに分割します。サブエージェントは、表キューを介して、データをコーディネーター・エージェントか他のサブエージェントに渡します。パーティション・データベースでは、サブエージェントは、表キューを介して、他のデータベース・パーティションのサブエージェントとの間でデータの送受信を行うことができます。

パーティション内の並列スキャン方式

リレーショナル・スキャンおよび索引スキャンは、同じ表または索引上で並列して実行することができます。並列リレーショナル・スキャンの場合、表は、ページ範

囲または行範囲に分割されます。分割後、ページ範囲または行範囲がサブエージェントに割り当てられます。サブエージェントは割り当てられた範囲をスキャンし、その現行の範囲での作業が完了した時点で別の範囲が割り当てられます。

並列索引スキャンの場合には、索引は、索引キー値およびキー値あたりの索引項目数に基づいて、レコード範囲に分割されます。並列索引スキャンは、並列表スキャンと同様に、レコード範囲を割り当てられたサブエージェントを使用して行われます。サブエージェントには、現行の範囲での作業が完了した時点で新しい範囲が割り当てられます。

オプティマイザーは、スキャンの単位 (ページまたは行) と細分度を決定します。

並列スキャンは、サブエージェント間で均等になるように作業を分散します。並列スキャンの目標は、サブエージェント間の負荷を均衡させて、サブエージェントが同等に使用されるようにすることです。使用中のサブエージェントの数が使用可能なプロセッサーの数と等しく、ディスクが入出力要求で過度に作動しているということがない場合には、マシン・リソースは効率的に使用されていると言えます。

他のアクセス・プラン方式によっては、照会の実行時にデータの不均衡が生じることがあります。オプティマイザーは、データのバランスを維持できるように並列方式を選択します。

パーティション内の並列ソート方式

オプティマイザーは、以下のいずれかの並列ソート方式を選択します。

- **ラウンドロビン・ソート**

  このソートは、*再配分ソート* とも呼ばれます。このソート方式では、共用メモリーを効率的に使用し、すべてのサブエージェントに対して可能な限り均一にデータを再配分します。このソートは、ラウンドロビン・アルゴリズムを使用して、均等な分散を行います。まず最初に、各サブエージェントごとに個々のソートを作成します。挿入フェーズでは、サブエージェントが、ラウンドロビン様式で個々のソートにそれぞれデータを挿入していくことによって、より均等なデータの分散を行います。

- **パーティション・ソート**

  このソートは、ソートが各サブエージェントごとに作成されるという点では、ラウンドロビン・ソートに似た働きをします。このソートでは、サブエージェントはハッシュ関数をソート列に適用して、行をどのソートに挿入するかを判別します。たとえば、マージ結合の内部表と外部表がパーティション・ソートの場合、サブエージェントは、マージ結合を使用することによって、対応するパーティションを結合し並列で実行できます。

- **複製ソート**

  このソートは、各サブエージェントがすべてのソート出力を必要とする場合に使用されます。あるソートが作成されると、サブエージェントは、そのソートに行が挿入されるときに同期化されます。ソートが完了すると、各サブエージェントがソート全体の読み取りを行います。このソートは、行数が少ないとき、データ・ストリームのバランスをとり直すのに使用できます。

- **共有ソート**

このソートは、複製ソートと同様の働きをしますが、共有ソートの場合は、ソートされた結果に対してサブエージェントが並列スキャンをオープンし、ラウンドロビン・ソートと同様の方法でサブエージェント間にデータが分配されます。

パーティション内並列一時表

サブエージェントが共同して同じ表に行を挿入することによって、一時表を生成できます。この表は、**共有一時表**と呼ばれます。サブエージェントは、データ・ストリームが複製されるかパーティション化されるかに応じて、専用スキャンまたは並列スキャンのいずれかを共有一時表上でオープンします。

パーティション内の並列集約方式

集約操作は、サブエージェントによって並列に実行することができます。集約操作では、データをグループ化列上に配列する必要があります。サブエージェントがグループ化列の値の集合に関する行をすべて確実に受け取ることができれば、集約を最後まで完全に実行できます。これは、以前のパーティション・ソートのためにグループ化列上のストリームがすでにパーティション化されている場合に生じます。

上記以外の場合は、サブエージェントは部分的に集約を実行し、別の方式を使用して集約を完了させます。その方式は以下のとおりです。

- 表キューをマージして、部分的に集約されたデータをコーディネーター・エージェントに送る。コーディネーターにより集約が完全に行われます。
- 部分的に集合データをパーティション・ソートに挿入します。このソートは、グループ化列上でパーティション化されるため、グループ化列の集合に関するすべての行が確実に 1 つのソート・パーティションに入れられます。
- 処理のバランスをとるためにストリームの複製が必要な場合は、部分的に集合データを複製ソートに挿入できます。個々のサブエージェントは複製ソートを使用して集約を完成させ、集約の結果と同じ内容のコピーを受け取ります。

パーティション内の並列結合方式

結合操作は、サブエージェントによって並列に実行することができます。並列結合の方式は、データ・ストリームの特性によって決められます。

結合は、結合の内部表または外部表でデータ・ストリームをパーティションに分割または複製 (あるいは、その両方) することによって、並列化できます。たとえば、ネスト・ループ結合は、外部のストリームが並列スキャンのためにパーティション化され、さらに内部のストリームが各サブエージェントで別々に再評価されると並列化できます。マージ結合は、内部ストリームと外部ストリームがパーティション・ソートのために値でパーティション化されると並列化できます。

関連概念:

- 99 ページの『アプリケーションの並列処理』
- 195 ページの『MDC 表の最適化ストラテジー』

MDC 表の最適化ストラテジー

マルチディメンション・クラスタリング (MDC) 表を作成した場合、オプティマイザーは追加の最適化ストラテジーを適用できるので、多くの照会のパフォーマンスが

向上する可能性があります。これらのストラテジーは主にブロック索引の効率が改善されたことに基づいていますが、 1 つ以上のディメンションでのクラスタリングによる利点もデータ検索の高速化を可能にしています。

注: MDC 表の最適化ストラテジーは、パーティション内並列処理およびパーティション内並列処理によるパフォーマンス上の利点もインプリメントすることができます。

MDC 表による以下の特定の利点を検討してください。

- ディメンション・ブロック索引の参照数により、表の必要な部分を識別して必要なブロックだけを高速にスキャンすることができます。
- ブロック索引は RID 索引よりも小さいので、参照数はより高速になります。
- 索引の AND 操作および OR 操作をブロック・レベルで実行して、 RID と結合することができます。
- データはエクステント上でクラスター化されることが保証されているので、検索がより高速になります。

sales という名前で、ディメンションが **region** および **month** 列に定義されている MDC 表についての次の簡単な例を検討してください。

```
SELECT * FROM SALES
    WHERE MONTH='March' AND REGION='SE'
```

この照会では、オプティマイザーはディメンション・ブロック索引のルックアップを実行して、 month が March で region が SE のブロックを検索することができます。その後、その結果となる表のブロックだけを高速にスキャンして、結果セットを取り出すことができます。

関連概念:

- 23 ページの『MDC 表のための表および索引管理』

マテリアライズ照会表

マテリアライズ照会表 (MQT) は、複雑な照会、とりわけ以下の操作が必要な照会の応答時間を改善するのに高い効果を発揮します。

- 1 つ以上のディメンションを超えたデータの集約
- 表のグループを超えたデータの結合と集約
- 共通してアクセスされるデータのサブセット (つまり、「ホットな」水平パーティションまたは垂直パーティション) からのデータの照会
- パーティション・データベース環境での表、または表の一部をパーティションに再分割

MQT の情報は、SQL コンパイラーに組み込まれています。 SQL コンパイラーでは、照会書き直しのフェーズとオプティマイザーで照会と MQT の突き合わせを行って、基本表にアクセスする照会の MQT を代用するかどうかを決定します。
MQT を使用する場合は、EXPLAIN 機能を使用して、選択した MQT についての情報を得られます。

MQT は多くの点でレギュラー表のように働くため、表スペース定義を使用したデータ・アクセスの最適化、索引の作成、および RUNSTATS の発行に関する指針は、MQT にも当てはまります。

MQT がいかに役立つかを理解する助けとして、以下の表はマルチディメンション分析照会について示しています。この例では、照会がどれだけ MQT を活用するかが示されています。

この例では、データウェアハウスに一連の顧客と一連のクレジット・カード口座が含まれているデータベース方式を想定しています。ウェアハウスには、クレジット・カードが使用された一連のトランザクションが記録されています。各トランザクションには、一緒に購入された一連の品目が含まれます。ここでは、2 つの大きな表 (1 つはトランザクション品目を含む表、もう 1 つは購入トランザクションを識別する表) が共にスターのハブになっているため、このスキーマは複数スター環境として分類されます。

トランザクションの記述には、製品、場所、そして時刻という 3 つの階層ディメンションがあります。製品階層は、製品グループと製品ラインを表す 2 つの正規化された表に保管されます。場所階層には、市町村、都道府県、および国または地域の情報が含まれ、単一の非正規化された表で表されます。時刻階層には、日、月、および年情報が含まれ、単一データ・フィールドでエンコードされます。日付のディメンションは、組み込み機能を使用して、トランザクションの日付フィールドから抽出されます。このスキーマの他の表は、顧客の口座情報や顧客情報が表されます。

MQT は、以下の階層の各レベルごとに、売上の合計と数を使用して作成されます。
- 製品
- 場所
- 時刻 (年月日)

多くの照会では、この保管された集約データで用が足ります。次の例は、製品グループと製品ラインのディメンション、市町村、都道府県、国のディメンション、および時間のディメンションにしたがって売上の合計と数を計算する MQT の作成方法を示しています。この例では、GROUP BY 文節にいくつか別の列が含まれています。

```
CREATE TABLE dba.PG_SALESSUM
  AS (
      SELECT l.id AS prodline, pg.id AS pgroup,
             loc.country, loc.state, loc.city,
             l.name AS linename, pg.name AS pgname,
             YEAR(pdate) AS year, MONTH(pdate) AS month,
             t.status,
             SUM(ti.amount) AS amount,
             COUNT(*) AS count
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
         AND YEAR(pdate) > 1990
      GROUP BY l.id, pg.id, loc.country, loc.state, loc.city,
               year(pdate), month(pdate), t.status, l.name, pg.name
```

```
    )
DATA INITIALLY DEFERRED REFRESH DEFERRED;

REFRESH TABLE dba.SALESCUBE;
```

このような事前計算済み合計を利用できる照会には、次のものがあります。
- 月と製品グループごとの売上
- 1990 以降の売上の合計
- 1995 または 1996 の売上
- 製品グループまたは製品ラインの売上の合計
- 1995、1996 の特定の製品グループまたは製品ラインの売上の合計
- 特定の国の売上の合計

これらの照会の正確な答えはこの MQT にはありませんが、答えの一部がすでに計算されているので、MQT を使用して答えを計算する方が、大きな基本表を使用するよりもかかる費用がはるかに安くなります。 MQT では、コストのかかる基本データの結合、ソート、および集計を減らすことができます。

次のサンプル照会は、MQT 例にある、すでに計算された結果を使用することによってパフォーマンスを大幅に改善できる例です。

最初の例は、1995 と 1996 の売上の合計を戻します。

```
SET CURRENT REFRESH AGE=ANY

SELECT YEAR(pdate) AS year, SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY year(pdate);
```

2 番目の例は、1995 と 1996 の製品グループごとの売上の合計を戻します。

```
SET CURRENT REFRESH AGE=ANY

SELECT pg.id AS "PRODUCT GROUP",
       SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY pg.id;
```

MQT が大きくなるペースは基本表が大きくなるペースよりも遅いため、基本表が大きくなれば大きくなるほど、応答時間の向上は大きなものになります。 MQT では、MQT が作成されるときと更新されるときに一度計算を行い、その内容を多くの照会で再利用することによって、照会間での作業の重複を効果的に取り除くことができます。

関連概念:

- 227 ページの『設計アドバイザー』
- 181 ページの『パーティション・データベースの複製マテリアライズ照会表』

フェデレーテッド・データベースの照会コンパイラー・フェーズ

このセクションでは、フェデレーテッド・データベース・システムにおける、追加の照会処理フェーズについて説明します。さらに、フェデレーテッド・データベース照会のパフォーマンスを改善するための情報も載せられています。

フェデレーテッド・データベースのプッシュダウン分析

フェデレーテッド・データベースでの照会において、オプティマイザーは、プッシュダウン分析を実行することによって、リモート・データ・ソースで操作を実行できるかどうかを判別します。操作は、関係オペレーター、システムまたはユーザー関数、または SQL オペレーター (GROUP BY、ORDER BY など) の関数とすることができます。

注: DB2® SQL コンパイラーには、データ・ソース SQL サポートについての情報が多数含まれていますが、データ・ソースはアップグレードまたはカスタマイズできるため、そのようなデータをたびたび調整する必要があります。 その場合は、ローカル・カタログ情報を変更することによって、機能の拡張を DB2 に通知してください。カタログを更新するときには、 DB2 DDL ステートメント (CREATE FUNCTION MAPPING や ALTER SERVER など) を使用します。

関数がリモート・データ・ソースにプッシュダウンできない場合、その関数は、照会のパフォーマンスに大きな影響を与える可能性があります。選択述部をデータ・ソースで評価したときではなく、ローカルに評価したときの影響を考慮してください。このような評価では、DB2 にリモート・データ・ソースから表全体を検索させ、それを述部に対してローカルにフィルター操作しなければならない場合があります。またネットワークに制約があったり、表のサイズが大きかったりする場合も、それによってパフォーマンスが影響を受ける可能性があります。

プッシュダウンされないオペレーターも、照会のパフォーマンスに大きな影響を与えることがあります。たとえば、GROUP BY オペレーターでリモート・データ・ソースをローカルに集約するなら、 DB2 にリモート・データ・ソースから表全体を検索させる必要も生じるかもしれません。

例として、ニックネーム N1 が、 DB2 for OS/390 and z/OS のデータ・ソースに含まれるデータ・ソース表 EMPLOYEE を指しているとします。さらに、この表には 10,000 の行があり、列の 1 つには従業員の名前が、そして別の列には給与が入っています。次のようなステートメントを考えてみましょう。

```
SELECT LASTNAME, COUNT(*)  FROM N1
   WHERE LASTNAME  > 'B' AND SALARY > 50000
   GROUP BY LASTNAME;
```

照合シーケンスが DB2 と DB2 for OS/390 and z/OS で同じかどうかによって、いくつかの可能性が考えられます。

- 照合シーケンスが同じ場合、照会の述部が DB2 for OS/390 and z/OS にプッシュダウンできる可能性は高くなります。通常は、DB2 に表全体をコピーしてからロ

ーカルに操作を実行するよりも、データ・ソースで結果をフィルターに掛けてグループ分けする方が効率的です。このような照会では、述部および GROUP BY 操作をデータ・ソースで実行できます。

- 照合シーケンスが異なる場合は、述部の全体をデータ・ソースで評価することはできません。ただし、オプティマイザーは、述部の SALARY > 50000 部分をプッシュダウンする方法をとる場合があります。範囲の比較は、依然として DB2 で実行する必要があります。
- 照合シーケンスが同じで、ローカル DB2 サーバーが非常に高速であることをオプティマイザーが認識している場合は、 GROUP BY 操作を DB2 でローカルに実行するのが最善の (最もコストのかからない) 方法であると判断される可能性があります。述部はデータ・ソースで評価されます。これは、グローバル最適化によって結合されたプッシュダウン分析の一例です。

一般に、目的は、オプティマイザーに確実にデータ・ソース上で関数やオペレーターを評価させることにあります。関数や SQL オペレーターがリモート・データ・ソースで評価されるかどうかは、多くの要素に左右されます。この評価が行われる要因は、次の 3 つのグループに分類できます。

- サーバー特性
- ニックネーム特性
- 照会特性

プッシュダウンの使用に影響を与えるサーバー特性

特定のデータ・ソースに固有の要因により、プッシュダウンが行われるかどうかに影響が生じる場合があります。一般に、このような要因が存在するのは、DB2 で豊富な SQL ダイアレクトがサポートされているためです。このダイアレクトは、照会によってアクセスされるサーバーがサポートしている SQL ダイアレクトよりもさらに多くの機能を持つ場合があります。 DB2 はデータ・サーバーでの機能の不足を補正することができますが、そのようにすると、その操作は DB2 で実行する必要があります。

SQL 機能: 各データ・ソースは、さまざまな SQL ダイアレクト、そして異なるレベルの機能をサポートしています。たとえば、GROUP BY リストを考慮してください。ほとんどのデータ・ソースは GROUP BY オペレーターをサポートしていますが、その中には、GROUP BY リストの項目数が制限されているものもあれば、GROUP BY リストで式が許可されるかどうかに制限があるものもあります。リモート・データ・ソースで制限がある場合、 DB2 は GROUP BY 操作をローカルに実行しなければならないことがあります。

SQL 制約: それぞれのデータ・ソースには、異なる SQL 制約が存在する可能性があります。たとえば一部のデータ・ソースでは、パラメーター・マーカーをリモート SQL ステートメントへの値にバインドする必要があります。したがって、パラメーター・マーカーの制限を調べ、各データ・ソースがそのようなバインド機構をサポートできることを確かめる必要があります。ある関数の値をバインドする適切な方法を DB2 が判別できない場合、この機能はローカルに評価する必要があります。

SQL 制限: DB2 では、リモート・データ・ソースよりも大きい整数を使用できる場合がありますが、リモート制限を超える値をデータ・ソースへの送信ステートメン

トに組み込むことはできません。したがって、この定数を操作する関数やオペレーターは、ローカルに評価される必要があります。

サーバーの特性: いくつかの要因は、このカテゴリーに分類されます。 1 つの例としては、NULL 値が最高値や最低値としてソートされているか、配列に依存するかという要素があります。 NULL 値がデータ・ソースにおいて DB2 とは異なるソートを受ける場合は、 NULL 可能な式での ORDER BY 操作をリモートに評価することはできません。

照合シーケンス: ローカルでソートや比較を行うためにデータを取り出すと、通常はパフォーマンスが低下します。したがって、データ・ソースで使うのと同じ照合シーケンスを使用するように、フェデレーテッド・データベースを構成することを考慮してください。データ・ソースが使用するのと同じ照合シーケンスを使うようにフェデレーテッド・データベースを構成し、 *collating_sequence* サーバー・オプションを「Y」に設定しているなら、オプティマイザーは、結果としてパフォーマンスの改善が得られる場合に多くの照会操作をプッシュダウンすることを考慮に入れることができます。

照合シーケンスが同一である場合は、以下の操作をプッシュダウンできるかもしれません。

- 文字データや数値データの比較
- 文字範囲比較述部
- ソート

ただし、フェデレーテッド・データベースとデータ・ソースとの間で NULL 文字の並び順序が違うと、異常な結果が発生する可能性があります。大文字小文字が区別されるデータ・ソースに比較ステートメントをサブミットすると、ステートメントが予期せぬ結果を戻すことがあります。大文字小文字を区別しないデータ・ソースでは、文字「I」と「i」に割り当てられている並び順序は同じです。デフォルトでは DB2 は大文字小文字を区別し、それぞれの文字に異なる並び順序を割り当てます。

パフォーマンスを向上させるため、フェデレーテッド・サーバーでは、データ・ソースでソートや比較を行うことが可能です。たとえば、DB2 UDB for OS/390 and z/OS では、ORDER BY 文節で定義されたソートは、 EBCDIC コード・ページに基づく照合シーケンスで実行されます。フェデレーテッド・サーバーを使用し、ORDER BY 文節でソートされた DB2 for OS/390 and z/OS データを検索する場合は、 EBCDIC コード・ページに基づいて事前定義された照合シーケンスを使用するようにフェデレーテッド・データベースを構成してください。

フェデレーテッド・データベースとデータ・ソースの照合シーケンスが異なる場合、DB2 はデータをフェデレーテッド・データベースに取り出します。これは、ユーザーが、フェデレーテッド・サーバーに定義されている照合シーケンスで並べられた照会結果を要求しているためで、フェデレーテッド・サーバーはデータをローカルに並べることによってこの要求に応答するからです。データ・ソースの照合シーケンスにある順序でデータを示させる必要がある場合は、照会をパススルー・モードでサブミットするか、データ・ソース・ビューでその照会を定義してください。

サーバー・オプション: いくつかのサーバー・オプションは、プッシュダウンが行われるかどうかに影響を与える場合があります。特に、*collating_sequence*、*varchar_no_trailing_blanks*、および *pushdown* の設定を検討してください。

DB2 タイプ・マッピングおよび関数マッピングの要因: DB2 のデフォルトのローカル・データ・タイプ・マッピングは、データの損失を防ぐため、データ・ソースの各データ・タイプに十分なバッファー・スペースを割り振るよう設計されています。特定のアプリケーションに合わせて、特定のデータ・ソースのタイプ・マッピングをユーザーの手でカスタマイズすることも可能です。たとえば、Oracle データ・ソースの DATE データ・タイプ (デフォルトでは、 DB2 TIMESTAMP データ・タイプにマップされる) にアクセスする場合は、ローカル・データ・タイプを DB2 DATE データ・タイプに変更できます。

次の 3 つのケースでは、データ・ソースでサポートされていない関数を DB2 で補うことができます。

- 関数がリモート・データ・ソースに存在しない。
- 関数は存在するが、オペランドの特性が関数の制限に違反している。この状況の一例として、IS NULL 関係オペレーターをあげることができます。ほとんどのデータ・ソースはこのオペレーターをサポートしていますが、 IS NULL オペレーターの左辺には列名しか使用できないなど、制限が存在する場合もあります。
- 関数は、リモート側で評価されると違う値を戻すことがあります。この状況の一例として、'>' (より大) オペレーターをあげることができます。照合シーケンスの異なるデータ・ソースでは、「より大」オペレーターが DB2 によってローカルに評価されると、異なった結果が返される可能性があります。

プッシュダウンの使用に影響を与えるニックネーム特性

次のニックネーム固有の要因により、プッシュダウンが行われるかどうかに影響が生じる場合があります。

ニックネーム列のローカル・データ・タイプ: 列のローカル・データ・タイプがデータ・ソースでの述部の評価を妨げていないことを確認します。オーバーフローが生じるのを潜在的に防ぐため、デフォルトのデータ・タイプ・マッピングを使用してください。しかし、長さの異なる 2 つの列の間での述部の結合は、DB2 が長い方の列をバインドする方法によっては、短い列が存在するデータ・ソースでは行われません。このような状況が生じると、 DB2 のオプティマイザーが結合シーケンスで評価を行える機会の数に影響することがあります。たとえば、INTEGER または INT データ・タイプを使用して作成された Oracle データ・ソース列は、タイプ NUMBER(38) になります。 DB2 整数の範囲は 2**31 から (-2**31)-1 で、NUMBER(9) とほぼ等しいため、この Oracle データ・タイプのニックネーム列は、ローカル・データ・タイプ FLOAT となります。この場合、DB2 整数列と Oracle 整数列の結合は、 DB2 データ・ソース (短い方の結合列) では行われません。ただし、DB2 INTEGER データ・タイプがこの Oracle 整数列の領域を包含している場合は、 ALTER NICKNAME ステートメントを使用してローカル・データ・タイプを変更することによって、 DB2 データ・ソースで結合を実行できます。

列オプション: ニックネームの列オプションを追加したり変更したりするには、SQL ステートメント ALTER NICKNAME を使用します。

末尾ブランクが含まれていない列の識別には、 *varchar_no_trailing_blanks* オプションを使用します。コンパイラーのプッシュダウン分析ステップでは、列に対して実行するすべての操作を検査するときに、この情報を利用します。この指示に基づき、DB2 は異なってはいても等価な形式の述部を生成し、データ・ソースに送信されるリモート SQL ステートメントで使用できます。データ・ソースに対して異なる述部が評価される可能性はありますが、最終的な結果は同じになります。

この列の値が常に末尾ブランクなしの数値であるかどうかを識別するには、*numeric_string* オプションを使用してください。

次の表は、これらのオプションについて説明しています。

表 28. 列オプションとその設定値

オプション	有効な設定値		デフォルト設定
numeric_string	'Y'	はい - この列には数値データのストリングだけが含まれます。重要: この列に、数値ストリングと末尾ブランクしか含まれない場合は、'Y' は指定しないでください。	'N'
	'N'	いいえ - この列は数値データのストリングに限定されていません。	
	列の numeric_string を 'Y' に設定すると、列データのソートに干渉するブランクがこの列には含まれないことを、オプティマイザーに知らせることになります。このオプションは、データ・ソースの照合シーケンスが DB2 の照合シーケンスとは異なる場合に役立ちます。このオプションでマークされた列は、照合シーケンスが異なるためにローカルな (データ・ソースの) 評価から除かれるということはありません。		
varchar_no_trailing_blanks	このデータ・ソースが、ブランクが埋め込まれていない VARCHAR 比較セマンティクスを使用するかどうかを指定します。後書きブランクを含んでいない可変長文字ストリングについて、一部の DBMS のブランク埋め込みなしの比較セマンティクスでは、 DB2 の比較セマンティクスと同じ結果が戻されます。データ・ソースにあるすべての VARCHAR 表/ビューの列に後書きブランクが含まれていないことが確かである場合は、データ・ソースについてこのサーバー・オプションを「Y」に設定することを考慮してください。このオプションは、Oracle データ・ソースでしばしば使用されます。ビューを含め、ニックネームを持つ可能性のあるすべてのオブジェクトを考慮に入れてください。		'N'
	'Y'	このデータ・ソースのブランク埋め込みなしの比較セマンティクスは、DB2 と同じです。	
	'N'	このデータ・ソースのブランク埋め込みなしの比較セマンティクスは、DB2 と同じではありません。	

プッシュダウンの使用に影響を与える照会特性

照会では、複数のデータ・ソースにあるニックネームを使用する SQL オペレーターを参照できます。セット・オペレーター (たとえば UNION) などの 1 つのオペレーターを使用して、参照された 2 つのデータ・ソースからの結果を組み合わせる場合は、この操作を DB2 で実行する必要があります。このオペレーターは、リモート・データ・ソースで直接に評価することはできません。

関連概念:

- 204 ページの『フェデレーテッド照会を評価する場合の分析のガイドライン』

フェデレーテッド照会を評価する場合の分析のガイドライン

DB2® には、照会が評価される場面を示す 2 つのユーティリティーがあります。

- Visual Explain。このツールは、**db2cc** コマンドで開始します。照会アクセス・プランのグラフを表示するには、このツールを使用してください。各オペレーターの実行位置は、オペレーターの詳細表示に示されます。

  照会をプッシュダウンする場合は、RETURN オペレーターが示されるはずです。この RETURN オペレーターは、標準の DB2 オペレーターです。ニックネームからデータを選択する SELECT ステートメントの場合は、SHIP オペレーターも示されます。 SHIP オペレーターは、フェデレーテッド・データベース操作にユニークなものです。このオペレーターはデータ・フローのサーバー・プロパティーを変更し、リモート・オペレーターとローカル・オペレーターを区別します。この SELECT ステートメントは、データ・ソースによってサポートされている SQL ダイアレクトを使用して生成されます。該当するデータ・ソースのための有効な照会を含めることができます。

  INSERT、DELETE、または UPDATE 照会が完全にリモート・データベースにプッシュダウンできる場合は、アクセス・プランに SHIP ステートメントが示されることはないかもしれません。リモート側で実行されるすべての INSERT、UPDATE、および DELETE ステートメントは、RETURN オペレーターで示されます。ただし、照会の全体をプッシュダウンできない場合は、 SHIP オペレーターに、どの操作がリモート側で実行されるかが示されます。
- SQL Explain。このツールは、**db2expln** または **dynexpln** コマンドで開始します。アクセス・プラン戦略をテキストとして表示するには、このツールを使用してください。

照会がデータ・ソースや DB2 で評価される理由

プッシュダウンの機会を増やす方法を考慮するときには、次のかぎとなる問題を考慮してください。

- この述部はなぜリモートで評価されないのか?

  述部が全く選択的なものであり、これを使用して行をフィルターに掛け、ネットワーク通信量を減らせる場合に、このような疑問が発生します。リモートでの述部評価は、同じデータ・ソースの 2 つの表間での結合をリモートに評価できるかどうかにも影響します。

  調べる必要のある分野は、以下のものがあります。

  – 副照会の述部。この述部には、別のデータ・ソースに関係する副照会が含まれていますか? この述部には、このデータ・ソースでサポートされていない SQL オペレーターに関係する副照会が含まれていますか? すべてのデータ・ソースが、副照会の述部でのセット・オペレーターをサポートしているわけではありません。
  – 述部関数。この述部には、このリモート・データ・ソースで評価できない関数が含まれていますか? 関係オペレーターは、関数として分類されます。

– 述部のバインド要件。この述部をリモートに評価する場合には、特定の値をバインドする必要がありますか? その必要がある場合、このデータ・ソースでの SQL 制限に違反していませんか?
– 最適化のグローバル度。オプティマイザーの側で、ローカル処理の方が費用効率が高いと判断している可能性があります。

- GROUP BY オペレーターがリモートに評価されないのはなぜか?

  いくつかの点を調べることができます。
  – GROUP BY オペレーターへの入力はリモートに評価されますか? 評価されない場合、入力を調べてください。
  – そのデータ・ソースには、このオペレーターについての何らかの制約事項がありますか? たとえば、以下の例があります。
    - GROUP BY 項目の数が限定されている
    - 結合する GROUP BY 項目のバイト・カウントが限定されている
    - GROUP BY リストでは列だけが指定される
  – そのデータ・ソースはこの SQL オペレーターをサポートしていますか?
  – 最適化のグローバル度。オプティマイザーの側で、ローカル処理の方が費用効率が高いと判断している可能性があります。
  – GROUP BY オペレーターの文節には文字式が含まれているか? 含まれている場合は、大文字小文字の区別がリモート・データ・ソースと DB2 で一致しているかどうかを調べてください。
- セット・オペレーターがリモートに評価されないのはなぜか?

  いくつかの点を調べることができます。
  – それぞれのオペランドはどちらも、同じリモート・データ・ソースで完全に評価されていますか? 評価されていない場合で、完全に評価しなければならない場合は、それぞれのオペランドを調べてください。
  – そのデータ・ソースには、このセット・オペレーターについての何らかの制約事項がありますか? たとえば、ラージ・オブジェクトまたは長フィールドは、この特定のセット・オペレーターへの入力として有効ですか?
- ORDER BY 演算がリモートに評価されないのはなぜか?

  以下の点を考慮してください。
  – ORDER BY 演算への入力はリモートに評価されますか? 評価されない場合、入力を調べてください。
  – ORDER BY 文節には文字式が含まれていますか? 含まれている場合は、リモート・データ・ソースと DB2 の照合シーケンスや大文字小文字の区別が同じかどうかを確認してください。
  – そのデータ・ソースには、このオペレーターについての何らかの制約事項がありますか? たとえば、ORDER BY 項目の数が限定されていませんか? データ・ソースは、ORDER BY リストに対しての指定を列に制限していませんか?

関連概念:

- 97 ページの『文字変換のガイドライン』
- 209 ページの『フェデレーテッド・データベース照会のグローバル分析』

- 206 ページの『フェデレーテッド・データベースにおけるリモート SQL 生成とグローバル最適化』
- 668 ページの『フェデレーテッド照会の情報』

フェデレーテッド・データベースにおけるリモート SQL 生成とグローバル最適化

リレーショナル・ニックネームを使用するフェデレーテッド・データベースの照会の場合、アクセス戦略には、元の照会を一連のリモート照会単位に分解してから結果を結合することが関係する場合があります。このようにリモート SQL を生成することは、照会のためのグローバルで最適なアクセス戦略を作成するのに役立ちます。

オプティマイザーは、プッシュダウン分析の出力を使用して、各操作が DB2® でローカルに評価されるのか、データ・ソースでリモートに評価されるのかを決定します。これは、コスト・モデルの出力での決定に基づくものです。コスト・モデルには、操作を評価するときのコストだけではなく、 DB2 とデータ・ソースの間でデータやメッセージを伝送するときのコストも含まれています。

目標は最適化された照会を生成することですが、グローバルな最適化の出力は、大きく分けて次の 2 つの要素の影響を受けます。この影響は、照会のパフォーマンスにも及びます。

- サーバー特性
- ニックネーム特性

グローバルな最適化に影響を与えるサーバー特性とオプション

グローバルな最適化に影響を与える、データ・ソース・サーバーの要素には、次のようなものがあります。

- CPU 速度の相対比率

  データ・ソースの CPU 速度が DB2 の CPU と比較してどの程度速いのか、あるいは遅いのかを指定するには、 *cpu_ratio* サーバー・オプションを使用します。比率が低ければ、データ・ソースのコンピューターの CPU は、DB2 のコンピューターの CPU よりも速いということです。比率が低い場合、DB2 オプティマイザーは、データ・ソースに対してプッシュダウンによる CPU 集中の操作を実行しようとします。

- 入出力速度の相対比率

  データ・ソースのシステム入出力速度が DB2 のシステムと比較してどの程度速いのか、あるいは遅いのかを示すには、 *io_ratio* サーバー・オプションを使用します。比率が低ければ、データ・ソースのワークステーション入出力速度は、DB2 のワークステーション入出力速度よりも速いということです。比率が低い場合、DB2 オプティマイザーは、データ・ソースに対してプッシュダウンによる入出力集中の操作を考慮します。

- DB2 とデータ・ソース間の通信速度

  ネットワーク容量を表示するには、 *comm_rate* サーバー・オプションを使います。比率が低い (DB2 とデータ・ソース間のネットワーク通信が遅いことを示す)

場合、 DB2 オプティマイザーは、このデータ・ソースとの間でやりとりするメッセージ数を減らそうとします。比率を 0 に設定すると、オプティマイザーは、必要なネットワーク通信量が最小になるアクセス・プランを作成します。

- データ・ソースの照合シーケンス

  データ・ソースの照合シーケンスが、ローカル DB2 の照合シーケンスと一致しているかどうかを指定するには、 *collating_sequence* サーバー・オプションを使用します。このオプションを「Y」に設定しないと、オプティマイザーは、このデータ・ソースから検索したデータが整列されていないと見なします。

- リモート・プラン・ヒント

  プラン・ヒントがデータ・ソースで生成または使用されるかどうかを指定するには、 *plan_hints* サーバー・オプションを使用します。デフォルトでは、DB2 はプラン・ヒントを一切データ・ソースに送信しません。

  プラン・ヒントはステートメントの一部分であり、データ・ソース・オプティマイザーについての追加情報を提供します。一部の照会では、この情報がパフォーマンスの向上に役立つ場合があります。プラン・ヒントは、データ・ソース・オプティマイザーが索引を使用するかどうか、どの索引を使用するか、またはどの表結合順序を使うかを判別するのに役立ちます。

  プラン・ヒントが使用可能であれば、データ・ソースに送信される照会には、追加情報が含まれます。たとえば、プラン・ヒントを使用して Oracle オプティマイザーへ送信するステートメントは、次のようになります。

  ```
  SELECT /*+ INDEX (table1, t1index)*/
     col1
     FROM table1
  ```

  プラン・ヒントは、ストリング `/*+ INDEX (table1, t1index)*/` です。

- DB2 オプティマイザー・ナレッジ・ベースでの情報

  DB2 にはオプティマイザー・ナレッジ・ベースがあり、そこには、固有のデータ・ソースについてのデータが含まれています。 DB2 オプティマイザーは、特定の DBMS で生成できないリモート・アクセス・プランを生成しません。つまり DB2 は、リモート・データ・ソースでのオプティマイザーが理解できない、あるいは受け入れられないプランの生成を避けます。

グローバルな最適化に影響を与えるニックネーム特性

次のニックネーム固有の要因が、グローバルな最適化に影響を与える場合があります。

索引の考慮事項: 照会を最適化するには、データ・ソースにある索引に関する情報を DB2 で使用することができます。この理由から、DB2 で使用できる索引情報は、最新のものであることが重要です。ニックネームの索引情報は、ニックネームが作成されるときに最初に取得されます。ビューのニックネームについては、索引情報が収集されることはありません。

ニックネームでの索引の指定の作成: ニックネームのための索引の指定を作成できます。索引の指定は、DB2 オプティマイザーが使用するカタログに索引定義 (実際の索引ではない) を構築します。索引の指定を作成するには CREATE INDEX

SPECIFICATION ONLY ステートメントを使用します。ニックネームの索引の指定を作成する構文は、ローカル表で索引を作成する構文と似ています。

索引の指定の作成は、次のような場合に考慮してください。

- DB2 が、ニックネームの作成時にデータ・ソースから索引情報を取り出せない場合。
- ビューのニックネームのために索引が必要な場合。
- DB2 オプティマイザーで、 NESTED LOOP 結合の内部表として特定のニックネームを使うようにする場合。索引が存在しない場合、ユーザーは結合列上に索引を作成できます。

ビューのニックネームに対して CREATE INDEX ステートメントを発行するときは、まず、その必要があるかどうかを考慮してください。ビューが索引付きの表での単なる SELECT ステートメントである場合は、データ・ソースにある表の索引と一致するニックネームのローカル索引を作成すると、照会のパフォーマンスを大幅に改善できます。しかし、2 つの表を結合させて作成したビューのような、単なる SELECT ステートメントではないビューでローカル索引を作成すると、照会のパフォーマンスは低下する場合があります。たとえば、2 つの表を結合させたビューで索引を作成した場合、オプティマイザーは、そのビューを、NESTED LOOP 結合の内部エレメントとして選択する可能性があります。この結合は複数回評価されるため、照会のパフォーマンスは低下します。別の方法としては、データ・ソースのビューで参照される表ごとにニックネームを作成し、両方のニックネームを参照するローカル・ビューを DB2 で作成することができます。

カタログ統計の考慮事項: システム・カタログ統計には、ニックネーム全体のサイズと、関連する列での値の範囲が示されます。オプティマイザーは、これらの統計を、ニックネームが含まれている照会の処理において最もコストが低いパスを計算するのに使用します。ニックネーム統計は、表統計と同じカタログ・ビューに格納されます。

DB2 では、データ・ソースに保管されている統計データを検索することはできますが、データ・ソースにある既存の統計データに対して行われた更新を自動的に検出することはできません。さらに DB2 では、オブジェクト定義に加えられた変更や、列の追加のような、データ・ソースのオブジェクトに対する構造上の変更は処理できません。オブジェクトの統計データまたは構造データに変更がある場合は、以下の 2 つから処置を選択することができます。

- データ・ソースで RUNSTATS と同等の機能を実行します。次いで、現在のニックネームをドロップして、ニックネームを再作成します。この方法は、構造情報が変更された場合に使用してください。
- SYSSTAT.TABLES ビューの統計を手動で更新します。この方法は、実行するステップは少ないですが、構造情報が変更された場合には機能しません。

関連概念:

- 106 ページの『フェデレーテッド・データベースに影響を与えるサーバー・オプション』
- 204 ページの『フェデレーテッド照会を評価する場合の分析のガイドライン』
- 209 ページの『フェデレーテッド・データベース照会のグローバル分析』

フェデレーテッド・データベース照会のグローバル分析

グローバル・アクセス・プランは、提供されている以下の 2 つのユーティリティーで示されます。

- Visual Explain。コントロール・センターから開始するか、コントロール・センターを始動させる *db2cc* コマンドを実行します。 Visual Explain は、照会アクセス・プランのグラフを表示するのに使用します。各オペレーターの実行位置は、オペレーターの詳細表示に示されます。照会のタイプによって、 SHIP または RETURN オペレーターで各データ・ソースのために生成されたリモート SQL ステートメントを見出すこともできます。各オペレーターの詳細を調べるなら、DB2® オプティマイザーが各オペレーターに関する入出力として見積もる行数が分かります。 さらに、各オペレーターを実行するときの、通信コストを含めた見積もりコストも確認できます。
- SQL Explain。このツールは、*db2expln* または *dynexpln* コマンドで開始します。SQL Explain は、アクセス・プラン戦略をテキストとして表示するのに使用します。 SQL Explain では、コスト情報は示されませんが、リモート Explain 機能でサポートされているデータ・ソース用に、リモート・オプティマイザーで生成されるアクセス・プランを入手できます。

DB2 最適化の決定

パフォーマンスの向上のため、以下の最適化に関する問題とかぎとなる分野について考慮します。

- 同じデータ・ソースの 2 つのニックネームの結合がリモートに評価されないのはなぜか?

  調べる必要のある分野は、以下のものがあります。
  - 結合操作。データ・ソースでは結合操作をサポートしていますか?
  - 結合述部。その結合述部はリモート・データ・ソースで評価されますか? 評価されない場合、その結合述部を調べてください。
  - (Visual Explain を使用した) 結合結果の行数。この結合により作成される行数は、2 つのニックネームを結合するときよりも多いですか? 数値は有効なものですか? 数値が無効である場合、ニックネーム統計を手動で更新することを考慮してください (SYSSTAT.TABLES)。
- GROUP BY オペレーターがリモートに評価されていないのはなぜか?

  調べる必要のある分野は、以下のものがあります。
  - オペレーター構文。そのオペレーターが、リモート・データ・ソースで評価できることを確認してください。
  - 行数。 Visual Explain を使用して、 GROUP BY オペレーターによる入出力の見積行数を調べてください。この 2 つの数値は近いものですか? 近いものである場合、DB2 オプティマイザーは、この GROUP BY をローカルに評価する方が効率的であると見なします。さらに、これら 2 つの数値は有効なものですか? 数値が無効である場合、ニックネーム統計を手動で更新することを考慮してください (SYSSTAT.TABLES)。
- リモート・データ・ソースによってステートメントが完全に評価されないのはなぜか?

DB2 オプティマイザーは、コスト・ベースの最適化を実行します。プッシュダウン分析で、各オペレーターはリモート・データ・ソースで評価できることが示されても、オプティマイザーは、グローバル最適化プランを生成するときには、コストによる見積もりを利用します。そのプランに関与する要因は非常にたくさんあります。たとえば、リモート・データ・ソースが元の照会でそれぞれの操作を処理できても、リモート・データ・ソースの CPU 速度が DB2 の CPU 速度よりはるかに遅いため、 DB2 で操作を実行する方が有利であることが分かる場合があります。結果に満足できない場合、SYSCAT.SERVEROPTIONS のサーバー統計を調べてください。

- オプティマイザーによって生成され、リモート・データ・ソースで完全に評価されるプランのパフォーマンスが、リモート・データ・ソースで直接に実行される元の照会よりもはるかに劣るのはなぜか?

  調べる必要のある分野は、以下のものがあります。

  – DB2 オプティマイザーによって生成されたリモート SQL ステートメント。これが元の照会と等しいことを確認します。述部の順序変更がないか調べます。適切な照会オプティマイザーであれば、照会での述部の順序に影響されることはありません。ただし、すべての DBMS オプティマイザーが同じ動作をするわけではないので、リモート・データ・ソースのオプティマイザーによっては、入力述部の順序に基づいて異なるプランを生成してしまう可能性があります。これは、そのリモート・オプティマイザーにおける固有の問題です。 DB2 への入力時に述部の順序を変更するか、そのリモート・データ・ソースのサービス団体に援助を依頼してください。

    また、述部が置き換えられていないか調べます。適切な照会オプティマイザーであれば、等価な述部の置き換えに影響されることはありません。ただし、すべての DBMS オプティマイザーが同じ動作をするわけではないので、リモート・データ・ソースのオプティマイザーによっては、入力述部に基づいて異なるプランを生成してしまう可能性があります。たとえば、オプティマイザーによっては、述部のトランシティブ・クロージャー・ステートメントを生成できないものもあります。

  – 戻された行の数。この数値は、Visual Explain から入手できます。照会によって多数の行が戻される場合、ネットワーク通信量がボトルネックになっている可能性があります。

  – その他の関数。リモート SQL ステートメントに、元の照会と比較して余分な関数が含まれていませんか? その余分な関数の中には、データ・タイプを変換するために生成されたものが含まれている可能性があります。それの関数が必要であることを確かめてください。

関連概念:

- 106 ページの『フェデレーテッド・データベースに影響を与えるサーバー・オプション』
- 199 ページの『フェデレーテッド・データベースのプッシュダウン分析』
- 204 ページの『フェデレーテッド照会を評価する場合の分析のガイドライン』
- 206 ページの『フェデレーテッド・データベースにおけるリモート SQL 生成とグローバル最適化』
- 668 ページの『フェデレーテッド照会の情報』

- 680 ページの『例 5 : フェデレーテッド・データベースのプラン』

第 7 章 SQL Explain 機能

Explain 機能を使用すると、オプティマイザーによって選択されたアクセス・プランについての情報、および照会を調整するために役立つパフォーマンス情報を取得することができます。

SQL Explain 機能

SQL コンパイラーは、アクセス・プランに関する情報と、静的または動的 SQL ステートメントの環境に関する情報をキャプチャーします。キャプチャーされた情報は、ステートメントおよびデータベース・マネージャー構成を調整して、パフォーマンスを向上させるために個々の SQL ステートメントをどのように実行したらよいかを理解するのに役立ちます。

以下の理由で、 Explain データを収集して使用します。

- 照会を満たすために、データベース・マネージャーがどのように表および索引にアクセスするかを理解する
- パフォーマンス・チューニング・アクションを評価する

  データベース・マネージャー、SQL ステートメント、およびデータベースのいくつかの面を変更する場合、 Explain データを調べて、行ったアクションによってパフォーマンスがどのように変更されたかを調査しなければなりません。

キャプチャーされた情報には、次のものが含まれます。

- 照会を処理する操作の順序
- コスト情報
- 述部および述部ごとの選択可能性の見積もり
- Explain 情報かキャプチャーされた時点の、 SQL ステートメントで参照されている全オブジェクトに関する統計
- SQL ステートメントを再最適化するのに使用する、ホスト変数、パラメーター・マーカー、または特殊レジスターの値。

Explain 情報をキャプチャーするには、その前に、リレーショナル表 (オプティマイザーが Explain 情報を保管する) を作成し、キャプチャーする Explain 情報の種類を判別する特殊レジスターを設定しなければなりません。

Explain 情報を表示するために、コマンド行ツールまたは Visual Explain のいずれかを使用できます。使用するツールによって、収集される Explain データを判別するレジストリー変数をどのように設定するかが決まります。たとえば、 Visual Explain のみを使用する必要がある場合、スナップショット情報だけをキャプチャーする必要があります。 Explain 表に対して、コマンド行ユーティリティーのいずれかを使用するか、またはカスタム SQL ステートメントを使用して、詳細な分析を実行する必要がある場合、すべての Explain 情報をキャプチャーしなければなりません。

関連概念:

- 214 ページの『Explain ツール』
- 216 ページの『Explain 情報の使用のガイドライン』
- 217 ページの『Explain 表および Explain 情報の編成』
- 223 ページの『Explain 情報のキャプチャーのガイドライン』
- 226 ページの『Explain 情報の分析のガイドライン』
- 643 ページの『SQL Explain ツール』

Explain 情報の収集および分析用のツール

Explain 機能は、実行したいパフォーマンス分析の種類により、いくつかの方法で実行することができます。このセクションでは、 Explain 機能をインプリメントするツールについて、および収集した情報の使用方法について説明します。

Explain ツール

DB2® で提供されている包括的な Explain 機能により、 SQL ステートメントでオプティマイザーが選択するアクセス・プランについての、詳細な情報が提供されます。 Explain データが保管される表は、サポートされるすべてのプラットフォームでアクセス可能であり、静的と動的の両方の SQL ステートメントに関する情報が含まれています。いくつかのツールおよび方法が提供されているため、 Explain 情報のキャプチャー、表示、および分析を柔軟に行うことができます。

アクセス・プランを徹底的に分析するために使用できる詳細オプティマイザー情報は、実際のアクセス・プランとは別の Explain 表に保管されます。 Explain 表から情報を入手するための以下の方法のうち、 1 つ以上を使用してください。

- Visual Explain を使用して、Explain スナップショット情報を表示します。

  コントロール・センターから Visual Explain を呼び出して、照会アクセス・プランのグラフィカル表示を参照します。静的と動的の両方の SQL ステートメントを分析できます。

  Visual Explain では、別のプラットフォームで収集または取得したスナップショットを見ることができます。 たとえば、Windows® NT クライアントでは、 DB2 for HP-UX サーバーで生成されたスナップショットをグラフ化することができます。 これが可能なのは、両方のプラットフォームがバージョン 5 レベル以降の場合に限られます。
- *db2exfmt* ツールを使用して、事前に形式設定されている出力の Explain 情報を表示します。
- *db2expln* および *dynexpln* ツールを使用します。

  静的 SQL ステートメントの 1 つまたは複数のパッケージで利用できるアクセス・プラン情報を見るには、コマンド行から *db2expln* ツールを使用します。 *db2expln* は、選択したアクセス・プランを実際に具体化したものを示します。オプティマイザー情報については示しません。

  dynexpln ツールは内部で *db2expln* を使用しますが、パラメーター・マーカーが入っていない動的 SQL ステートメントに対して素早い方法で Explain を実行し

ます。 *dynexpln* 内部からの *db2expln* の使用は、入力 SQL ステートメントを疑似パッケージ内の静的ステートメントに変換することによって行われます。これが実行されても、情報は必ずしも完全に正確であるとは限りません。完全に正確な情報が必要な場合には、 Explain 機能を使用してください。

db2expln ツールは、生成された実際のアクセス・プランを調べることにより、ランタイムにどのような操作が行われるのかに関する比較的コンパクトで英語式の概要を提供します。

- Explain 表に対する独自の照会を作成します。

  独自の照会を作成することにより、簡単な操作で出力したり、別の照会と比較したり、または同じ照会を時間をかけて比較したりすることができます。

注: このコマンド行の Explain ツールや、ほかのツール (*db2batch*、 *dynexpln*、 *db2vexp*、 *db2_all* など) は、 *sqllib* ディレクトリーの *misc* サブディレクトリー内にあります。ツールをこのパスから移動させると、コマンド行の方法はうまくいかなくなります。

次の表では、 DB2 Explain 機能と共に使用できる他のツールとそれらの個々の特性について要約します。 この表を使用して、使用中の環境とニーズに最も適したツールを選択してください。

表 29. Explain 機能ツール

希望する特性	Visual Explain	Explain 表	db2exfmt	db2expln	dynexpln
GUI インターフェース	可				
テキスト出力			可	可	可
「簡易」静的 SQL 分析				可	
サポートされる静的 SQL	可	可	可	可	
サポートされる動的 SQL	可	可	可	可	可*
サポートされる CLI アプリケーション	可	可	可		
DRDA® アプリケーション・リクエスターで使用可能		可			
詳細オプティマイザー情報	可	可	可		
複数ステートメントの分析に適合		可	可	可	可
アプリケーション内部からアクセス可能な情報		可			
注: * db2expln を間接的に使用します。制限がいくつかあります。					

関連概念:

- 651 ページの『dynexpln』
- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

関連資料:

- 683 ページの『付録 D. db2exfmt - Explain 表形式』

- 644 ページの『db2expln - SQL Explain』

Explain 情報の使用のガイドライン

Explain 情報は、以下の 2 つの主な理由で使用します。

- アプリケーションのパフォーマンスが変化した理由を理解する
- パフォーマンス・チューニングの努力を評価する

パフォーマンス変化の分析

照会パフォーマンスが変化した理由を理解するには、前と後の Explain 情報が必要です。これは、以下のステップを実行すると取得できます。

- 変更前の照会の Explain 情報をキャプチャーし、結果の Explain 表を保管します。または、db2exfmt Explain ツールからの出力を保管します。
- この情報を表示するために Visual Explain にアクセスしない場合またはアクセスできない場合は、現行のカタログ統計を保管または印刷します。 db2look 生産性向上ツールを使用して、このタスクの実行に役立てることもできます。
- データ定義言語 (DDL) ステートメント (CREATE TABLE、CREATE VIEW、CREATE INDEX、CREATE TABLESPACE のステートメントを含む) を保管または印刷します。

このようにして収集した情報では、将来の分析での参照点が提供されます。動的 SQL ステートメントの場合は、アプリケーションを最初に実行するときに、この情報を収集することができます。 静的 SQL ステートメントの場合は、バインド時にこの情報を収集することもできます。パフォーマンスの変化を分析するために、収集した情報を、分析を開始したときの照会および環境に関して収集する情報と比較します。

簡単な例として、分析によって、索引がアクセス・パスの一部として使用されていないことが示されたとします。 カタログ統計情報を使用すると、Visual Explain では、索引レベルの数 (NLEVELS 列) が、照会が最初にデータベースにバインドされたときよりもかなり大きくなっていることに気付きます。そこで、以下のアクションのいずれかを実行するように、選択することになります。

- 索引を再編成します。
- 表と索引の新規の統計を収集します。
- 照会を再バインドするときに Explain 情報を収集します。

アクションのいずれかを実行したあとで、アクセス・プランを再度調べます。索引が再度使用されている場合、照会のパフォーマンスはもう問題ではなくなっています。索引がまだ使用されていない場合、またはパフォーマンスにまだ問題がある場合、 2 番目のアクションを実行してその結果を調べます。問題が解決されるまで、これらのステップを繰り返します。

パフォーマンス・チューニングの努力の評価

構成パラメーターの調整、コンテナーの追加、新しいカタログ統計の収集などの、多数のアクションを行うことにより、照会パフォーマンスの向上に寄与することができます。

これらの領域のいずれかで変更を行ってから、SQL Explain 機能を使用して、変更によって選択したアクセス・プランに与える影響があるならば、それを判別することができます。たとえば、索引の指針に基づいて索引またはマテリアライズ照会表 (MQT) を追加した場合、 Explain データを参考にして、実際に索引またはマテリアライズ照会表が期待したとおりに使用されているかどうかを判別できます。

Explain 出力は、選択したアクセス・プランとその相対コストを判別できるようにする情報を提供しますが、照会のパフォーマンスの向上を正確に測定する唯一の方法は、ベンチマーク・テスト技法を使用することです。

関連概念:

- 213 ページの『SQL Explain 機能』
- 217 ページの『Explain 表および Explain 情報の編成』
- 223 ページの『Explain 情報のキャプチャーのガイドライン』
- 227 ページの『設計アドバイザー』
- 196 ページの『マテリアライズ照会表』

関連資料:

- 683 ページの『付録 D. db2exfmt - Explain 表形式』

収集された Explain 情報

このセクションでは、 Explain データを保管するそれぞれの表をリストおよび説明します。また、収集したデータから検索できる情報の種類を説明し、目的のパフォーマンス分析に役立つ情報を取得するための指針を示します。

Explain 表および Explain 情報の編成

Explain インスタンスの概念について、すべての Explain 情報が編成されています。Explain インスタンスは 1 つまたは複数の SQL ステートメントごとに、 1 回の Explain 機能の呼び出しを示します。 1 つの Explain インスタンス内でキャプチャーされた Explain 情報には、 SQL コンパイル環境ならびにコンパイルされる SQL ステートメントを実行するために選ばれたアクセス・プランが入っています。 たとえば、Explain インスタンスは、以下のいずれかから成り立っています。

- 静的 SQL ステートメントでは、 1 つのパッケージに入っているすべての適格な SQL ステートメント。 SELECT、SELECT INTO、UPDATE、INSERT、VALUES、VALUES INTO、および DELETE ステートメントに関する Explain 情報をキャプチャーすることができます。
- 増分バインド SQL ステートメントでは、1 つの特定の SQL ステートメント
- 動的 SQL ステートメントでは、1 つの特定の SQL ステートメント
- 各 EXPLAIN SQL ステートメント (動的か静的のどちらか)

Explain 表の情報は、アクセス・プラン内のオペレーターとデータ・オブジェクトとの間の関連を反映します。次の図は、これらの表の間の関連を示します。

Explain 情報は、以下の表に保管されます。

表 30. Explain データを保管するリレーショナル表

表名	説明
EXPLAIN_ARGUMENT	個々のオペレーターにユニークな特性がある場合、それを示します。
EXPLAIN_INSTANCE	すべての Explain 情報用の主コントロール表。 Explain 表中のデータの各行は、この表内のあるユニークな 1 行に明示的にリンクされます。 Explain 対象の SQL ステートメントのソースに関する基本情報および環境情報は、この表に保持されます。
EXPLAIN_OBJECT	SQL ステートメントを満たすために生成されるアクセス・プランに必要なデータ・オブジェクトを識別します。
EXPLAIN_OPERATOR	SQL コンパイラーが SQL ステートメントを満たすために必要とするすべてのオペレーターが含まれます。
EXPLAIN_PREDICATE	特定のオペレーターによって適用される述部を識別します。
EXPLAIN_STATEMENT	さまざまなレベルの Explain 情報に関する SQL ステートメントのテキストが含まれます。 この表には、ユーザーが入力した元の SQL ステートメントと、アクセス・プランを選択するのにオプティマイザーで使用されるバージョンとが保管されます。 Explain スナップショットが要求されると、 SQL オプティマイザーが選択したアクセス・プランを説明する付加的な Explain 情報が記録されます。この情報は、Visual Explain が求める書式で EXPLAIN_STATEMENT 表の SNAPSHOT 列に保管されます。この書式は他のアプリケーションでは使用できません。
EXPLAIN_STREAM	個々のオペレーターとデータ・オブジェクトの間の入出力データ・ストリームを表します。データ・オブジェクト自体は、EXPLAIN_OBJECT 表に示されています。データ・ストリームに関連するオペレーターは、EXPLAIN_OPERATOR 表にあります。
ADVISE_WORKLOAD	データベースへのワークロードを記述できます。 表の各行はワークロードにおける 1 つの SQL ステートメントであり、関係する頻度によって記述されます。 `db2advis` ツールは、この表を使ってワークロード情報を収集し、保管します。
ADVISE_INSTANCE	`db2advis` の実行に関する情報が入ります。これには開始時刻に関する情報も含まれます。 `db2advis` の実行ごとに 1 行が入ります。
ADVISE_INDEX	推奨索引に関する情報が格納されます。この表のデータは、SQL コンパイラー、 `db2advis` ユーティリティー、またはユーザーによって入力されます。この表は、次の 2 つの目的で使用します。 • 推奨索引を入手します。 • 提案された索引についての入力に基づき索引を評価します。
ADVISE_MQT	CREATE DDL、推奨される各 MQT を定義する照会、 XML 形式での COLSTATS (列情報) などの各 MQT の統計情報、NUMROWS などに加え、各 MQT のサンプリングされた統計を取得するためのサンプリング照会が入ります。
ADVISE_TABLE	推奨される MQT、MDC、およびパーティション化に関する設計アドバイザーの最終的な推奨値を使った、表作成の DDL を保管します。これは指定したオプションと生成された推奨値に応じて異なります。
ADVISE_PARTITION	`db2advis` によって生成および評価される仮想パーティションを保管します。

注: 上記の表すべてがデフォルトに作成されるわけではありません。 これらを作成するには、 *sqllib* サブディレクトリーの *misc* サブディレクトリーにある EXPLAIN.DDL を実行します。

Explain 表は、複数のユーザーに共通にすることができます。ただし、Explain 表は、1 人のユーザーに対して定義して、それぞれの追加ユーザーに対しては、その定義済みの表を指すために同じ名前を使用して、別名を定義することができます。共通の Explain 表を共用する各ユーザーには、それらの表に対する挿入許可が必要です。

関連概念:

- 213 ページの『SQL Explain 機能』
- 219 ページの『データ・オブジェクトの Explain 情報』
- 221 ページの『インスタンスの Explain 情報』
- 220 ページの『データ・オペレーターの Explain 情報』
- 643 ページの『SQL Explain ツール』

データ・オブジェクトの Explain 情報

1 つのアクセス・プランは、 1 つまたは複数のデータ・オブジェクトを使用して SQL ステートメントを実行します。

オブジェクト統計: Explain 機能は、オブジェクトに関して次のような情報を記録します。

- 作成時刻
- オブジェクトの統計が最後に収集された時刻
- オブジェクト内のデータが順番に並べられたかどうかを示す (表または索引オブジェクトのみ)。
- オブジェクトの列数 (表または索引オブジェクトのみ)。
- オブジェクト内の行数の見積もり (表または索引オブジェクトのみ)
- オブジェクトがバッファー・プール内で占有するページ数
- 指定された表スペース (このオブジェクトが保管されている表スペース) にランダム入出力を 1 回行うための、合計見積オーバーヘッド (ミリ秒単位)
- 指定された表スペースから 4K ページを読み取るための、見積転送速度 (ミリ秒単位)
- プリフェッチ・サイズおよびエクステント・サイズ (4K ページ単位)
- 索引を用いたデータ・クラスタリングの程度
- このオブジェクトの索引が使用するリーフ・ページの数、および木のレベル数
- このオブジェクトの索引内の個別全キー値の数
- 表内の合計オーバーフロー・レコード数

関連概念:

- 217 ページの『Explain 表および Explain 情報の編成』
- 221 ページの『インスタンスの Explain 情報』
- 220 ページの『データ・オペレーターの Explain 情報』

- 226 ページの『Explain 情報の分析のガイドライン』

データ・オペレーターの Explain 情報

単一のアクセス・プランでは、 SQL ステートメントを実行し、結果をユーザーに戻すために、データ上でいくつかの操作を実行します。 SQL コンパイラーは必要な操作を判別します。表スキャン、索引スキャン、ネストされたループの結合、またはグループ化オペレーターなどです。

アクセス・プランで使用されるオペレーターを示すことに加えて、 Explain 情報はアクセス・プランの累積効果も示します。

コスト情報の見積もり: オペレーターについては、以下に示した累積コストの見積もりを表示することができます。 これらのコストは選択したアクセス・プランに関するもので、情報がキャプチャーされているオペレーターまでのコストが表示されます。

- 合計コスト (timeron)
- ページ入出力の数
- CPU 命令の数
- 最初の行を取り出すためのコスト (timeron)。必要な初期オーバーヘッドがあるならば、それも含む。
- コミュニケーション・コスト (フレーム単位)。

timeron は、架空の相対メジャー単位です。 timeron は、オプティマイザーによって、内部値、たとえばデータベースの使用に応じて変わる統計などに基づいて決定されます。そのため、 timeron の見積もりコストが決定されるたびに、 SQL ステートメントの timeron メジャーが同じになるという保証はありません。

オペレーターの特性: 以下に示す情報は Explain 機能によって記録されるもので、各オペレーターの特性を記述します。

- アクセスされた表のセット
- アクセスされた列のセット
- データが順番に並べられた列 (オプティマイザーが、この順番付けを後続のオペレーターが使用できると判別する場合)
- 適用された述部のセット
- 戻される行数の見積もり (カーディナリティー)

関連概念:

- 217 ページの『Explain 表および Explain 情報の編成』
- 219 ページの『データ・オブジェクトの Explain 情報』
- 221 ページの『インスタンスの Explain 情報』
- 226 ページの『Explain 情報の分析のガイドライン』

インスタンスの Explain 情報

Explain インスタンス情報は EXPLAIN_INSTANCE 表に保管されます。 Explain インスタンス内の各 SQL ステートメントに関する特定の付加的な情報は、EXPLAIN_STATEMENT 表に保管されます。

Explain インスタンスの識別: 以下の情報を用いると、それぞれの Explain インスタンスを固有に識別し、 SQL ステートメントの情報をこの機能の特定の呼び出しに関連づけることができます。

- Explain 情報を要求したユーザー
- Explain 要求が開始された時刻
- Explain が実行された SQL ステートメントが入っていたパッケージの名前
- Explain が実行された SQL ステートメントが入っていたパッケージのスキーマ
- SQL ステートメントが入っていたパッケージのバージョン
- スナップショット情報が収集されたかどうか

環境設定: SQL コンパイラーが照会をどのように最適化したかに関係するデータベース・マネージャー環境情報が取得されます。環境情報には、以下のものが含まれます。

- DB2® のレベルの、バージョンおよびリリースの番号。
- 照会のコンパイルに使用される並列処理の多重度。

  CURRENT DEGREE 特殊レジスター、DEGREE BIND オプション、SET RUNTIME DEGREE API、および *dft_degree* 構成パラメーターを使用すると、特定の照会のコンパイル時に使用される並列処理の多重度を決めることができます。
- SQL ステートメントが動的と静的のどちらか
- 照会のコンパイルに使用される照会最適化クラス
- 照会のコンパイル時に指定されたカーソルの行ブロッキングのタイプ。
- 照会が実行される分離レベル
- 照会がコンパイルされたときのさまざまな構成パラメーターの値。 Explain スナップショットがとられるときに、以下のパラメーターが記録されます。
  - ソート・ヒープ・サイズ (*sortheap*)
  - アクティブ・アプリケーションの平均数 (*avg_appls*)
  - データベース・ヒープ (*dbheap*)
  - ロック・リスト用最大ストレージ (*locklist*)
  - エスカレーション前のロック・リストの最大パーセント (*maxlocks*)
  - CPU 速度 (*cpuspeed*)
  - 通信スピード (*comm_bandwidth*)

SQL ステートメントの識別: それぞれの Explain インスタンスごとに、複数の SQL ステートメントが Explain されている場合があります。 Explain インスタンスを固有に識別する情報に加えて、次の情報は個々の SQL ステートメントを識別するのに役に立ちます。

- ステートメントのタイプ。SELECT、DELETE、INSERT、UPDATE、定位置 DELETE、定位置 UPDATE
- SYSCAT.STATEMENTS カタログ・ビューに記録された、 SQL ステートメントを発行するパッケージのステートメントおよびセクション番号

EXPLAIN_STATEMENT 表内の QUERYTAG および QUERYNO フィールドには、Explain 処理の一部として設定されている ID が含まれています。 CLP または CLI セッション中にサブミットされた動的 Explain SQL ステートメントの場合、EXPLAIN MODE または EXPLAIN SNAPSHOT がアクティブになっていると、QUERYTAG が「CLP」か「CLI」に設定されます。この場合、各ステートメントごとに 1 かそれ以上ずつ大きくなっている番号のデフォルトが QUERYNO 値になります。その他の動的 Explain SQL ステートメント (CLP、CLI 以外から、または EXPLAIN SQL ステートメントを使用しない) の場合は、 QUERYTAG がブランクに設定され、QUERYNO が常に「1」になります。

コスト見積もり: Explain が実行されたステートメントごとに、選択されたアクセス・プランを実行するのに要する相対コストの見積もりが記録されます。このコストは、*timeron* という架空の相対メジャー単位で示されます。経過時間の見積もりは、次の理由で提供されません。

- SQL オプティマイザーは経過時間ではなく、リソースの消費のみを見積もる。
- オプティマイザーは、経過時間に影響を及ぼす可能性のある因数をすべてモデル化するわけではありません。アクセス・プランの効果性に影響を及ぼさない因数は無視します。実行時の因数の数は経過時間に影響します。これには、次のものが含まれます。システムのワークロード、リソース競合の量、並列処理と入出力の量、行をユーザーに戻すためのコスト、およびクライアントとサーバーの間の通信時間。

ステートメント・テキスト: Explain が実行されたステートメントごとに、 SQL ステートメントのテキストが 2 つのバージョンで記録されます。 1 つのバージョンは、SQL コンパイラーがアプリケーションから受信するコードです。もう 1 つのバージョンは、照会の内部コンパイラー表記からの逆変換です。この変換は他の SQL ステートメントに似ているように見えますが、必ずしも正しい SQL 構文に従っているわけでも、内部表記の実際の内容を全体として反映しているわけでもありません。この変換は、単に、 SQL オプティマイザーがアクセス・プランを選択する元となる SQL コンテキストを理解できるようにするために提供されています。よりよい最適化のために SQL コンパイラーが照会を書き直した方法を理解するには、ユーザー作成のステートメント・テキストを SQL ステートメントの内部表記と比較してください。書き直されたステートメントは、トリガーや制約などのステートメントに影響を及ぼす、環境内の他のエレメントも示します。この「最適化された」テキストが使用するキーワードの一部は、以下のようなものです。

$C*n*	派生列の名前。*n* は整数値を表します。
$CONSTRAINT$	コンパイル中の元の SQL ステートメントに追加された制約の名前を示すタグ。 $WITH_CONTEXT$ 接頭部と組み合わせて表示されます。
$DERIVED.T*n*	派生表の名前。 *n* は整数値を示します。

$INTERNAL_FUNC$ Explain が実行された照会について SQL コンパイラーが使用しても、汎用にはできない機能があることを示すタグ。

$INTERNAL_PRED$ Explain が実行された照会のコンパイル中に、SQL コンパイラーが追加した述部があっても、汎用には使用できない述部があることを示すタグ。内部述部は、トリガーおよび制約のために元の SQL ステートメントに追加された付加的な文脈を満たすために、コンパイラーが使用します。

RID 特定の行の行識別名 (RID) 列を識別するためのタグ。

$TRIGGER$ コンパイル中に元の SQL ステートメントに追加されたトリガーの名前を示すタグ。
$WITH_CONTEXT$ 接頭部と組み合わせて表示されます。

$WITH_CONTEXT$(...) 元の SQL ステートメントに付加的なトリガーまたは制約が追加されると、この接頭部がテキストの最初に表示されます。この接頭部の後に、 SQL ステートメントのコンパイルおよび解決に影響を与えるトリガーまたは制約の名前のリストが表示されます。

関連概念:

- 217 ページの『Explain 表および Explain 情報の編成』
- 219 ページの『データ・オブジェクトの Explain 情報』
- 220 ページの『データ・オペレーターの Explain 情報』
- 226 ページの『Explain 情報の分析のガイドライン』

関連資料:

- 533 ページの『comm_bandwidth - 通信スピード』
- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
- 389 ページの『dbheap - データベース・ヒープ』
- 534 ページの『cpuspeed - CPU 速度』
- 438 ページの『avg_appls - アクティブ・アプリケーションの平均数』
- 503 ページの『dft_degree - デフォルト多重度』

Explain 情報のキャプチャーのガイドライン

Explain データがキャプチャーされるのは、SQL ステートメントがコンパイルされるときに Explain データを要求する場合です。 Explain データを要求するときに、キャプチャーした情報を使用する方法を考慮してください。

注:

1. 増分バインド SQL ステートメントは、ランタイムにコンパイルされると、データはランタイムおよび非バインド・ランタイムに Explain 表に置かれます。これらのステートメントの場合、挿入される Explain 表の修飾子および許可 ID は、パッケージ所有者のものであり、パッケージを実行する使用者のものではありません。
2. Explain 情報がキャプチャーされるのは、 SQL ステートメントがコンパイルされるときだけです。初期コンパイルの後、動的 SQL ステートメントは、環境の変化に伴って再コンパイルが必要になると、あるいは Explain 機能がアクティブになると、再コンパイルされます。同じ PREPARE ステートメントを同じ SQL ステートメントに対して発行すると、このステートメントを準備または実行するたびに、SQL ステートメントがコンパイルされ、 Explain データがキャプチャーされます。
3. BIND オプション REOPT ONCE/ALWAYS を使用してパッケージをバインドすると、ホスト変数、パラメーター・マーカー、または特殊レジスターを含む SQL ステートメントはコンパイルされ、これらの変数が分かっている場合には実際の値を使用してアクセス・パスが作成され、コンパイル時でこうした値の分からない場合にはデフォルトの推定値を使用してアクセス・パスが作成されます。
4. FOR REOPT ONCE 文節を使用すると、指定された SQL ステートメントを、パッケージ・キャッシュ内の同じステートメントと突き合わせようとします。既に再最適化されてキャッシュに入れられた SQL ステートメントの値が、指定された SQL ステートメントの再最適化に使用されます。ユーザーに必要なアクセス権があるなら、 Explain 表には、新たに生成された、再最適化されたアクセス・プランと、この再最適化に使用された値が入ることになります。
5. マルチ・パーティション・システムでステートメントを Explain する場合は、REOPT ONCE を使用して、最初にコンパイルおよび再最適化されたのと同じパーティション上で Explain する必要があります。そうしないと、エラーが戻ります。

Explain 表内の情報のキャプチャー

- **静的または追加バインド SQL ステートメントの場合**:

  BIND または PREP コマンドに EXPLAIN ALL または EXPLAIN YES オプションのどちらかを指定するか、またはソース・プログラムに静的 EXPLAIN SQL ステートメントを含めます。

- **動的 SQL ステートメント**:

  次のいずれかの状況について、Explain 表情報がキャプチャーされます。

  – CURRENT EXPLAIN MODE 特殊レジスターが以下のように設定されます。
    - YES: SQL コンパイラーは、Explain データをキャプチャーし、SQL ステートメントを実行します。
    - EXPLAIN: SQL コンパイラーは Explain データをキャプチャーしますが、SQL ステートメントは実行しません。
    - RECOMMEND INDEXES: SQL コンパイラーは Explain データをキャプチャーし、推奨索引が ADVISE_INDEX 表に入れられますが、 SQL ステートメントは実行されません。

- EVALUATE INDEXES: SQL コンパイラーは、評価のために ADVISE_INDEX 表に置かれた索引を使用します。 EVALUATE INDEXES モードで実行するすべての動的ステートメントについては、それらの仮想索引が使用可能であるとして Explain が実行されます。仮想索引によってステートメントのパフォーマンスが改善される場合、 SQL コンパイラーは次に、その仮想索引を使用することを選択します。パフォーマンスが改善されないのであれば、その索引は無視されます。提案された索引が役立つかどうかを調べるには、 EXPLAIN 結果を検討してください。

- EXPLAIN ALL オプションが BIND または PREP コマンドで設定されています。この設定により、CURRENT EXPLAIN MODE 特殊レジスターの設定値が NO であっても、 SQL コンパイラーはランタイムに動的 SQL の Explain データをキャプチャーします。さらに、SQL ステートメントも実行して、照会の結果を戻します。

Explain スナップショット情報のキャプチャー

Explain スナップショットが要求されると、 Explain スナップショット情報は、Visual Explain が求める書式で EXPLAIN_STATEMENT 表の SNAPSHOT 列に保管されます。この書式は他のアプリケーションでは使用できません。 Explain スナップショット情報の内容に関する付加的な情報は、 Visual Explain 自体から使用できます。この情報には、データ・オブジェクトおよびデータ・オペレーターに関する情報が含まれています。

SQL ステートメントがコンパイルされ、Explain データが要求されていると、以下のように、Explain スナップショット・データがキャプチャーされます。

- **静的または追加バインド SQL ステートメントの場合**:

  EXPLSNAP ALL か EXPLSNAP YES 文節のどちらかが BIND または PREP コマンドに指定されたり、または FOR SNAPSHOT か WITH SNAPSHOT 文節を使用する静的 EXPLAIN SQL ステートメントがソース・プログラムに含まれている場合に、 Explain スナップショットがキャプチャーされます。

- **動的 SQL ステートメント**:

  次のいずれかの場合に、Explain スナップショットがキャプチャーされます。

  – FOR SNAPSHOT または WITH SNAPSHOT 文節を使用する EXPLAIN SQL ステートメントを発行します。 FOR SNAPSHOT 文節では、Explain スナップショット情報のみがキャプチャーされます。 WITH SNAPSHOT 文節では、スナップショット情報に加えて、すべての Explain 情報がキャプチャーされます。

  – CURRENT EXPLAIN SNAPSHOT 特殊レジスターが以下のように設定されます。

    - YES: SQL コンパイラーは、スナップショット Explain データをキャプチャーし、 SQL ステートメントを実行します。

    - EXPLAIN: SQL コンパイラーは、スナップショット Explain データをキャプチャーしますが、 SQL ステートメントは実行しません。

  – BIND または PREP コマンドに EXPLSNAP ALL オプションを指定します。CURRENT EXPLAIN SNAPSHOT 特殊レジスターの設定値が NO であって

も、 SQL コンパイラーはランタイムにスナップショット Explain データをキャプチャーします。さらに、SQL ステートメントも実行します。

関連概念:

- 213 ページの『SQL Explain 機能』
- 216 ページの『Explain 情報の使用のガイドライン』
- 217 ページの『Explain 表および Explain 情報の編成』
- 227 ページの『設計アドバイザー』
- 226 ページの『Explain 情報の分析のガイドライン』
- 643 ページの『SQL Explain ツール』

Explain 情報の分析のガイドライン

Explain 情報は、主に SELECT ステートメントのアクセス・パスの分析のために使用しますが、 Explain データの分析が照会や環境の調整に役立つ多数の方法があります。以下の種類の分析を考慮してください。

- **索引の使用**

  適切な索引は、パフォーマンスをかなり向上させることができます。 Explain 出力を使用すると、一連の特定の照会に役に立つように作成した索引が使用されているかどうかを判別することができます。 Explain 出力では、次の領域での索引の使用法が探せます。

  – 結合述部
  – ローカル述部
  – GROUP BY 文節
  – ORDER BY 文節
  – 選択リスト

  さらに、Explain 機能を使用して、既存の索引の代わりに別の索引を使用できるかどうか、あるいは、全く索引を使用できないのかを評価することができます。 新規索引を作成した後、 RUNSTATS コマンドを使用してその索引の統計を収集したり、照会を再コンパイルします。すでに Explain データを介して、索引スキャンの代わりに表スキャンが使用されていることに気付いているかもしれません。これは、表データのクラスタリングを変更すると、起こる可能性があります。以前に使用していた索引のクラスター率が現在低下している場合は、その索引に応じてそのデータをクラスター化する表を認識し、 RUNSTATS コマンドを使用して索引と表の両方の統計を収集してから、照会を再コンパイルすることができます。表の再編成によりアクセス・プランが向上したかどうかを判別するために、再コンパイルされた照会の Explain 出力を再検査します。

- **アクセス・タイプ**

  Explain 出力を分析して、データに対するアクセスのタイプ (概して、実行しているアプリケーションのタイプには最適ではない) を探すことができます。たとえば、以下のようにします。

  – **オンライン・トランザクション処理 (OLTP) 照会**

OLTP アプリケーションは、述部を区切る範囲を指定した索引スキャンを使用するのに最高の候補です。これは、そのアプリケーションが、キー列に対して等価述部を使用して修飾された数行しか戻さないようになっているためです。OLTP 照会が表スキャンを使用している場合は、Explain データを分析して、索引スキャンが使用されなかった理由を判別することができます。

– **表示専用照会**

「ブラウズ」タイプの照会の探索基準は不明確で、多数の行を修飾しなければならなくなります。ユーザーが通常、出力データの数個の画面を見るだけならば、一部の結果が戻される前に、応答セット全体を計算する必要はないことを確かめるようにすることができます。この場合、ユーザーのゴールはオプティマイザーの基本操作方針、つまりデータの最初の数画面だけではなく、照会全体のリソースの消費を最小化することとは異なっています。

たとえば、マージ・スキャン結合とソート・オペレーターの両方がアクセス・プランで使用されたことを Explain 出力が示す場合は、何行かがアプリケーションに戻される前に、応答セット全体が一時表でマテリアライズされます。この場合、SELECT ステートメントの OPTIMIZE FOR 文節を用いてアクセス・プランを変更することができます。このオプションを指定する場合、オプティマイザーは一時表の応答セット全体を作成しないアクセス・プランを選択してから、最初の行をアプリケーションに戻そうとすることができます。

- **結合方式**

照会が 2 つの表を結合する場合は、使用されている結合のタイプを調べます。複数行が含まれる結合 (意思決定支援照会での結合など) は、通常、マージ結合を使用するよりも速く実行します。 数行しか含まれない結合 (OLTP 照会など) は、一般に、 NESTED LOOP 結合を使用するよりも速く実行します。 しかし、どちらの場合にも、上記の一般的な結合の作業方法を変更することになる酌量すべき状況があります。たとえば、ローカル述部またはローカル索引の使用などがあります。

関連概念:

- 213 ページの『SQL Explain 機能』
- 643 ページの『SQL Explain ツール』
- 651 ページの『db2expln および dynexpln 出力の説明』

設計アドバイザー

DB2® 設計アドバイザーは、ワークロード・パフォーマンスを大幅に向上させるのに役立つツールです。複雑なワークロードの場合、索引、MQT、クラスタリング・ディメンション、またはパーティションを選択することは、非常に困難なことがあります。設計アドバイザーは、ワークロードのパフォーマンスを改善するのに必要なすべてのオブジェクトを識別します。設計アドバイザーは、ワークロードの SQL ステートメントのセットを考慮に入れ、以下の推奨を生成します。

- 新規索引
- 新規マテリアライズ照会表 (MQT)
- マルチディメンション・クラスタリング (MDC) 表への変換

- 表の再パーティション化
- 指定されたワークロードが使用していない索引および MQT の削除 (GUI ツールから)

こうした推奨のすべて、またはその一部を、設計アドバイザーにただちにインプリメントさせることができます。あるいは後でインプリメントするようスケジュールすることもできます。

設計アドバイザーの GUI またはコマンド行ツールのいずれかを使用することにより、設計アドバイザーで以下のタスクを簡単に行えます。

新規データベースの計画またはセットアップ

データベースの設計の際に、以下の目的で設計アドバイザーを使用します。

- パーティション・データベース環境のテスト環境、および索引、MQT、MDC 表のテスト環境に、代替の設計を生成する。
- パーティション・データベース環境では、次の目的で設計アドバイザーを使用できる。
  - データベースにデータをロードする前に、パーティション化ストラテジーを決定する。
  - 単一パーティション DB2 データベースから複数パーティション DB2 データベースへの移行を支援する。
  - 別のデータベース製品から複数パーティション DB2 データベースへの移行を支援する。
- 手動で生成した 索引、MQT、MDC 表、またはパーティション化ストラテジーを評価する。

ワークロード・パフォーマンスのチューニング

データベースをセットアップした後、次の目的で設計アドバイザーを使用できます。

- 特定のステートメントまたはワークロードのパフォーマンスを改善する。
- サンプル・ワークロードのパフォーマンスを基準に、一般的なデータベース・パフォーマンスを向上させる。
- アクティビティー・モニターによって識別される最も頻繁に実行される照会のパフォーマンスを向上させる。
- 新しいキー照会のパフォーマンスを最適化する方法を判別する。
- 共用メモリー・ユーティリティーに関するヘルス・センターからの推奨、またはソートを集中的に行うワークロードのソート・ヒープの問題に関するヘルス・センターからの推奨にこたえる。
- ワークロードで使用していないオブジェクトを検索する。

設計アドバイザーの出力

設計アドバイザー GUI を使用する場合、設計アドバイザーからの推奨を表示、保管、またはインプリメントできます。コマンド行ツールから設計アドバイザーを実行する場合、出力はデフォルトでは標準出力に印刷され、 ADVISE_TABLE および ADVISE_INDEX 表に保管されます。

- ADVISE_TABLE には、MQT、MDC 表、および パーティション化ストラテジーの推奨に関して、USE_TABLE='Y' が入っています。

MQT 推奨は ADVISE_MQT 表にもあります。MDC 推奨は ADVISE_TABLE 表にもあります。パーティション化ストラテジー推奨は ADVISE_PARTITION 表にもあります。これらの表の RUN_ID 値は、設計アドバイザーの毎回の実行ごとに ADVISE_INSTANCE 表に記録される START_TIME 値に対応しています。

- ADVISE_INDEX 表には、索引の推奨に関して USE_INDEX='Y' または 'R' が入ります。

また ADVISE_INSTANCE 表は、設計アドバイザーが実行される度に、 1 行が更新されます。

- START_TIME フィールドおよび END_TIME フィールドは、それぞれユーティリティーの開始時間と停止時間を示します。
- ユーティリティーが正常に終了すると、STATUS フィールドに 'COMPLETED' が入ります。
- MODE フィールドは、-m オプションが使用されたかどうかを示します。
- COMPRESSION フィールドは、圧縮タイプが使用されたかどうかを示します。

-o オプションを使用すると、設計アドバイザーの推奨をファイルに保管できます。保管される設計アドバイザー出力は、以下のエレメントで構成されています。

- 新規索引、MQT、パーティション化ストラテジー、および MDC 表のための CREATE STATEMENTS。
- MQT の REFRESH ステートメント。
- 新規オブジェクトの RUNSTATS コマンド。
- 既存の MQT と索引があって、これをワークロードの実行に使用する場合は、推奨されるスクリプトにそれらの MQT と索引が示されます。

  注: ADVISE_MQT 表の COLSTATS 列には、MQT の列統計が含まれます。統計は、XML 構造です。以下のとおりです。

```
<?xml version=¥"1.0¥" encoding=¥"USASCII¥"?>
<colstats>
    <column>
         <name>COLNAME1</name>
         <colcard>1000</colcard>
         <high2key>999</high2key>
         <low2key>2</low2key>
    </column>
    ....

    <column>
         <name>COLNAME100</name>
         <colcard>55000</colcard>
         <high2key>49999</high2key>
         <low2key>100</low2key>
    </column>
</colstats>
```

  XML 構造には複数の列を含めることができます。各列には、列カーディナリティー (つまり、列内の値の数) が示され、オプションで high2 および low2 キーが示されます。

若干の変更を加えたら、この出力ファイルを CLP スクリプトとして実行して、推奨オブジェクトを作成できます。実行できる変更には、次のものが含まれます。

- すべての RUNSTATS コマンド・ステートメントをまとめて、新規または変更オブジェクトに対する単一の RUNSTATS 呼び出しにする。
- システム生成 ID ではなく、もっと使用しやすいオブジェクト名を指定する。
- すぐにインプリメントしないオブジェクトの DDL を、除去またはコメント化する。

関連概念:

- 「*管理ガイド: プランニング*」の『データのパーティション』
- 277 ページの『索引の利点と欠点』
- 279 ページの『索引の計画のヒント』
- 196 ページの『マテリアライズ照会表』
- 「*管理ガイド: プランニング*」の『マルチディメンション・クラスター化索引』

関連資料:

- 「コマンド・リファレンス」の『設計アドバイザー・コマンド』

設計アドバイザーのワークロードの定義

ワークロード とは、データベース・マネージャーが所定の期間に処理しなければならない一連の SQL ステートメントのことです。たとえば、1 か月の間にデータベース・マネージャーが INSERT を 1000 行、 UPDATE を 10000 行、SELECT を 10000 行、および DELETE を 1000 行処理する必要があるとします。設計アドバイザーは指定されたワークロードを分析し、ワークロード・ステートメントのタイプ、特定のステートメントが使用される頻度、およびデータベースの特性などの要素を考慮して、ワークロードの実行に要するトータル・コストを最小化するための推奨値を生成します。

手順:

設計アドバイザー GUI を使用する方法:

「設計アドバイザー GUI ワークロード (Design Advisor GUI workload)」ページから、新規のワークロード・ファイルを作成するか、または既存のワークロード・ファイルを変更できます。以下の複数のソースからファイルにステートメントをインポートできます。

- 区切り文字で区切られているテキスト・ファイル
- イベント・モニター表
- Query Patroller 履歴データ表 (コマンド行から -qp オプションを使用する)
- EXPLAINED_STATEMENT 表で EXPLAIN されているステートメント
- DB2 スナップショットでキャプチャーした最新の SQL ステートメント

SQL ステートメントをインポートした後で、ステートメントの追加・変更・修正・削除、およびステートメントの頻度の変更を行えます。

設計アドバイザーのコマンド行を使用する方法:

コマンド行から、以下を使用して設計アドバイザーを実行します。

- インラインでコマンドを入力した、単一の SQL ステートメント

- DB2 スナップショットでキャプチャーした動的 SQL ステートメントのセット
- ワークロード・ファイルに組み込んだ SQL ステートメントのセット

動的 SQL ステートメントから設計アドバイザーを実行するには、以下のようにします。

1. 次のコマンドを実行して、データベース・モニターをリセットします。

   ```
   db2 reset monitor for database database-name
   ```

2. データベースに対する動的 SQL ステートメントを実行するまで、しばらく待機します。
3. **db2advis** コマンドに -g オプションを指定して発行します。後で参照できるように、動的 SQL ステートメントを ADVISE_WORKLOAD 表に保管する場合は、 -p オプションも使用します。

ワークロード・ファイルに含まれている SQL ステートメントから設計アドバイザーを実行するには、以下のようにします。

1. 手動でワークロード・ファイルを作成するか (各 SQL ステートメントをセミコロンで区切る)、または前述のソース (複数可) から SQL ステートメントをインポートします。
2. ワークロード内のステートメントの頻度を設定します。ワークロード・ファイル内の各ステートメントには、デフォルトで頻度 1 が割り当てられています。SQL ステートメントの頻度は、そのステートメントがワークロード内に出現する回数を表しますが、この回数は他のステートメントが出現する回数との相対値で示されています。たとえば、ある SELECT ステートメントがワークロード内に 100 回出現し、別の SELECT ステートメントが 10 回出現するとします。これら 2 つのステートメントの相対頻度を表すために、最初の SELECT ステートメントに頻度 10 を割り当て、 2 番目の SELECT ステートメントの頻度を 1 にします。ワークロード内の特定のステートメントの頻度または重みを手動で変更することも可能で、そのステートメントの次の行に - - # SET FREQUENCY *n* を挿入します。 *n* はステートメントに割り当てる頻度値です。
3. **db2advis** コマンドに -i オプションとワークロード・ファイルの名前を指定して実行します。

ADVISE_WORKLOAD 表を含むワークロードから設計アドバイザーを実行するには、 **db2advis** に -w オプションとワークロード名を指定して実行します。

関連概念:

- 227 ページの『設計アドバイザー』

設計アドバイザーを使用した、単一パーティション・データベースから複数パーティション・データベースへの移行

単一パーティション・データベースから複数パーティション・データベースへの移行の際に、設計アドバイザーを役立てることができます。設計アドバイザーは、データのパーティション化についての勧告だけでなく、新規の索引、マテリアライズ照会表 (MQT)、およびマルチディメンション・クラスタリング (MDC) 表についての勧告も提供します。

手順:

1. DB2 UDB ESE の製品ライセンス・キーを更新します。
2. 複数パーティション・データベースのパーティション・グループに少なくとも 1 つの表スペースを作成します。

   注: 設計アドバイザーは既存の表スペースへの再パーティション化のみを推奨する可能性があるため、設計アドバイザーを実行する前に、設計アドバイザーの考慮対象にする表スペースがパーティション化されたデータベース内にあるようにしてください。
3. パーティション化機能を設計アドバイザー GUI で選択するか、または **db2advis** コマンドにパーティション化オプションを指定して、設計アドバイザーを実行します。
4. コントロール・センターで設計アドバイザーを使用している場合は、パーティション化の推奨値を自動的にインプリメントできます。 **db2advis** コマンドを使用している場合は、 **db2advis** 出力ファイルを若干変更してから、設計アドバイザーによって生成された DDL ステートメントを実行します。

関連概念:

- 227 ページの『設計アドバイザー』

関連タスク:

- 「*インストールおよび構成 補足*」の『db2licm コマンドによる DB2 製品ライセンス・キーの登録』

設計アドバイザーの制限と制約事項

1. **索引に関する推奨事項の制限**
   - マテリアライズ照会表 (MQT) 上で索引の使用が推奨されているのは、REFRESH TABLE のパフォーマンスではなくワークロードのパフォーマンスを改善するためです。また、更新、挿入、または削除がワークロードに含まれていない場合、 MQT を変更する (たとえば、更新する) パフォーマンスには、 IMMEDIATE MQT に関するものは含まれません。このため、IMMEDIATE MQT が暗黙指定されたユニーク・キー上で作成されたユニーク索引を MQT に含めることをお勧めします。暗黙指定されたユニーク・キーは、 MQT 照会定義の GROUP BY 文節にある列に基づいたものになります。
   - マルチディメンション・クラスタリングを選択する場合には、クラスタリング RID 索引のみをお勧めします。アドバイザーは、表の MDC 構造を作成するのではなく、オプションとして RID クラスタリング索引をオプションとして組み込みます。
2. **MQT に関する推奨事項の制限**
   - 設計アドバイザーはインクリメンタル MQT を推奨しません。インクリメンタル MQT を作成する場合は、 REFRESH IMMEDIATE MQT を取り、選択したステージング表でインクリメンタルに変換します。
   - MQT 用に推奨される索引は、ワークロードのパフォーマンスを改善するためのものであり、 MQT リフレッシュのパフォーマンスを改善するためのものではありません。

- 指定したワークロードに更新、挿入、または削除が組み込まれていない場合、推奨されている REFRESH IMMEDIATE MQT の更新がパフォーマンスに及ぼす影響は考慮されません。暗黙指定されたユニーク・キー上で作成されたユニーク・キーを REFRESH IMMEDIATE MQT に含めることをお勧めします。暗黙指定されたユニーク・キーは、 MQT 照会定義の GROUP BY 文節にある列で構成されます。

3. **MDC に関する推奨事項の制限**
   - 既存の表にはデータが含まれていなければなりません。データが含まれていない場合、表の MDC は考慮されません。
   - サンプリング・オプション -r がコマンドに指定されていない場合、または MQT サンプリングが GUI ツールで選択されていない場合は、新規 MQT に関する MDC の推奨事項は考慮されません。
   - 設計アドバイザーは、タイプ表または一時表に対しては MDC に関する推奨を行いません。
   - 設計アドバイザーは、フェデレーテッド表に対しては MDC に関する推奨を行いません。
   - 設計アドバイザーの実行中に使用するサンプリング・データ用にストレージ・スペースが必要です。このスペースがない場合は、サンプリング済みの表は基本列に対してのみ、無相関である前提事項に基づいて検査されます。このケースでは、警告メッセージが生成されます。
   - 収集された統計のない表は、スキップして考慮対象から外されます。
   - 設計アドバイザーは複数列のディメンションに対しては推奨を行いません。
   - MDC 選択でサンプリングが実行されるようにするため、既存の表にはデータが含まれていなければなりません。
4. **パーティション化に関する推奨事項の制限**

   設計アドバイザーは、DB2® Enterprise Server Edition でのパーティション化のみを推奨します。 **db2advis** コマンドにパーティション化オプションを指定すると、エラーが戻されます。設計アドバイザー GUI では、単一パーティション・データベース環境でパーティション化機能を選択することはできません。
5. **追加の制限**

   シミュレーション・カタログ表は、設計アドバイザーの実行中に作成されます。設計アドバイザーの実行が完了すると、この表はドロップされます。設計アドバイザーの実行が不完全であると、一部のシミュレーション・カタログ表がドロップされない場合があります。このとき、コマンド行からユーティリティーを再始動することによって、設計アドバイザーを使用してシミュレーション・カタログ表をドロップできます。シミュレーション・カタログ表を除去するには、 -f オプションと -n オプションの両方を指定します (-n オプションには、不完全な実行で使用されたのと同じユーザー名を指定します)。 -f オプションを指定しない場合は、設計アドバイザーは表の除去に必要な DROP ステートメントを生成します。

   これらシミュレーション済みのカタログ表を保管するために別個の表スペースを作成し、 DROP TABLE RECOVERY を ″OFF″ に設定してください。こうすることにより、クリーンアップは容易になり、設計アドバイザーの実行はより高速になります。

関連概念:

- 227 ページの『設計アドバイザー』

第 3 部 システムのチューニングと構成

第 8 章 操作上のパフォーマンス

この章では、実行時の SQL 照会のパフォーマンスに影響を与える要因について説明します。また、特に、パーティション分割やマルチディメンション・クラスタリング (MDC) 表、および同様の機能の利点については、物理データベースの設計上のプラン情報を参照することができます。

メモリー使用

このセクションでは、データベース・マネージャーがどのようにメモリーを使用するかを説明し、データベース・マネージャーおよびデータベースのメモリー使用を制御するパラメーターをリストしています。

メモリー使用の編成

DB2® がどのようにメモリーを編成するかを理解しておくと、メモリーの使用を調整してパフォーマンスを向上させるのに役立ちます。メモリーの使用に影響を与える構成パラメーターは、数多くあります。このパラメーターの中には、サーバーのメモリーを制御するパラメーター、クライアントのメモリーを制御するパラメーター、およびその両方を制御するパラメーターがあります。また、メモリーの割り振りまたは割り振り解除を実行するタイミングやシステム領域もそれぞれ異なります。データベース・サーバーが稼働しているときは、データベース共用メモリー内のメモリー領域のサイズを増加または削減できます。

システム管理者は、システムでのメモリーの使用法に関して、全体のバランスを考慮する必要があります。メモリーの使用法は、オペレーティング・システム上で実行されるアプリケーションの種類によってそれぞれ異なる可能性があります。たとえば、一部のアプリケーションはオペレーティング・システムのキャッシュを使用しますが、データベース・マネージャーは、オペレーティング・システムのキャッシュではなく、独自のバッファー・プールを使用してデータのキャッシュを行います。

次の図は、データベース・マネージャーによって各種用途に割り振られる、メモリーの様々な部分を示しています。

注: この図で示されている使用法は、複数の論理ノードで構成される Enterprise Server Edition 環境でのメモリーの使用法とは異なります。このような環境では、各ノードにデータベース・マネージャー共用メモリーのセットが含まれています。

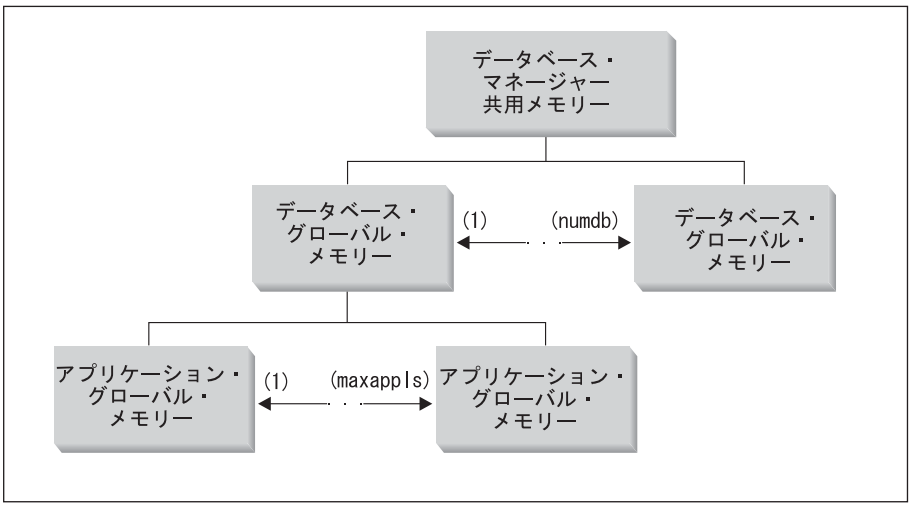


図 19. データベース・マネージャーによって使用されるメモリーのタイプ

メモリーは、次のイベントが発生したときに、データベース・マネージャーの各インスタンスに割り振られます。

- **データベース・マネージャーが開始されたとき (db2start):** データベース・マネージャーのグローバル共用メモリーが割り振られ、データベース・マネージャーが停止される (db2stop) まで割り振られたままになります。この領域には、すべてのデータベース接続でのアクティビティーを管理するためにデータベース・マネージャーが使用する情報が含まれています。最初のアプリケーションがデータベースに接続されると、グローバル・メモリー領域と専用メモリー領域の両方が割り振られます。
- **データベースが最初に活動化または接続されるとき:** データベース・グローバル・メモリーが割り振られます。データベース・グローバル・メモリーは、このデータベースに接続するすべてのアプリケーションで使用できます。データベース・グローバル・メモリーのサイズは、 *database_memory* 構成パラメーターで指定されます。メモリーのサイズは、最初の時点で、必要とされるよりも大きく指定しておけば、後から動的に追加のメモリーを分配することが可能になります。データベース・グローバル・メモリーの合計量は、データベースがアクティブである間は増やしたり減らしたりできませんが、データベース・グローバル・メモリーに含まれる領域のメモリーは調整できます。このような領域には、バッファー・プール、ロック・リスト、データベース・ヒープ、パッケージ・キャッシュ、およびカタログ・キャッシュがあります。データベース・マネージャーのパーティション間並列構成パラメーター (intra_parallel) が有効な環境、または接続コンセントレーターが使用可能な環境では、共有ソート・ヒープもデータベース・グローバル・メモリーの一部として割り振られます。
- **アプリケーションがデータベースに接続するとき:** パーティション・データベース環境、データベース・マネージャーのパーティション間並列構成パラメーター (*intra_parallel*) が有効な非パーティション・データベース環境、または接続コンセントレーターが使用可能な環境では、複数のアプリケーションをアプリケーション・グループ に割り当て、メモリーを共有させることができます。各アプリケーション・グループには、独自の共用メモリーの割り振りがあります。各アプリ

ケーションには独自のアプリケーション制御ヒープがありますが、アプリケーション・グループの共用メモリーでは、アプリケーション・グループの共用ヒープが使用されます。

アプリケーション・グループ・メモリーのサイズは、次の 3 つのデータベース構成パラメーターで決定されます。

- *appgroup_mem_sz* パラメーター - アプリケーション・グループの共用メモリーのサイズを指定する
- *groupheap_ratio* パラメーター - 共用ヒープにできるアプリケーション・グループ共用メモリーの割合 (%) を指定する
- *app_ctl_heap_sz* パラメーター - グループ内の各アプリケーションの制御ヒープのサイズを指定する

アプリケーション・メモリーの使用をグループ分けすることのパフォーマンス上の利点は、キャッシュやメモリーの使用効率を向上できる、という点です。

アプリケーション・グローバル・メモリーのいくつかのエレメントは、動的にサイズ変更することもできます。

- **エージェントが作成されるとき:** このイベントは、図には示されていません。エージェント専用メモリーは、並列環境における接続要求や新しい SQL 要求の結果としてエージェントが割り当てられるときにエージェントに割り振られます。このエージェント専用メモリーには、ソート・ヒープやアプリケーション・ヒープといった、この特定のエージェントでのみ使用される割り振りメモリーが含まれています。

  データベースがすでにある特定のアプリケーションで使用されている場合、後続の接続アプリケーションには、エージェント専用メモリーとアプリケーション・グローバル共用メモリーだけが割り振られます。

図には、それぞれの特定の目的のために割り振られるメモリーの量を制限する、以下の構成パラメーターの設定もリストされています。パーティション・データベース環境では、このメモリーが各データベース・パーティションに割り振られることを覚えておいてください。

- *numdb*

  このパラメーターは、別のアプリケーションで使用するために並行してアクティブにできるデータベースの最大数を指定します。各データベースにはそれぞれのグローバル・メモリー領域があるため、このパラメーターの値を大きくすると、割り振られるメモリーの量も大きくなる可能性があります。

- *maxappls*

  このパラメーターは、1 つのデータベースに同時に接続できるアプリケーションの最大数を指定します。この値は、そのデータベースのエージェント専用メモリーとアプリケーション・グローバル・メモリーに割り振られるメモリーの量に影響します。このパラメーターは、すべてのデータベースで個別に設定できることを覚えておいてください。

- *maxagents* および *max_coordagents* (並列処理用)

これらのパラメーターは、図には示されていません。これらのパラメーターは、インスタンス内でアクティブであるすべてのデータベースの間で同時に存在できるデータベース・マネージャー・エージェントの数を制限します。 *maxappls* と合わせて使用することにより、これらのパラメーターでは、エージェント専用メモリーとアプリケーション・グローバル・メモリーに割り振られるメモリーの量を制限できます。

関連概念:

- 240 ページの『データベース・マネージャー共用メモリー』
- 242 ページの『FCM バッファー・プールとメモリーの要件』
- 243 ページの『グローバル・メモリーとその制御パラメーター』
- 245 ページの『メモリー使用に影響を与えるパラメーターのチューニングのガイドライン』
- 37 ページの『メモリー管理』

データベース・マネージャー共用メモリー

メモリー・スペースは、データベース・マネージャーを実行するには必須のものです。このスペースは非常に大きくなる場合もあり、特にパーティション内並列処理およびパーティション間並列処理の場合などはそれが顕著です。

次の図は、アプリケーションのサポートにメモリーがどのように使用されるかを示しています。ここに示されている構成パラメーターでは、メモリー・セグメント (論理メモリーを成している各部分) の数とサイズを制限することにより、このメモリーのサイズを制御することを可能にします。

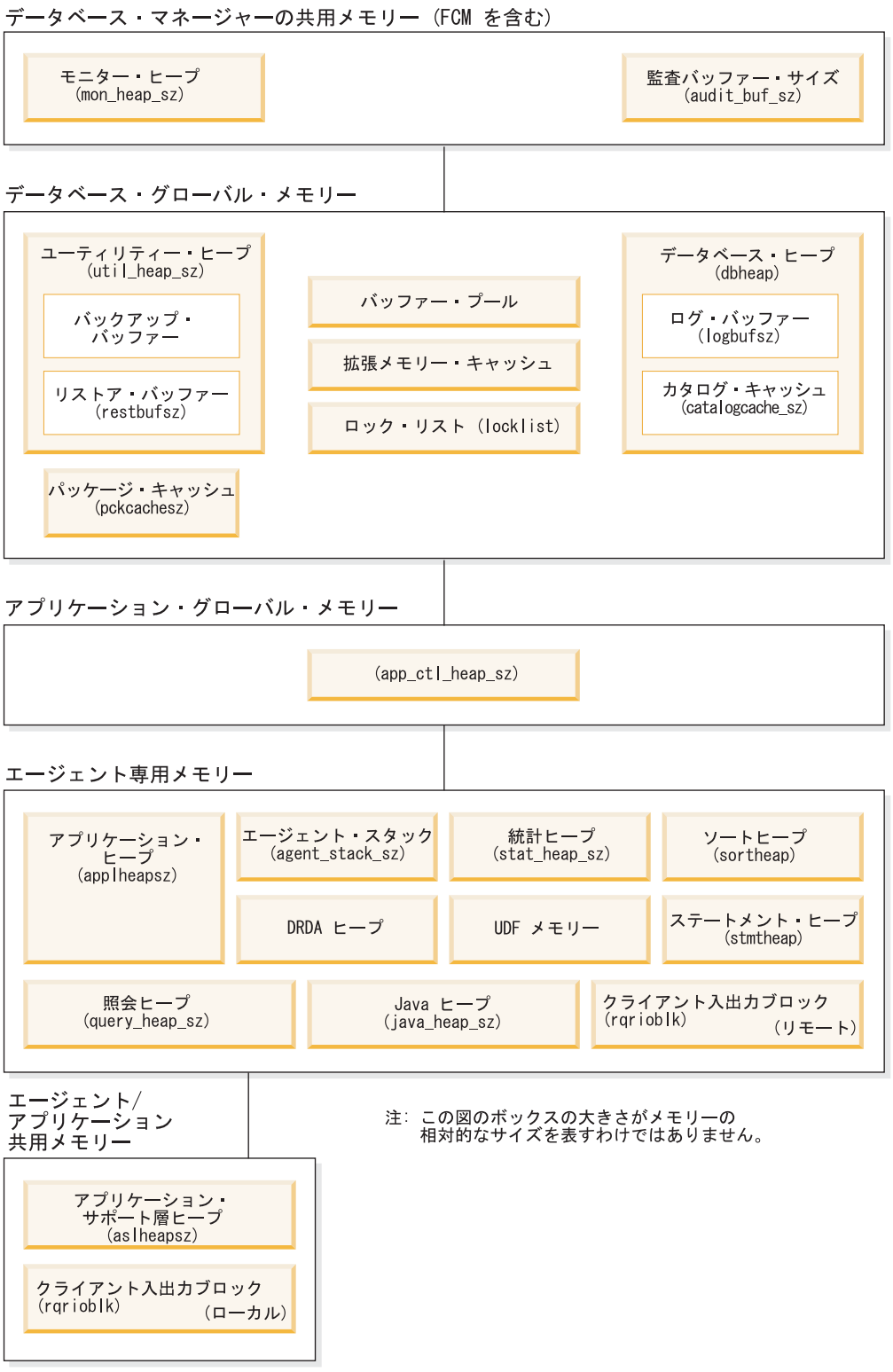


図 20. データベース・マネージャーでのメモリーの使用方法

このスペースのサイズは、データベース・エージェントについての情報を調べることによって予測および制御できます。アプリケーションの代理として実行されているエージェントは、 *maxagents* が非常に大きい値に設定されている場合には特に、相当量のメモリー・スペースを必要とします。これは一部のメモリーが、*maxagents*

の値に基づいて各エージェントに事前割り振りされるためです。 *maxagents* が必要以上に大きい値に設定されると、そのメモリーが事前割り振りはされても、実際にはデータベース・マネージャーによって使用されることはなく、結果としてメモリーの消費が無駄になります。

パーティション・データベース・システムでは、 *fcm_num_buffers* の値が大きい場合は特に、高速コミュニケーション・マネージャー (FCM) に相当量のメモリー・スペースが必要です。加えて、FCM のメモリー要件は、FCM バッファー・プールから割り振られるか、データベース・マネージャーの共用メモリーと FCM バッファー・プールの両方から割り振られます。どちらになるかは、パーティション・データベース・システムが複数の論理ノードを使うかどうかで決まります。

関連概念:

- 237 ページの『メモリー使用の編成』
- 243 ページの『グローバル・メモリーとその制御パラメーター』
- 248 ページの『バッファー・プール管理』
- 245 ページの『メモリー使用に影響を与えるパラメーターのチューニングのガイドライン』
- 37 ページの『メモリー管理』

FCM バッファー・プールとメモリーの要件

パーティション・データベース・システムに複数の論理ノードが存在しない場合、データベース・マネージャーの共用メモリーと FCM バッファー・プールは、下に示されているようになります。

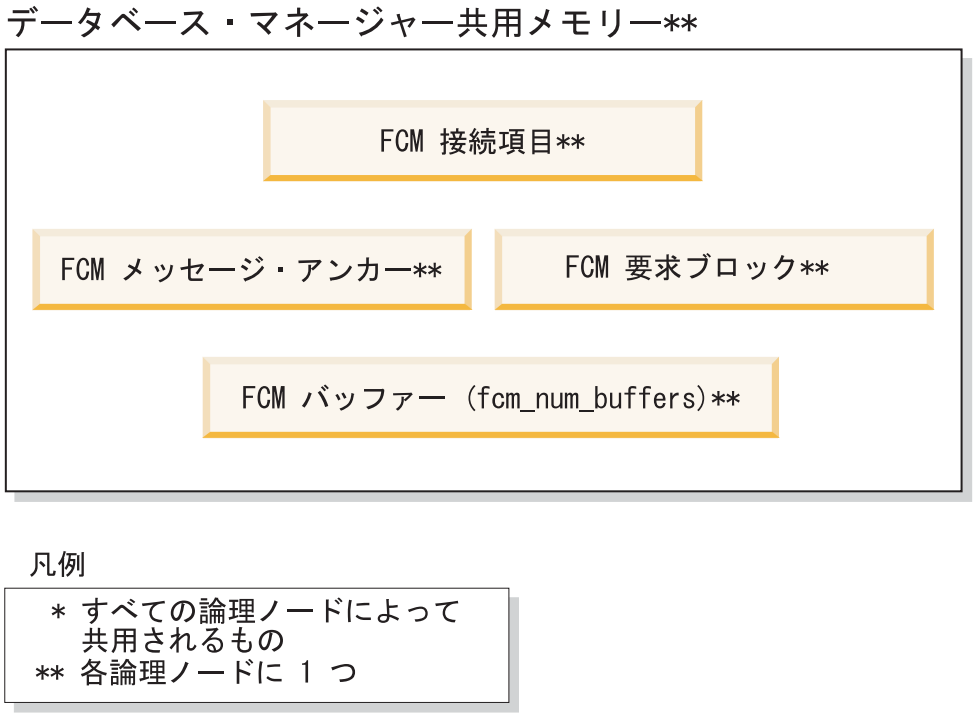


図 21. 複数の論理ノードが使用されていない場合の FCM バッファー・プール

パーティション・データベース・システムで複数の論理ノードが使用されている場合、データベース・マネージャーの共用メモリーと FCM バッファー・プールは下に示されているようになります。

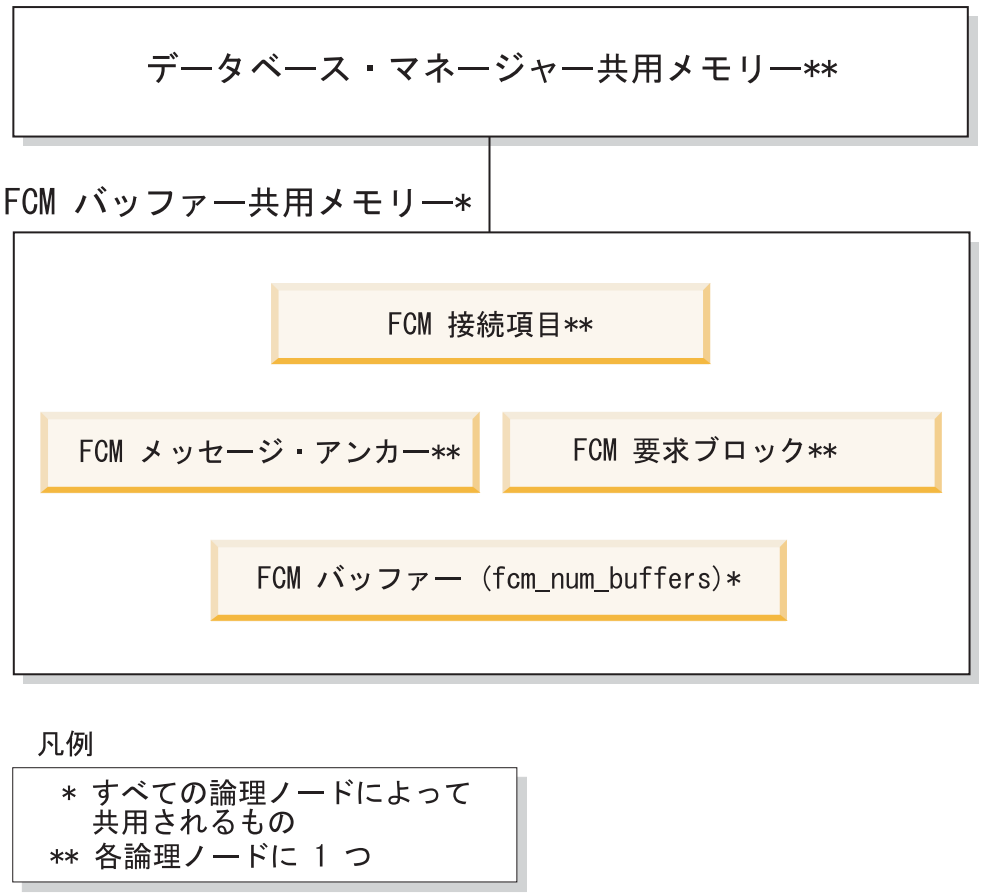


図 22. 複数の論理ノードが使用されている場合の FCM バッファー・プール

高速コミュニケーション・マネージャー (FCM) を構成する場合は、まず、FCM バッファー数 (*fcm_num_buffers*) のデフォルト値から始めてください。 FCM と AIX® プラットフォームについての詳細は、DB2_FORCE_FCP_BP レジストリー変数の説明を参照してください。

このパラメーターをチューニングするには、データベース・システム・モニターを使用して空きバッファーの最低水準点をモニターしてください。

関連概念:

- 240 ページの『データベース・マネージャー共用メモリー』

グローバル・メモリーとその制御パラメーター

データベース・マネージャーの共用メモリーは、以下のコンポーネントから成っています。

データベース・グローバル・メモリー
データベース・グローバル・メモリーは、以下の構成パラメーターから影響を受けます。

- *database_memory* パラメーター。データベース・グローバル・メモリーのサイズの下限を示します。
- 以下のパラメーターは、メモリー・セグメントの最大値を指定します。
  - バッファー・プールのサイズ
  - ロック・リスト用最大ストレージ (*locklist*)
  - データベース・ヒープ (*dbheap*)
  - ユーティリティー・ヒープ・サイズ (*util_heap_sz*)
  - 拡張記憶域メモリー・セグメント・サイズ (*estore_seg_sz*)
  - 拡張記憶域メモリー・セグメントの数 (*num_estore_segs*)
  - パッケージ・キャッシュ・サイズ (*pckcachesz*)
  - 共有ソート・ヒープ (*sheapthres_shr*)

アプリケーション・グローバル・メモリー

アプリケーション・グローバル・メモリーは、アプリケーション制御 (*app_ctl_heap_sz*) 構成パラメーターの影響を受けます。

並列システムの場合には、アプリケーション制御ヒープ用のスペースも必要になります。これは、1 つのデータベース・パーティションにおいて、同じアプリケーションのために作業を行うエージェントの間で共用されるものです。ヒープは、アプリケーションからの要求を受け取る最初のエージェントが接続を要求したときに割り振られます。エージェントは、コーディネーター・エージェントかサブエージェントのいずれかになります。

エージェント専用メモリー

- メモリー・セグメントの数は、次の値のうち小さい方の値によって制限されます。
  - すべてのアクティブ・データベースの *maxappls* 構成パラメーターの合計。許可されるアクティブ・アプリケーションの最大数を指定します。
  - *maxagents* 構成パラメーターの値。許可されるエージェントの最大数を指定します。
- メモリー・セグメントの最大サイズは、次のパラメーターの値によって決められます。
  - アプリケーション・ヒープ・サイズ (*applheapsz*)
  - ソート・ヒープ・サイズ (*sortheap*)
  - ステートメント・ヒープ・サイズ (*stmtheap*)
  - 統計ヒープ・サイズ (*stat_heap_sz*)
  - 照会ヒープ・サイズ (*query_heap_sz*)
  - エージェント・スタック・サイズ (*agent_stack_sz*)

エージェント/アプリケーション共用メモリー

- ローカル・クライアントの場合には、エージェント/アプリケーション共用メモリー・セグメントの合計数は、次のデータベース構成パラメーターのうちの小さい方の値によって制限されます。
  - すべてのアクティブなデータベースに設定された *maxappls* の合計。
  - 並列システムの *maxagents* または *max_coordagents* の値。

- エージェント/アプリケーション共用メモリーは以下のデータベース構成パラメーターからも影響を受けます。
  - アプリケーション・サポート層ヒープ・サイズ (*aslheapsz*) パラメーター
  - クライアント入出力ブロック・サイズ (*rqrioblk*) パラメーター

関連資料:

- 431 ページの『estore_seg_sz - 拡張ストレージ・メモリー・セグメント・サイズ』
- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 432 ページの『num_estore_segs - 拡張ストレージ・メモリー・セグメント数』
- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 441 ページの『maxagents - エージェントの最大数』
- 413 ページの『aslheapsz - アプリケーション・サポート層ヒープ・サイズ』
- 404 ページの『applheapsz - アプリケーション・ヒープ・サイズ』
- 394 ページの『pckcachesz - パッケージ・キャッシュ・サイズ』
- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 416 ページの『rqrioblk - クライアント入出力ブロック・サイズ』
- 389 ページの『dbheap - データベース・ヒープ』
- 412 ページの『stmtheap - ステートメント・ヒープ・サイズ』
- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 402 ページの『agent_stack_sz - エージェント・スタック・サイズ』
- 407 ページの『query_heap_sz - 照会ヒープ・サイズ』
- 397 ページの『util_heap_sz - ユーティリティー・ヒープ・サイズ』
- 411 ページの『stat_heap_sz - 統計ヒープ・サイズ』
- 388 ページの『database_memory - データベース共用メモリー・サイズ』

メモリー使用に影響を与えるパラメーターのチューニングのガイドライン

メモリーを割り振るパラメーターを設定する際は、絶対に許容最大値を設定しないでください。値を設定するときは、十分な注意が必要です。これが、メモリー割り振りパラメーターの設定についての第一の規則です。この規則は、メモリーの最大量を持つシステムにも当てはまります。メモリーに影響を与えるほとんどのパラメーターでは、使い方によっては、データベース・マネージャーがコンピューター上で使用可能な全メモリーを簡単かつ瞬時に使い果たす可能性があるからです。また、メモリー量が大きいとその管理のためにデータベース・マネージャーのほうで余計な作業が必要になり、オーバーヘッドが増大してしまいます。

UNIX® オペレーティング・システムの中には、プロセスがスワップ・スペースにページアウトされるときではなく、プロセスがメモリーを割り振る時点でスワップ・スペースを割り振るものがあります。このようなシステムの場合、共用メモリー・スペースの合計と同じ大きさのページング・スペースを確保するようにしてください。

ほとんどの構成パラメーターでは、メモリーは必要なときだけコミットされます。これらのパラメーターは特定のメモリー・ヒープの最大サイズを決めます。ただし、以下のような場合、パラメーターによって指定されたすべてのメモリーが割り振られます。

- ロック・リスト用最大ストレージ (*locklist*)
- アプリケーション・サポート層ヒープ・サイズ (*aslheapsz*)
- FCM バッファー数 (*fcm_num_buffers*)

注: バッファー・プールのサイズを変更するには、 DDL ステートメント ALTER BUFFERPOOL を使用します。

アプリケーション・グループ・メモリーの使用に影響を与えるパラメーター

アプリケーション・グループのメモリー使用に影響を与えるパラメーターは、パーティション・データベース、パーティション内並列処理が使用可能になっているデータベース、および接続コンセントレーターが使用可能になっているデータベースにのみ当てはまります。以下のパラメーターは、アプリケーション・グループ内のアプリケーションがその共用メモリーを使う方法を決定します。

- *appgroup_mem_sz* パラメーターは、アプリケーション・グループの共用メモリーのサイズを指定します。

  appgroup_mem_sz 構成パラメーターの設定値が高すぎると、悪い影響を及ぼします。アプリケーション・グループ内のすべてのアプリケーションが、アプリケーション・グループ・ヒープ内のキャッシュを共用するため、アプリケーションがあまり多すぎるとキャッシュの競合が多くなります。他方、各アプリケーション・グループに少ししかアプリケーションがないと、キャッシュの影響も限られます。

- *groupheap_ratio* パラメーターは、共用ヒープで使用可能なメモリーの比率を指定します。

  groupheap_ratio の設定値が低すぎると、キャッシュのサイズは限られてしまいます。 *groupheap_ratio* の設定値が高すぎると、アプリケーション制御ヒープは小さくなりすぎて、 SQL エラー SQL0973 が起きる可能性があります。このエラーは、ランタイム時に制御ヒープ・メモリーを使い果たしつつあることを警告するものです。

- *app_ctl_heap_sz* パラメーターは、グループ内の各アプリケーションごとに制御ヒープのサイズを指定します。

データベース・サーバーを構成する際は、これらのパラメーターのデフォルト設定値を受け入れてください。パフォーマンスが低下する場合にのみ設定値を調整してください。たとえば、*appgroup_mem_sz* を設定して、各アプリケーション・グループのアプリケーションの数を制御します。経験法則からすると、10 だと少なすぎますし、100 だと多すぎます。おそらく、デフォルトが妥当でしょう。その後、平均ワークロードを実行し、コントロール・センターからヘルス・センター・ユーティリティー、またはシステム・モニターを使用して、カタログ・キャッシュ、パッケージ・キャッシュ、および共用ワークスペースのヒット率に関する情報を収集します。

- sql0973 エラーが頻発する場合、 *groupheap_ratio* 設定値が高すぎます。

- アプリケーション制御ヒープの平均および最大使用量が app_ctl_heap_sz * (100 - groupheap_ratio)/ 100 と比べてはるかに少ないことをメモリー・トラッカーが示す場合、 *app_ctl_heap_sz* 構成パラメーターの値を小さくしてください。
- キャッシュが限界に達していることをキャッシュの使用が示す場合、 *group_heap_ratio* 構成パラメーターの値を大きくするか、アプリケーション・グループ内のアプリケーションの数を減らしてください。

注:

- ベンチマーク・テストは、メモリー・パラメーターの適切な値を設定するための最適な情報を提供します。ベンチマーク・テストでは、通常の SQL ステートメントおよび最悪ケースの SQL ステートメントをサーバーに対して実行し、そこでパラメーターの値を修正しながら、パフォーマンスがこれ以上あがらないポイントを見つけます。パフォーマンスとパラメーター値の関係をグラフで表すと、曲線が停滞または降下を始めるポイントが、割り振りを増加してもアプリケーションには利益をもたらさずに、単にメモリーの無駄使いになってしまうポイントということになります。
- パラメーターによっては、メモリー割り振りの上限が、現存のハードウェアおよびオペレーティング・システムのメモリー容量を超える場合があります。これらの限界値は、将来的な増加のためのものです。
- 有効なパラメーター範囲については、各パラメーターの詳細情報を参照してください。

関連概念:

- 237 ページの『メモリー使用の編成』
- 240 ページの『データベース・マネージャー共用メモリー』
- 243 ページの『グローバル・メモリーとその制御パラメーター』

関連資料:

- 398 ページの『app_ctl_heap_sz - アプリケーション・コントロール・ヒープ・サイズ』
- 519 ページの『fcm_num_buffers - FCM バッファー数』
- 408 ページの『sheapthres - ソート・ヒープしきい値』
- 413 ページの『aslheapsz - アプリケーション・サポート層ヒープ・サイズ』
- 394 ページの『pckcachesz - パッケージ・キャッシュ・サイズ』
- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 389 ページの『dbheap - データベース・ヒープ』
- 397 ページの『util_heap_sz - ユーティリティー・ヒープ・サイズ』
- 386 ページの『catalogcache_sz - カタログ・キャッシュ・サイズ』
- 399 ページの『appgroup_mem_sz - アプリケーション・グループ・メモリー・セットの最大サイズ』
- 401 ページの『groupheap_ratio - アプリケーション・グループ・ヒープ用メモリーのパーセント』

バッファー・プール

バッファー・プールは重要なメモリー・コンポーネントです。このセクションでは、バッファー・プールについて説明し、よいパフォーマンスを得るためにそれを管理することについての情報を提供しています。

バッファー・プール管理

バッファー・プールとは、表や索引のデータ・ページをディスクから読み取る際や変更する際に、これをキャッシュに入れておくためのメモリーのことをいいます。このバッファー・プールがあると、ディスク上ではなく、メモリー上のデータにアクセスできるため、データベース・システムのパフォーマンスが向上します。メモリーへのアクセスはディスクへのアクセスよりもはるかに速いため、データベース・マネージャーがディスクへの読み書きを行う必要が少なければ少ないほど、パフォーマンスは良くなります。ほとんどのデータ操作はバッファー・プール内で行われるため、バッファー・プールの構成は、単独の最も重要なチューニングの分野となっています。バッファー・プールで扱うことができないのは、ラージ・オブジェクトと長フィールド・データだけです。

アプリケーションが表のある行に最初にアクセスしたときに、データベース・マネージャーは、その行を含むページをバッファー・プールに入れます。何らかのアプリケーションが次にデータを要求してきたときには、データベース・マネージャーはまずバッファー・プール内にそのデータがないかどうかを確認します。要求データがバッファー・プール内にあれば、ディスクにアクセスすることなくデータを取り出せるため、それだけ応答は速くなります。

バッファー・プールへのメモリーの割り振りは、データベースが活動化されたときか、最初にアプリケーションがデータベースに接続したときに行われます。データベース・マネージャーが稼働しているときにバッファー・プールを作成、ドロップ、およびサイズ変更することも可能です。 ALTER BUFFERPOOL を使用してバッファー・プールのサイズを大きくするときに IMMEDIATE キーワードを使用すると、割り振り可能なメモリーがある限り、コマンドの入力後即時にメモリーが割り振られます。ただし、割り振り可能なメモリーがない場合は、すべてのアプリケーションの接続が切断され、データベースが再活動化されたときに変更が行われます。またバッファー・プールのサイズを小さくする場合は、コミットしたときにメモリーの割り振りが解除されます。そして、すべてのアプリケーションの接続が切断されると、バッファー・プールのメモリーが割り振り解放されます。

注: バッファー・プールのサイズを大きくする際に、可能な限り *dbheap* データベース構成パラメーターのサイズを大きくしなくて済むようにするため、ページ記述子、バッファー・プール記述子、およびハッシュ表を含むほぼすべてのバッファー・プール・メモリーは、データベース共用メモリー・セットから出され、自動的にサイズ調整されています。

あらゆる状況において適切なバッファー・プールが使用できるようにするため、DB2® は、それぞれ 4K、8K、16K、および 32K のページ・サイズを持つ小さなバッファー・プールを作成します。各バッファー・プールのサイズは 16 ページです。これらのバッファー・プールはユーザーからは隠されています。システム・カ

タログやバッファー・プール・システム・ファイルには示されません。これらのバッファー・プールは、直接使用または変更することはできませんが、以下のような場合に DB2 によって使用されます。

- IMMEDIATE キーワードを指定して CREATE BUFFERPOOL ステートメントが実行されたが、作成に使用できるメモリーが不足しているために、必要なページ・サイズのバッファー・プールが活動化されない場合。

  管理通知ログにメッセージが書き込まれます。そして、必要に応じ、隠れたバッファー・プールに表スペースが再マップされます。パフォーマンスは大幅に低下する可能性があります。
- データベース接続時に通常のバッファー・プールを得られない場合。

  この問題には、メモリー不足などの深刻な原因が関係していると思われます。隠れたバッファー・プールがあるため、DB2 は完全な機能を提供しますが、パフォーマンスは大幅に低下するでしょう。この問題は、直ちに対処を必要とします。この問題が発生すると、警告が出され、管理通知ログにメッセージが書き込まれます。

ページは、データベースがシャットダウンされるまで、あるいは、あるページが占めるスペースが別のページによって要求されるまで、バッファー・プール内に留まります。別のページをプールに入れるときに除去されるページは、次の基準で決定されます。

- そのページが最後に参照されてからの経過時間
- そのページを最後に参照したエージェントが再びページを参照する可能性
- ページ上のデータ・タイプ
- メモリー内で変更されたページがディスクに書き出されたかどうか (変更されたページは常に、上書きされる前にディスクに書き込まれます。)

再びメモリーからページにアクセスできるようにするため、変更されたページは、ディスクに書き出された後でも、スペースが必要にならない限りバッファー・プールからは除去されません。

バッファー・プールを作成するときには、 4 KB、8 KB、16 KB、32 KB (デフォルトは 4KB) の中から任意のページ・サイズを指定できます。ページは、表スペースのページ・サイズとバッファー・プールのページ・サイズが一致しないとバッファー・プールに読み込むことができないため、バッファー・プールのページ・サイズを指定する際には、使用している表スペースのページ・サイズを考慮してください。バッファー・プールのページ・サイズは、バッファー・プールを一度作成してしまうと後からは変更できません。ページ・サイズを変えるためには、異なるページ・サイズの新しいバッファー・プールを作成しなければなりません。

注: Windows® NT が稼働する 32 ビット・プラットフォームでは、 Address Windowing Extensions (AWE) が使用可能になっているか、 Windows 2000 上で Advanced Server と Data Center Server が使用可能になっていれば、大規模なバッファー・プールも作成できます。

関連概念:

- 237 ページの『メモリー使用の編成』

- 250 ページの『32 ビット・プラットフォームにおける拡張メモリー内の 2 次バッファー・プール』
- 251 ページの『データ・ページのバッファー・プール管理』
- 254 ページの『バッファー・プールのデータ・ページ管理の図』
- 255 ページの『複数のデータベース・バッファー・プールの管理』

32 ビット・プラットフォームにおける拡張メモリー内の 2 次バッファー・プール

64 ビット・プラットフォームでは、特別なテクニックを使わなくても、通常の方法で大規模な仮想アドレス可能メモリーを使用できます。しかし 32 ビット・プラットフォームでは、仮想アドレス可能メモリーは 2 GB から 4 GB に制限されています。ご使用の 32 ビット・マシンに、最大量を超える実アドレス可能メモリーがある場合は、仮想アドレス可能メモリーを超える分の実メモリーを*拡張記憶キャッシュ* として構成できます。拡張記憶キャッシュは、任意の定義されたバッファー・プールで、パフォーマンスを向上させるために使用できます。その場合は、拡張記憶キャッシュをいくつかのメモリー・セグメントとして定義します。

実アドレス可能メモリーの一部を拡張記憶キャッシュとして定義する場合、定義されたメモリーは、 JFS キャッシュやプロセス専用アドレス・スペースなどの他の目的には使用できなくなります。また、実アドレス可能メモリーを拡張記憶キャッシュに割り振ると、システム・ページングが増える可能性があります。

1 次レベルのキャッシュはバッファー・プールで実行されるため、すべての拡張記憶キャッシュは、2 次レベルのキャッシュとしてバッファー・プールで使用されます。理想としては、最もアクセス頻度の高いデータがバッファー・プールに保管され、比較的アクセス頻度の低いデータが拡張記憶キャッシュに保持されます。

注: Windows® 2000 Address Windowing Extensions (AWE) のバッファー・プールは、 DB2_AWE レジストリー変数を使用して割り振ることができます。Windows AWE というのは、アプリケーションで特定の制限 (アプリケーションのプロセス・モデルによって異なる) を超えたメモリー操作を行えるようにする、一連のメモリー管理用拡張機能です。詳細については、ご使用の Windows システムに付属している資料を参照してください。ただし、この目的でメモリーを使用すると拡張記憶キャッシュは使用できなくなりますので、ご注意ください。

以下に示すデータベース構成パラメーターは、拡張記憶で使用可能なメモリーの量とサイズに影響を与えます。

- *num_estore_segs* は、拡張記憶メモリー・セグメントの数を定義します。この構成パラメーターのデフォルトはゼロであり、これは拡張記憶キャッシュは持たないことを指定するものです。
- *estore_seg_sz* は、各拡張メモリー・セグメントのサイズを定義します。このサイズは、拡張記憶キャッシュが使用されるプラットフォームによって異なります。

拡張記憶キャッシュはバッファー・プールの延長なので、常に、1 つまたは複数の特定のバッファー・プールに関連付けておく必要があります。したがって、キャッシュを作成した場合には、どのバッファー・プールがキャッシュを利用することが

できるかを宣言する必要があります。 CREATE BUFFERPOOL ステートメントと ALTER BUFFERPOOL ステートメントには、キャッシュの使用について制御する属性 NOT EXTENDED STORAGE と EXTENDED STORAGE があります。デフォルトを使用すると、IBMDEFAULTBP および新しく作成されるバッファー・プールは、いずれも拡張記憶を使用しません。

注: 別のページ・サイズで定義されたバッファー・プールを使用する場合は、これらのバッファー・プールのうち任意のものを、拡張記憶域を使用するために定義できます。拡張記憶域サポートで使用されるページ・サイズは、定義される中で最大です。

データベース・マネージャーでは、拡張記憶キャッシュに常駐しているデータを直接操作することはできませんが、ディスク装置からバッファー・プールに転送するよりもはるかに速い速度で、データを拡張記憶キャッシュからバッファー・プールに転送することが可能です。

拡張記憶キャッシュ内のページのデータが 1 行必要になった場合は、そのページ全体が対応するバッファー・プールに読み込まれます。

バッファー・プールと、それに定義される関連拡張記憶キャッシュは、データベースが活動化されるときか最初の接続が行われるときに割り振られます。

関連概念:

- 248 ページの『バッファー・プール管理』
- 37 ページの『メモリー管理』

関連資料:

- 431 ページの『estore_seg_sz - 拡張ストレージ・メモリー・セグメント・サイズ』
- 432 ページの『num_estore_segs - 拡張ストレージ・メモリー・セグメント数』

データ・ページのバッファー・プール管理

バッファー・プールのページの状況は*使用中* かそうではないか、およびダーティーかクリーン かで区別されます。

- 「使用中」のページとは、現在読み取り中または更新中のページのことです。ページがエージェントによって使用中である場合、データベース内の他のエージェントやプリフェッチャーはそのページを読み取ることはできますが、更新はできません。
- 「ダーティー」のページというのは、変更されているがまだディスクに書き出されていないデータが含まれているページのことです。
- 変更されたページがディスクに書き出されると、そのページは「クリーン」になります。とはいえ、バッファー・プールには、別のページのためにそのスペースが必要になるまでページが残されます。クリーンなページは、関連付けられた拡張記憶キャッシュ (定義されている場合) に移行することもできます。

ページ・クリーナー・エージェント

調整状態の良いシステムでは、通常、変更されたページ、つまり「ダーティー」ページを、ページ・クリーナー・エージェントがディスクに書き込みます。ページ・クリーナー・エージェントは、バックグラウンド・プロセスとして入出力を実行し、実際のトランザクション作業をエージェントが実行できることでアプリケーションがより速く稼働できるようにします。ページ・クリーナー・エージェントは、他のエージェントの作業と連係しておらず、必要な場合にしか機能しないため、*非同期ページ・クリーナー* や*非同期バッファー書き込み機能* と呼ばれることもあります。

更新集中のワークロードでパフォーマンスを向上させるために、より多くのページ・クリーナー・エージェントを構成できます。ダーティー・ページをディスクに書き込むために使用できるページ・クリーナー・エージェントが多くなれば、パフォーマンスの向上が可能になります。これは、非同期のデータ・ページ書き込みや索引ページ書き込みの数に比べてデータ・ページ書き込みや索引ページ書き込みの数が多いことがスナップショットによって明らかである場合に、特に当てはまります。

ページ・クリーニングと高速リカバリー

システムがクラッシュしたときのデータベースのリカバリーは、より多くのページがディスクに書き込まれている方が速やかに行えます。より多くのページがディスクに書き込まれていれば、データベース・マネージャーは、データベース・ログ・ファイルからトランザクションを再生しなくても、バッファー・プールのより大きな部分をディスクから再構築できるからです。

リカバリー時に読み取らなければならないログのサイズは、ログの中の以下のレコードの位置によって異なります。

- 最も新しく書き込まれたログ・レコード
- バッファー・プール内で一番古い変更を記述するログ・レコード

ページ・クリーナーのデフォルトの動作は、リカバリー時に再生する必要のあるログのサイズが次の最大値を超えた場合に、ページ・クリーニングが実行されるというものです。

```
logfilsiz * softmax
```

説明:

- *logfilsiz* はログ・ファイルのサイズを表します。
- *softmax* は、データベース破壊が起きたらリカバリーする必要のあるログ・ファイルの比率 (パーセンテージ) を表します。 たとえば、*softmax* の値が 250 ならば、破壊が起きた場合にリカバリーする必要がある変更が 2.5 個のログ・ファイルに含まれていることになります。

リカバリー時のログの読み取り時間をできるだけ短くするため、データベース・システム・モニターを使用して、ページ・クリーニングが実行された回数をトラッキングしてください。データベースに対して先行ページ・クリーニングを使用可能にしていない場合には、システム・モニター *pool_lsn_gap_clns (起動されるバッファー・プール・ログ・スペース・クリーナー)* モニター・エレメントからこの情報が提供されます。この代替ページ・クリーニングが有効になっている場合は、この条件が起きることはなく、 *pool_lsn_gap_clns* モニター・エレメントは常に 0 です。

log_held_by_dirty_pages モニター・エレメントは、ユーザーによって設定されたリカバリー基準に見合う十分な量のページをページ・クリーナーがクリーニングしていないかどうかを判別するために使用できます。 *log_held_by_dirty_pages* が常に *logfilsiz* * *softmax* よりかなり大きい場合には、さらにページ・クリーナーが必要であるか、または *softmax* を調整することが必要です。

関連概念:

• 254 ページの『バッファー・プールのデータ・ページ管理の図』

関連資料:

• 591 ページの『パフォーマンス変数』

先行ページ・クリーニング

バージョン 8.1.4 から、システムにページ・クリーニングを構成する代替方法が提供されました。この代替方法は、特定の時点で書き出すダーティー・ページを選択する動作をページ・クリーナーが十分前もって行う点で、デフォルトの動作とは異なります。このページ・クリーニングの新しい方法がデフォルトのページ・クリーニングと異なっているのは、主に次の 2 つの点です。

1. ページ・クリーナーは chngpgs_thresh 構成パラメーターを重視しない。

   この代替方法では、ページ・クリーナーが chngpgs_thresh 構成パラメーターの値に応じて反応することはなくなっています。代替方法によるページ・クリーニングでは、バッファー・プールを一定の比率でクリーンに保つのではなく、スワップアウトされたページの位置を書き出した直後にエージェントに通知するというメカニズムが採用されています。そのため、スワップアウトするページをエージェントがバッファー・プールから検索する必要はありません。スワップアウトされるページの数が許容値よりも低いと、ページ・クリーナーが起動してバッファー・プール全体を検索し、スワップアウト可能なページを書き出し、それらのページの位置をエージェントに通知します。

2. ページ・クリーナーはロガーの発行する LSN ギャップ・トリガーに反応しない。

   バッファー・プール内の最も古いページを更新したログ・レコードから現行のログ位置までのログ・スペースの量が softmax パラメーターによって許可されている量を超えている状態を、データベースが「LSN ギャップ」状態であるといいます。デフォルトのページ・クリーニング方法では、ロガーが LSN ギャップ状態を検出すると、ページ・クリーナーを起動して、 LSN ギャップ状態を作り出しているページをすべて書き出します。つまり、softmax パラメーターが許可するより古いページを書き出します。 LSN ギャップが起きていない間は、ページ・クリーナーは活動を休止しています。そして、LSN ギャップが起きると、ページ・クリーナーは活動化して大量のページを書き出し、それが終わるとまた活動を休止します。こうした処理のために I/O サブシステムが飽和してしまい、その影響がページの読み書きをしている他のエージェントに及ぶ可能性があります。さらに、LSN ギャップになる以前に、ページ・クリーナーのクリーニング速度が遅いために、DB2® のログ・スペースが不足してしまうこともあり得ます。

ページ・クリーニングのこの代替方法では、長時間にわたって一定数の書き込みを継続することによって、動作を調節しています。そのために、現在 LSN ギャップ状態にあるページだけでなく、現在の活動レベルから判断してまもなく LSN ギャップ状態になりそうなページをも、十分前もってクリーニングします。

ページ・クリーニングのこの新しい方法を使用するには、DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数を「ON」に設定します。

関連概念:

- 251 ページの『データ・ページのバッファー・プール管理』

バッファー・プールのデータ・ページ管理の図

次の図は、ページ・クリーナー・エージェントとデータベース・エージェントとの間でのバッファー・プールの管理作業の分担状態と、データベース・エージェントがすべての入出力を行っている状態とを比較しています。

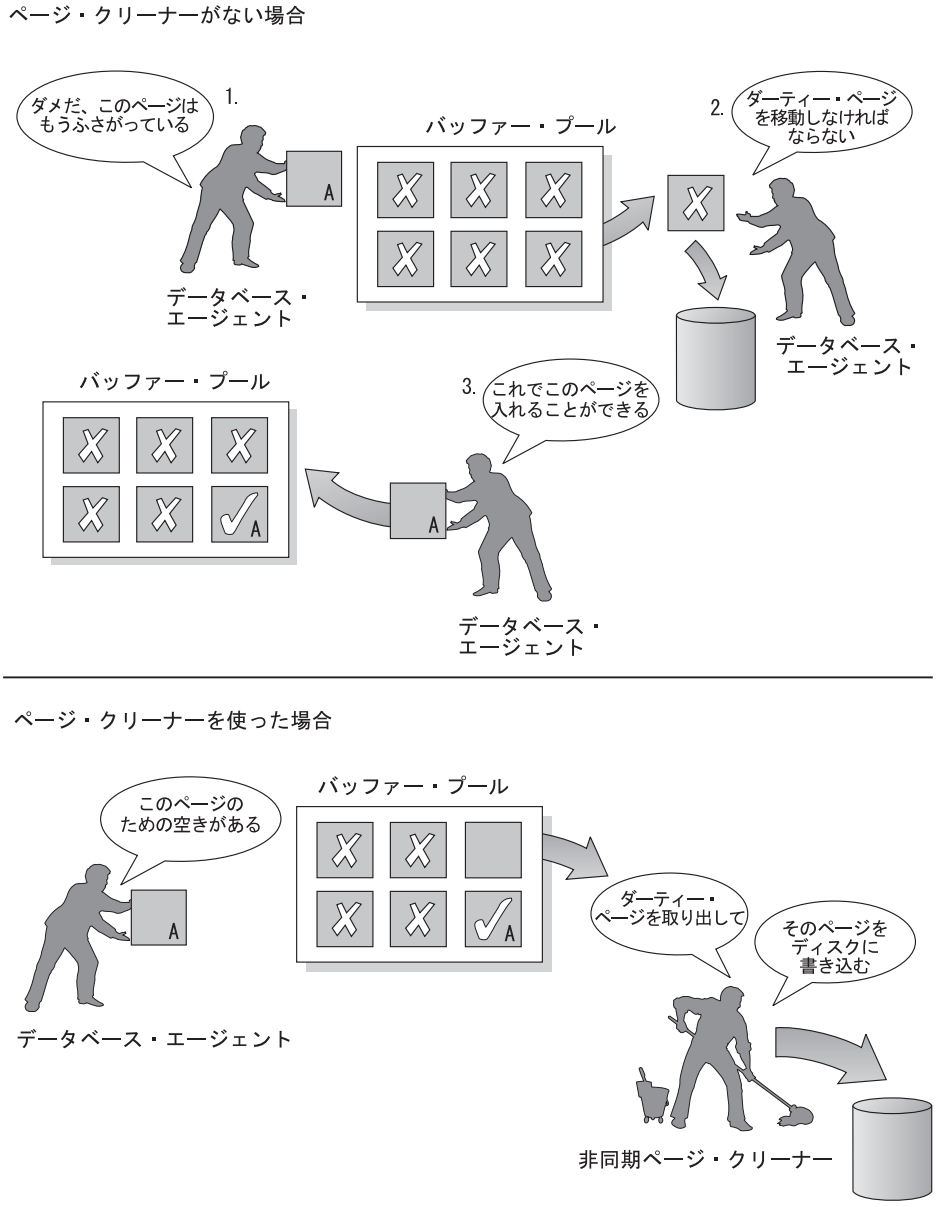


*図 23. 非同期*ページ・クリーナー： 「ダーティー」ページがディスクに書き出されます。

関連概念:

- 248 ページの『バッファー・プール管理』
- 251 ページの『データ・ページのバッファー・プール管理』

複数のデータベース・バッファー・プールの管理

各データベースは最低 1 つのバッファー・プールを必要としますが、複数のページ・サイズの表スペースを持つ 1 つのデータベースに対して、サイズやページ・サイズの異なる複数のバッファー・プールを作成することもできます。それぞれのバッファー・プールは最小サイズになります。この最小サイズは、プラットフォームによって異なります。

新規データベースは、IBMDEFAULTBP というデフォルト・バッファー・プールを持っています。そのサイズはプラットフォームによって決まり、そのデフォルト・ページ・サイズは 4 KB です。 4 KB のページ・サイズで表スペースを作成し、それを特定のバッファー・プールに割り当てていない場合、表スペースはデフォルトのバッファー・プールに割り当てられます。デフォルトのバッファー・プールのサイズ変更と、その属性の変更は可能ですが、それをドロップすることはできません。

注: 通常のデータベース・マネージャー操作では、 ALTER BUFFERPOOL コマンドを使ってバッファー・プールをサイズ変更することができます。

バッファー・プールのページ・サイズ

データベースを作成または移行した後、別のバッファー・プールを作成できます。たとえば、データベースを計画する際、 8 KB のページ・サイズが表に最適であると判断した場合、バッファー・プールも 8 KB のページ・サイズで作成し、なおかつ同じページ・サイズの 1 つ以上の表スペースも作成する必要があります。ALTER TABLESPACE ステートメントを使って、異なるページ・サイズを使用するバッファー・プールに表スペースを割り当てることはできません。

注: 4 KB より大きいページ・サイズ (8 KB、16 KB、32 KB など) で表スペースを作成する場合、同じページ・サイズを使用するバッファー・プールに割り当てる必要があります。このバッファー・プールが現在アクティブでない場合、DB2® は、同じページ・サイズを使用する別のアクティブ・バッファー・プール (もしあれば)、または最初のクライアントがデータベースに接続した際に DB2 が作成するデフォルトの「隠し」バッファー・プールに一時的に表スペースを割り当てようとします。データベースが再度アクティブにされ、最初に指定されたバッファー・プールがアクティブである場合、 DB2 は表スペースをそのバッファー・プールに割り当てます。

バッファー・プールを作成する際、バッファー・プールのサイズを DDL ステートメント CREATE BUFFERPOOL の必要パラメーターとして指定します。後でバッファー・プール・サイズを増やしたり減らしたりする場合は、 DDL ステートメント ALTER BUFFERPOOL を使用します。

パーティション・データベース環境の場合には、 1 つのデータベース用の各バッファー・プールのデフォルト定義は、全データベース・パーティションに対して同一になっています。ただし、それ以外の定義が CREATE BUFFERPOOL ステートメントに指定されている場合、または特定のデータベース・パーティションのバッファー・プールのサイズが ALTER BUFFERPOOL ステートメントによって変更された場合は除きます。

大容量バッファー・プールの利点

大容量バッファー・プールには次の利点があります。

- 頻繁に要求されるデータ・ページをバッファー・プール内に保持しておけるので、迅速にアクセスができるようになります。入出力操作を減らすと入出力の競合も減るので、結果として、応答時間が短縮したり、入出力操作に必要なプロセッサー・リソースを減らすことになります。
- 同じ応答時間で、より高いトランザクション率を達成する可能性があります。

- 頻繁に使用するカタログ表などのディスク記憶装置や頻繁に参照するユーザー表および索引に関する、入出力の競合を回避することができます。また、TEMPORARY 表スペースを含むディスク記憶装置上の入出力の競合が減少すると、照会によって要求されるソートの速度も速くなります。

バッファー・プールが多数ある場合の利点

使用しているシステムに以下の条件のいずれかが当てはまる場合には、単一のバッファー・プールのみを使用すべきです。

- 合計バッファー・スペースが 10 000 ページ (4 KB 単位) より小さい。
- アプリケーションの知識があって目的に合ったチューニングができる担当者がいない。
- テスト・システム上で作業中です。

それ以外のすべての状況では、複数のバッファー・プールの使用を検討してください。これは、次の理由によります。

- TEMPORARY 表スペースを別のバッファー・プールに割り当てることができるので、特にソートが多い照会など、一時記憶を必要とする照会のパフォーマンスを向上させることができます。
- たくさんの小さな更新トランザクション・アプリケーションから繰り返し迅速にアクセスする必要があるデータの場合には、そのデータを含む表スペースを別のバッファー・プールに割り当てることを検討してください。このバッファー・プールのサイズが適切ならば、ページが見つかる可能性が高くなるので、応答時間を短縮したりトランザクション・コストを下げるのに役立ちます。
- データを別のバッファー・プールに分離し、特定のアプリケーション、データ、索引を特別扱いにすることができます。たとえば、頻繁に更新される表や索引を入れるバッファー・プールを、頻繁に照会されるけれども更新は多くない表や索引のバッファー・プールとは別々にしたいという場合などに役立ちます。このようにすると、表の最初のセット上の頻繁な更新が表の 2 番目のセット上の頻繁な照会に与える影響を減らすことができます。
- めったに使用されないアプリケーションによってアクセスされるデータに対しては、小さなバッファー・プールを使用することができます。特に、非常に大きな表に対して非常にランダムなアクセスを要求するアプリケーションの場合は有効です。このような場合には、1 回の照会分より長い時間、バッファー・プール内にデータを保持しておく必要はありません。このデータには小さなバッファー・プールを用意して、それで余ったメモリーは他の用途 (たとえば、他のバッファー・プール) のために空けておくのが望ましいです。
- 活動やデータを種類によって別々のバッファー・プールにソートして入れると、パフォーマンスは向上はしたが相対的にコストがかかっていないという診断データが、統計および会計トレースから出される場合があります。

開始時のバッファー・プール・メモリーの割り振り

CREATE BUFFERPOOL コマンドを使ってバッファー・プールを作成するか、ALTER BUFFERPOOL コマンドを使ってバッファー・プールを変更する場合は、データベースの開始時にすべてのバッファー・プールの割り振りができるように、すべてのバッファー・プールで必要なメモリーの合計分がデータベース・マネージャーに対して使用可能になっている必要があります。データベース・マネージャーが

オンラインになっている間にバッファー・プールを作成または変更する場合、データベース・グローバル・メモリーの追加メモリーが使用できるようになっている必要があります。新規バッファー・プールを作成する際、または既存のバッファー・プールのサイズを増やす際に IMMEDIATE キーワードを指定し、その際必要とするメモリーが使用できない場合、データベース・マネージャーは、次にデータベースが活動化される際に変更を加えます。 32 ビット・プラットフォームの場合は、メモリーは使用可能になっているはずなので、グローバル・データベース・メモリーで予約することができます。これについては、 *database_memory* データベース構成パラメーターの詳細情報で説明されています。

データベースの開始時にこのメモリーが使用可能になっていない場合、データベース・マネージャーは別のページ・サイズで定義されているいずれかのバッファー・プールの開始を試みます。ただし、どのバッファー・プールも最小サイズの 16 ページでのみ開始します。異なる最小バッファー・プール・サイズを指定するには、DB2_OVERRIDE_BPF レジストリー変数を使用します。開始時にバッファー・プールを割り当てられないときは、 SQL1478W (SQLSTATE 01626) という警告が戻されます。データベースは、データベースの構成が変更されてデータベースが完全に再始動可能になるまでは、この操作状態で続行されます。

最小サイズの値でのみデータベース・マネージャーを開始するのは、データベース接続を可能にし、バッファー・プール・サイズを再構成して他の重要なタスクを実行できるようにするためです。これらのタスクを実行してすぐ、データベースを再始動します。そうした状態で、長時間に渡るデータベースの操作は行わないでください。

関連概念:

- 248 ページの『バッファー・プール管理』
- 250 ページの『32 ビット・プラットフォームにおける拡張メモリー内の 2 次バッファー・プール』

プリフェッチの概念

通常、データをバッファー・プールにプリフェッチすると、ディスク・アクセスの数が減り、頻繁にアクセスするデータがメモリー内に保存されることにより、パフォーマンスが向上します。

バッファー・プールへのデータのプリフェッチ

ページのプリフェッチ とは、いくつかのページがアプリケーションで必要になることを想定してディスクからリトリーブしておくということです。索引ページおよびデータ・ページをバッファー・プールにプリフェッチすると、入出力の待ち時間が減るので、パフォーマンスは向上します。加えて、並列入出力によってもプリフェッチの効率は拡張されます。

プリフェッチには、2 つのカテゴリーがあります。

- **順次プリフェッチ:** これは、アプリケーションでページが必要になる前に、連続したページをバッファー・プールに読み込む機構のことです。

- **リスト・プリフェッチ:** これは、*リスト順次プリフェッチ* とも呼ばれます。連続していないデータ・ページを効率的にプリフェッチします。

これらの 2 つのデータ・ページ読み取り方法は、通常の読み取りを補うものです。通常の読み取りは、1 ページまたは少数の連続ページがリトリーブされるときにのみ使用されます。通常の読み取り時には、データの 1 ページが転送されます。

プリフェッチおよびパーティション内並列処理

プリフェッチは、パーティション内並列処理 (索引または表のスキャン時に複数のサブエージェントを使用する操作) のパフォーマンスにとっては重要です。これらの並列スキャンによりデータの使用速度が高くなり、高速のプリフェッチが必要になります。

プリフェッチが不十分だと、順次スキャンよりも並列スキャンのほうがコストが高くつきます。順次スキャンでプリフェッチが行われない場合には、エージェントが常に入出力を待機しなければならないので、照会の実行速度が遅くなります。並列スキャンでプリフェッチが行われない場合には、 1 つのサブエージェントが入出力を待機するので、他のすべてのサブエージェントも待機しなければならなくなります。

パーティション内並列処理の場合にはプリフェッチの重要性が高いため、並列処理はより積極的に実行されます。順次検出機構は、隣接ページ間に大きなギャップがあってもそれを許容し、それらのページは順次であると見なされます。これらのギャップの幅は、スキャンに関係するサブエージェントの数が多いほど大きくなります。

関連概念:

- 248 ページの『バッファー・プール管理』
- 259 ページの『順次プリフェッチ』
- 263 ページの『リスト・プリフェッチ』
- 263 ページの『プリフェッチと並列処理のための入出力サーバー構成』
- 264 ページの『並列入出力を使用したプリフェッチの図』

順次プリフェッチ

1 回の入出力操作で複数の連続したページをバッファー・プールに読み込むと、アプリケーションのオーバーヘッドは大幅に短縮されます。加えて、複数の並列入出力操作で複数のページ範囲をバッファー・プールに読み込むことによっても、入出力の待ち時間を短縮することができます。

プリフェッチが開始されるのは、データベース・マネージャーが、順次入出力が適切であり、プリフェッチを行うとパフォーマンスが向上すると判断したときです。表スキャンや表ソートなどの場合、データベース・マネージャーは、順次プリフェッチにより入出力のパフォーマンスが向上することを容易に判断できます。このような場合は、データベース・マネージャーは自動的に順次プリフェッチを開始します。たとえば、以下の例ではおそらく表スキャンが必要になるので、順次プリフェッチが実行されます。

```
SELECT NAME FROM EMPLOYEE
```

表スペースの PREFETCHSIZE の含意

それぞれの表スペースごとにプリフェッチされるページ数を定義するには、CREATE TABLESPACE または ALTER TABLESPACE ステートメントのいずれかのステートメントの PREFETCHSIZE 文節を使用します。指定した値は、SYSCAT.TABLESPACES システム・カタログ表の PREFETCHSIZE 列に格納されます。

PREFETCHSIZE の値には、表スペース・コンテナーの数、各コンテナーの物理ディスクの数 (RAID 装置を使用している場合)、および表スペースの EXTENTSIZE の値を掛け合わせた値を、明示的に指定することをお勧めします。これは、データベース・マネージャーが別のコンテナーを使用する前にコンテナーに書き込むページ数です。たとえば、エクステント・サイズが 16 ページで、表スペースがコンテナーを 2 つ持っている場合には、プリフェッチ量を 32 ページに設定することができます。各コンテナーに 5 つの物理ディスクがある場合は、プリフェッチ量を 160 ページに設定できます。

データベース・マネージャーは、バッファー・プールの使用をモニターしているので、プリフェッチしたために、別の作業単位で必要なページが、バッファー・プールから除去されることはありません。データベース・マネージャーは、問題を避けるために、プリフェッチするページ数を、ユーザーが表スペースに対して指定するページ数未満に制限することができます。

プリフェッチ・サイズによっては、特に大きな表のスキャン時に、パフォーマンスを大幅に向上させることができます。ユーザーの表スペース用に指定された PREFETCHSIZE を調整するために、データベース・システム・モニターおよび他のシステム・モニター・ツールを使用します。次のような情報を得ることができます。

- オペレーティング・システムで使用可能なモニター・ツールを使用して、照会時に入出力待機時間が生じているかを調べることができます。
- データベース・システム・モニターによって提供された *pool_async_data_reads (バッファー・プール非同期データ読み取り)* データ・エレメントを調べることによって、プリフェッチが行われているかを知ることができます。

照会時にデータのプリフェッチが行われているのにまだ入出力の待機時間が生じている場合は、 PREFETCHSIZE の値を増やすことができます。プリフェッチャー以外の原因により入出力待機が生じている場合、 PREFETCHSIZE の値を増やしても照会のパフォーマンスは向上しません。

どのタイプのプリフェッチでも、プリフェッチ・サイズが表スペースのエクステント・サイズの倍数になっており、表スペースのエクステントが別個のコンテナーにあるときには、複数の入出力操作を並列に実行することができます。より高いパフォーマンスを得るには、コンテナーが別個の物理装置を使用するように構成します。

順次検出

順次プリフェッチがパフォーマンスの向上につながるかどうかが、瞬時には判断しにくい場合があります。このような場合、データベース・マネージャーが入出力をモニターし、ページが順次に読み取られている場合にプリフェッチを活動化するこ

とができます。このように行うプリフェッチでは、データベース・マネージャーがプリフェッチを活動化すべきと判断したときは活動化し、非活動化すべきと判断したときに非活動化することができます。このタイプの順次プリフェッチのことを*順次検出* といい、索引ページとデータ・ページの両方に適用されます。 *seqdetect* 構成パラメーターを使用して、データベース・マネージャーが順次検出を実行する必要があるかどうか制御できます。

たとえば、順次検出がオンになっている場合には、以下のような SQL ステートメントは順次プリフェッチを用いると効果があります。

```
SELECT NAME FROM EMPLOYEE
WHERE EMPNO BETWEEN 100 AND 3000
```

この例の場合、オプティマイザーは、EMPNO 列の索引を使って表スキャンを開始することがあります。この索引に関して表が高度にクラスター化されていると、データページの読み取りはほぼ順次になるので、プリフェッチによってパフォーマンスが向上する可能性があります。これによって、データ・ページ・プリフェッチが行われます。

この例では、索引ページのプリフェッチが行われます。大量の索引ページを検査する必要があるときに、データベース・マネージャーが索引ページの順次ページ読み取りが行われていることを見つけた場合には、索引ページのプリフェッチが行われます。

関連概念:

- 248 ページの『バッファー・プール管理』
- 258 ページの『バッファー・プールへのデータのプリフェッチ』
- 263 ページの『リスト・プリフェッチ』
- 261 ページの『改善された順次プリフェッチ用のブロック・ベースのバッファー・プール』

改善された順次プリフェッチ用のブロック・ベースのバッファー・プール

ディスクからページをプリフェッチすると、入出力オーバーヘッドのためにコストがかかります。入出力処理が並行して行われれば、スループットが非常に改善されます。多くのプラットフォームでは、ディスク上の連続する複数ページを読み取り、メモリーの不連続部分に格納するハイパフォーマンスのプリミティブが提供されています。このようなプリミティブを、通常、*散在データ読み取り* またはベクトル*化入出力* といいます。一部のプラットフォームでは、これらのプリミティブのパフォーマンスが、大きなブロック・サイズによる入出力処理よりも劣ります。

デフォルトでは、バッファー・プールはページに基づいています。つまりディスク上の連続する複数ページは、メモリー内の不連続ページとしてプリフェッチされます。ディスク上の連続するページを読み取り、連続するページとしてバッファー・プール内に格納することができれば、順次プリフェッチが拡張されます。

このような目的で、ブロック・ベースのバッファー・プールを作成することができます。ブロック・ベースのバッファー・プールは、ページ・エリアとブロック・エリアから構成されます。ページ・エリアは、非順次プリフェッチ・ワークロード用

に必要とされます。ブロック・エリアは複数のブロックで構成され、各ブロックには、指定された数の連続するページ (*ブロック・サイズ* という) が含まれます。

ブロック・ベースのバッファー・プールの最適な使用法は、指定するブロック・サイズによって異なります。ブロック・サイズは、順次プリフェッチを行う入出力サーバーがブロック・ベースの入出力を処理する際の細分度です。エクステントは、コンテナー間で表スペースがストライピングされるときの細分度です。ブロック・サイズが等しく定義された 1 つのバッファー・プールに、エクステント・サイズの異なる複数の表スペースをバインドできます。このため、エクステント・サイズとブロック・サイズを活用すれば、バッファー・プール・メモリーを効率的に使用することができます。以下のような状況では、バッファー・プール・メモリーがむだに消費される場合があります。

- (プリフェッチ要求サイズを決定する) エクステント・サイズが、バッファー・プールに指定された BLOCK_SIZE より小さい場合。
- プリフェッチ要求されたページのうち、いくつかのページがバッファー・プールのページ・エリアにすでに存在する場合。

入出力サーバーは各バッファー・プール・ブロックの一部のページがむだに消費されるのを許容しますが、ブロックのかなりの部分がむだに消費される場合、入出力サーバーはブロックに基づかないでバッファー・プールのページ・エリアにプリフェッチします。これは、最適なパフォーマンスではありません。

パフォーマンスを最適化するには、エクステント・サイズが等しいすべての表スペースを、表スペース・エクステント・サイズと同じブロック・サイズを持つバッファー・プールにバインドしてください。エクステント・サイズがブロック・サイズより大きい場合にも良好なパフォーマンスが得られますが、ブロック・サイズよりも小さい場合にはパフォーマンスが低下します。

ブロック・ベースのバッファー・プールを作成するには、 CREATE ステートメントおよび ALTER BUFFERPOOL ステートメントを使用します。ブロック・ベースのバッファー・プールには、以下のような制限があります。

- バッファー・プールをブロック・ベースすると、同時に拡張ストレージを使用することはできません。
- バッファー・プールで、ブロック・ベースの入出力と AWE サポートを同時に使用することはできません。特定のバッファー・プールに関して両方が使用可能になっている場合、 AWE サポートがブロック・ベースの入出力よりも優先されます。この状況では、そのバッファー・プールに関して、ブロック・ベースの入出力サポートは使用できません。 AWE サポートを使用不可にすると、これが再び使用可能になります。

注: ブロック・ベースのバッファー・プールの用途は、順次プリフェッチです。アプリケーションが順次プリフェッチを使用しない場合、バッファー・プールのブロック・エリアはむだになります。

関連概念:

- 248 ページの『バッファー・プール管理』
- 258 ページの『バッファー・プールへのデータのプリフェッチ』
- 259 ページの『順次プリフェッチ』

リスト・プリフェッチ

リスト・プリフェッチ または *リスト順次プリフェッチ* とは、必要なデータ・ページが連続していない場合でも、効果的にデータ・ページにアクセスする方法の 1 つです。リスト・プリフェッチは、単一または複数の索引アクセスで用います。

オプティマイザーが索引を使用して行にアクセスする場合、すべての行 ID (RID) が索引から取得されるまで、データ・ページの読み取りを据え置くことがあります。たとえば、オプティマイザーは以前定義した索引 IX1 で索引スキャンを実行することによって、取り出す行およびデータ・ページを決めます。

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

および、次の探索基準があるとします。

```
WHERE NAME BETWEEN 'A' and 'I'
```

この索引に従ってデータがクラスター化されていない場合、リスト・プリフェッチには、索引スキャンから獲得された RID のリストをソートするステップが含まれることになります。

関連概念:

- 248 ページの『バッファー・プール管理』
- 258 ページの『バッファー・プールへのデータのプリフェッチ』
- 259 ページの『順次プリフェッチ』

入出力管理

このセクションでは、入出力サーバーを調整して最適のパフォーマンスを得る方法について説明します。

プリフェッチと並列処理のための入出力サーバー構成

プリフェッチを使用可能にするため、データベース・マネージャーは、*入出力サーバー* と呼ばれる別の制御スレッドを開始してデータ・ページを読み取ります。その結果、照会処理は 2 つの並列のアクティビティー、つまりデータ処理 (CPU) とデータ・ページ入出力に分けられます。入出力サーバーは、CPU の処理アクティビティーからプリフェッチ要求が出るまで待機します。このプリフェッチ要求には、照会を満たすために必要な入出力記述が含まれます。データベース・マネージャーがプリフェッチ要求をいつどのように実行するかは、使用できるプリフェッチ方式によって異なります。

num_ioservers 構成パラメーターを使用して、十分な数の入出力サーバーを構成すると、データのプリフェッチが適用される照会のパフォーマンスは大幅に向上します。並列入出力の機会を最大にするには、 *num_ioservers* をデータベース内の物理ディスクの数以上に設定します。

入出力サーバーの数は、少なく評価するよりも多めに評価するほうが無難です。余分の入出力サーバーを指定した場合、これらのサーバーは使用されず、そのメモリー・ページはページアウトされます。その結果、パフォーマンスには影響しません。各入出力サーバー・プロセスには、番号が付けられます。データベース・マネージャーは、常に最も低い番号のプロセスを使用するため、比較的高い番号のプロセスは、まったく使用されない場合があります。

必要な入出力サーバーの数を見積もる場合は、以下の点を考慮してください。

- 入出力サーバーのキューに同時にプリフェッチ要求を書き込めるデータベース・エージェントの数。
- 入出力サーバーが並列で作業できる最高度。

非同期入出力の構成

一部のプラットフォームでは、ページ・クリーニングやプリフェッチといったアクティビティーのパフォーマンスを向上させるために、 DB2® が非同期入出力 (AIO) を使用します。 AIO は、コンテナーに入っているデータが複数のディスクに分配される場合に最も有効です。パフォーマンスは、基礎のオペレーティング・システム AIO インフラストラクチャーをチューニングすることによっても改善されます。

たとえば、AIX® では、オペレーティング・システムで AIO のチューニングを行うこともできます。 AIO が SMS ファイル・コンテナーと DMS ファイル・コンテナーの両方で稼働する場合は、 AIO サーバーというオペレーティング・システム・プロセスが入出力を管理します。このようなサーバーでは、まれに、AIO 要求の数が制限されることによって AIO の利点が制限される場合があります。 AIX 上の AIO サーバーの数を構成するには、 *smit* AIO *minservers* および *maxservers* パラメーターを使用します。

関連概念:

- 99 ページの『アプリケーションの並列処理』
- 264 ページの『並列入出力を使用したプリフェッチの図』
- 266 ページの『並列入出力の管理』

関連資料:

- 434 ページの『num_ioservers - I/O サーバーの数』

並列入出力を使用したプリフェッチの図

次の図は、データのプリフェッチを行ってバッファー・プールに入れる際に、入出力サーバーがどのように使用されるかについて示しています。

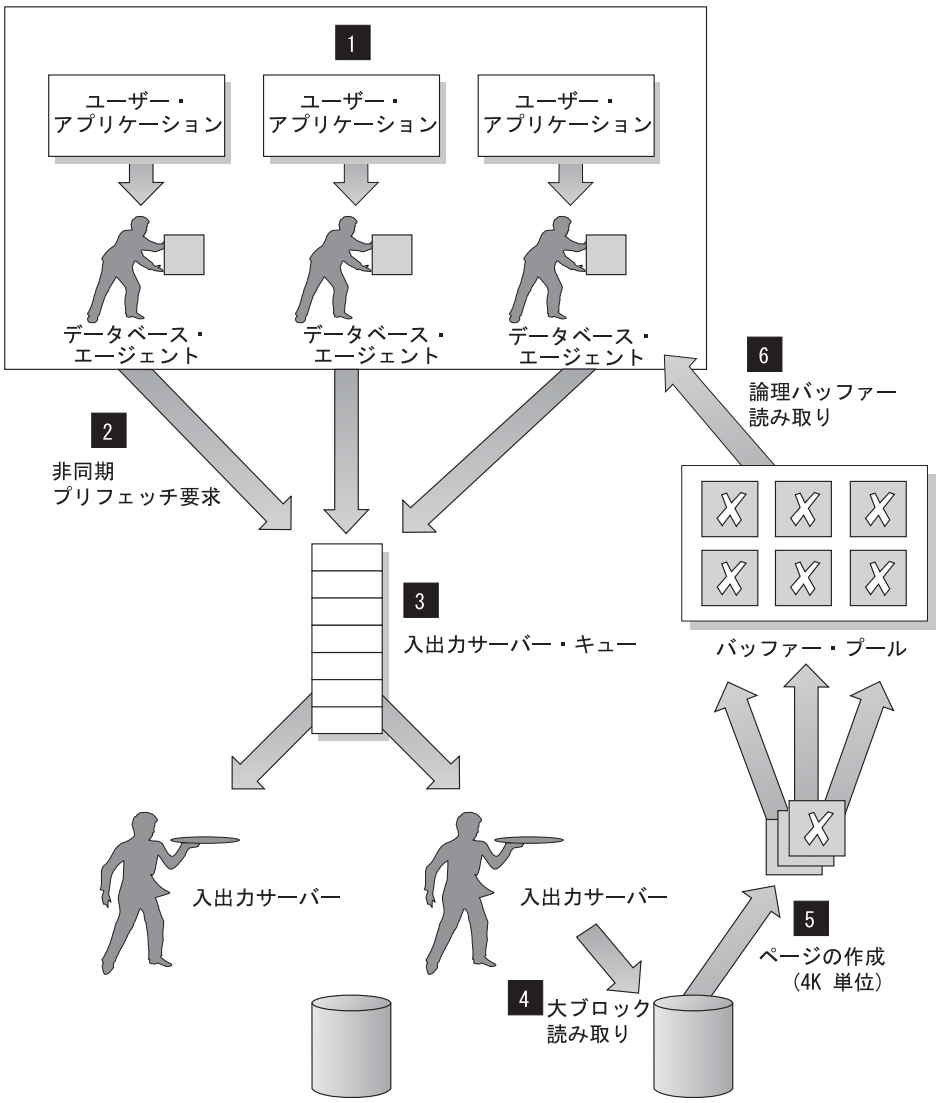


図 24. 入出力サーバーを使用したデータのプリフェッチ

1 ユーザー・アプリケーションが、データベース・マネージャーによってユーザー・アプリケーションに割り当てられているデータベース・エージェントに SQL 要求を渡します。

2、**3**

データベース・エージェントが、SQL 要求を満たすのに必要なデータを獲得するために、プリフェッチを使用しなければならないかどうかを判別し、プリフェッチ要求を入出力サーバー・キューに書き込みます。

4、**5**

利用可能な最初の入出力サーバーが、キューからプリフェッチ要求を読み取り、表スペースのデータをバッファー・プールに読み込みます。表スペースから同時にデータを取り出せる入出力サーバーの数は、キューに入っているプリフェッチ要求の数と *num_ioservers* データベース構成パラメーターによって構成される入出力サーバーの数に依存しています。

6 データベース・エージェントが、バッファー・プール内のデータ・ページに対して必須の走査実行し、その結果をユーザー・アプリケーションに戻します。

関連概念:

- 258 ページの『バッファー・プールへのデータのプリフェッチ』
- 259 ページの『順次プリフェッチ』
- 263 ページの『リスト・プリフェッチ』
- 263 ページの『プリフェッチと並列処理のための入出力サーバー構成』
- 266 ページの『並列入出力の管理』
- 298 ページの『パーティション・データベースにおけるエージェント』

並列入出力の管理

1 つの表スペースに対して複数のコンテナーが存在する場合、データベース・マネージャーは*並列入出力* を開始することができます。データベース・マネージャーはこの並列入出力において複数の入出力サーバーを使用して、 1 回の照会で入出力要件を処理します。各入出力サーバーは別個のコンテナーごとに入出力ワークロードを処理するため、複数のコンテナーを並列で読み取ることができます。入出力を並列で実行すると、入出力スループットが大幅に向上することになります。

別個の入出力サーバーはコンテナーごとにワークロードを処理することができますが、入出力を並列で実行できる入出力サーバーの実際の数は、要求されたデータが伝搬される物理装置の数に制限されます。そのため、物理装置と同じ数の入出力サーバーが必要になります。

並列入出力の開始は、以下に挙げる状況によりそれぞれ異なります。

- **順次プリフェッチ**

  順次プリフェッチの場合、プリフェッチ・サイズが表スペースのエクステント・サイズの倍数になっていると、並列入出力が開始されます。それぞれのプリフェッチ要求は、エクステント境界に沿って、多くの小さい要求に分割されます。その後、これらの小さい要求は別々の入出力サーバーに割り当てられます。
- **リスト・プリフェッチ**

  リスト・プリフェッチの場合、ページの各リストは、データ・ページが保管されるコンテナーに従って、小さいリストに分割されます。その後、これらの小さいリストは別々の入出力サーバーに割り当てられます。
- **データベースまたは表スペースのバックアップとリストア**

  データのバックアップまたはリストアの場合、並列入出力要求の数は、コンテナーの数と同じ最大値までに相当するエクステント・サイズに分割された、バックアップ・バッファー・サイズと等しくなります。
- **データベースまたは表スペースのリストア**

データのリストアの場合、並列入出力要求が開始され、順次プリフェッチで使用されるのと同じ方法で分割されます。データをバッファー・プールにリストアする代わりに、そのデータはリストア・バッファーからディスクに直接読み取られます。

- **ロード**

  データをロードする場合、LOAD コマンドの DISK_PARALLELISM オプションを使用すると、入出力並列処理のレベルを指定することができます。このオプションを指定しない場合は、データベース・マネージャーは、その表に関連するすべての表スペースの表スペース・コンテナーの累計値に基づくデフォルト値を使用します。

並列入出力で最適なパフォーマンスを得るには、以下を確認してください。

- 十分な入出力サーバーがあること。入出力サーバーの数は、データベース内のすべての表スペースに使用されるコンテナーの数より少し多めに指定します。
- エクステント・サイズおよびプリフェッチ・サイズが表スペースに適していること。バッファー・プールの使い過ぎを避けるために、プリフェッチ・サイズを大きくし過ぎないでください。理想的なサイズは、エクステント・サイズ、各コンテナーの物理ディスクの数 (RAID 装置を使用している場合)、および表スペース・コンテナーの数を掛け合わせた値です。エクステント・サイズはかなり小さくする必要があり、適した値は 8 から 32 ページの範囲です。
- コンテナーが、別々の物理ドライブに常駐すること。
- 並列処理の多重度に一貫性を持たせるために、すべてのコンテナーが同じサイズであること。

  他のコンテナーより小さいコンテナーが 1 つまたは複数ある場合には、並列プリフェッチが最適化される可能性が少なくなります。以下の例を検討してください。

  – 他より小さいコンテナーが満杯になると、それから先のデータは残りの、他のコンテナーに保管されることになるので、コンテナー間の均衡がとれなくなります。コンテナーが不均衡になると、データのプリフェッチができるコンテナーの数が、コンテナーの合計数より少なくなるので、並列プリフェッチのパフォーマンスが低下します。
  – 他より小さいコンテナーが後で追加されたときにデータの均衡が再度図られた場合には、小さなコンテナーには他のコンテナーより少ないデータが入れられることになります。他のコンテナーに比べて少量のデータでは、並列プリフェッチは最適化されません。
  – ある 1 つのコンテナーが他より大きいときに、他のコンテナーがすべて満杯になったとすると、そのコンテナーが追加のデータを保管できる唯一のコンテナーになります。したがって、データベース・マネージャーは、この追加データにアクセスする際には、並列プリフェッチを使用することができなくなります。
- パーティション内並列処理を使用する場合には、十分な入出力容量が用意されていること。 SMP マシンの場合、パーティション内並列処理を使用し、複数のプロセッサーで照会を実行することによって、経過時間を短縮することができま

す。各プロセッサーを十分に活用するには、十分な入出力容量が必要です。通常、十分な入出力容量を提供するためには追加の物理ドライブが必要になります。

プリフェッチを高速で行い、入出力容量を効率的に使用するためには、プリフェッチ・サイズをできるだけ大きくする必要があります。

必要となる物理ドライブの数は、ドライブと入出力バスの速度と容量、およびプロセッサーの速度によって変わります。

関連概念:

- 263 ページの『プリフェッチと並列処理のための入出力サーバー構成』
- 264 ページの『並列入出力を使用したプリフェッチの図』
- 268 ページの『ソート・パフォーマンスのガイドライン』

ソート・パフォーマンスのガイドライン

照会において、ソートされたまたはグループ化された結果が必要になることがよくあるので、ソートは頻繁に必要になります。また、ソート・ヒープ領域を正しく構成することも、照会で高いパフォーマンスを得るためにきわめて重要です。次のような場合に、ソートが必要になります。

- 要求された配列を満たす索引がない (たとえば、 ORDER BY 文節を使用する SELECT ステートメント)。
- 索引はあるが、索引を使用するよりもソートを行う方が効率が良い場合。
- 索引が作成される場合。
- 索引がドロップされることにより、索引ページ番号がソートされます。

ソートには 2 つのステップが関係しています。

1. ソート・フェーズ

   ソートには、*オーバーフロー* と、*オーバーフローしない* の 2 種類があります。ソートされるデータがソート・ヒープ (ソートの実行ごとに割り当てられるメモリーのブロック) に全く収まらない場合は、一時表にオーバーフローします。オーバーフローしないソートの方が常に、オーバーフローするソートよりも効率良く実行されます。
2. ソート・フェーズの結果を戻すフェーズ

   戻されるソートには、*パイプ* ・ソートと、*非パイプ* ・ソートの 2 種類があります。データの最終ソート結果リストを保管する一時表を使わなくても、ソートされた情報を直接戻すことができる場合は、そのソートはパイプ・ソートになります。ソートされた情報を戻すのに一時表が必要な場合には、そのソートは非パイプ・ソートになります。パイプ・ソートは、非パイプ・ソートよりも常に効率よく実行されます。

ソートに影響を与えるエレメント

ソート・パフォーマンスに影響を与える次のようなエレメントがあります。

- 次のデータベース構成パラメーターの設定値:

  - ソート・ヒープ・サイズ (*sortheap*)。これは、ソートごとに使用するメモリー量を指定します。
  - ソート・ヒープしきい値 (*sheapthres*)、および共用ソートのソート・ヒープしきい値 (*sheapthres_shr*)。これは、すべてのソートのインスタンス全体で使用可能な、ソート用の合計メモリー量の制御を行います。
- 大量のソートに関係するステートメント
- 不要なソートを避けるのに役立つ索引の脱落
- ソートを最小化しないアプリケーション論理
- 並列ソート。ソートのパフォーマンスを向上させるが、ステートメントがパーティション内並列処理を使用する場合にのみ行われます。

一般に、インスタンス (*sheapthres*) で使用可能なソート・メモリーの全体量は、過度のページングを引き起こさない限り、できるだけ大きくする必要があります。ソートはソート・メモリー内で全体的に実行されますが、これは過度のページ・スワッピングを引き起こす可能性があります。このような場合、ラージ・ソート・ヒープの利点が生かされません。そのため、ソート構成パラメーターを調整するときには、オペレーティング・システム・モニターを用いて、システム・ページングの変更を追跡してください。

また、パイプ・ソートでは、アプリケーションがそのソートに関連したカーソルをクローズするまで、ソート・ヒープが解放されないことに注意してください。パイプ・ソートはカーソルがクローズされるまでメモリーを消費し続けることができます。

注: DB2® 部分キーのバイナリー・ソート技法に、非整数のデータ・タイプ・キーを組み込むように改善がなされたので、長いキーをソートする時には追加のメモリーが必要です。ソートで長いキーが使用される場合、 *sortheap* 構成パラメーターを増やしてください。

ソート・パフォーマンスの管理技法

ソートが重大なパフォーマンス問題となっている特定のアプリケーションおよびステートメントを識別します。

- イベント・モニターをアプリケーションおよびステートメントのレベルでセットアップして、ソート合計時間が最も長いアプリケーションを識別します。
- そのようなアプリケーションのそれぞれにおいて、*ソート合計時間* が最も長いステートメントを見つけます。
- Visual Explain などのツールを用いて、そのようなステートメントを調整します。
- 適切な索引があるかどうかを確認します。 Visual Explain を用いて、指定されたステートメントのすべてのソート操作を識別することができます。次に、ステートメントがアクセスする表ごとに、適切な索引があるかどうかを調べます。

注: Explain 表を検索して、ソート操作が行われている照会を識別することができます。

データベース・システム・モニターとベンチマーク技法を使用すると、構成パラメーター *sortheap* および *sheapthres* を設定するのに役立ちます。データベース・マネージャーとそのデータベースごとに、次のことを実行してください。

- 代表的なワークロードを設定し稼働します。
- 適用するデータベースごとに、ベンチマーク・ワークロード期間中の次のパフォーマンス変数の平均値を収集します。
  - 使用中のソート合計ヒープ
  - アクティブなソート
- データベースごとに、 *sortheap* を *使用中のソート合計ヒープ* の平均値に設定します。
- *sheapthres* を設定します。適切なサイズを見積もるには、次のようにします。
  1. インスタンス中で最大の *sortheap* 値を持つ、データベースを判別します。
  2. このデータベースのソートヒープの平均サイズを判別します。

     判別するのが困難である場合、最大のソートヒープの 80% を使用します。
  3. アクティブなソートの平均数に、算出したソートヒープの平均サイズを掛けた値に *sheapthres* を設定します。

     これは、初期設定としてお勧めします。次に、ベンチマーク技法を使用して、この値をより良いものにすることができます。

関連資料:

- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 408 ページの『sheapthres - ソート・ヒープしきい値』
- 396 ページの『sheapthres_shr - 共用ソートのソート・ヒープのしきい値』

表管理

このセクションでは、表を管理してパフォーマンスを向上させる方法について説明します。

表の再編成

表データに多くの変更を加えた後、論理上順次データは、順序どおりになっていない物理データ・ページ上にある場合があるので、データベース・マネージャーはデータにアクセスするのに追加の読み取り操作を実行する必要があります。多数の行を削除した場合にも、追加の読み取り操作が必要になります。このような場合、表を再編成することにより、索引と一致させ、スペースをレクラメーション処理することを検討することができます。データベース表だけでなく、システム・カタログ表も再編成できます。

注: 通常、表の再編成は統計の実行より時間がかかるため、 RUNSTATS を実行してデータの現在の統計をリフレッシュし、アプリケーションを再バインドすることもできます。統計のリフレッシュでパフォーマンスが向上しない場合、再編成が役に立ちます。 REORG TABLE ユーティリティーのオプションと動作に関する詳細は、「コマンド・リファレンス」を参照してください。

表の再編成が必要であることを示す、次の要因を検討してください。

- 照会によってアクセスされる表における大量の挿入、更新、および削除アクティビティー
- クラスター率の高い索引を使用する照会のパフォーマンスにおける顕著な変化
- RUNSTATS を実行して統計情報をリフレッシュしてもパフォーマンスが向上しない
- REORGCHK コマンドが、表を再編成する必要があることを示している
- 照会パフォーマンスの低下を向上させた場合のコストと、表を再編成した場合のコスト (REORG ユーティリティーが再編成の完了まで表をロッキングすることによって生じる CPU 時間、経過時間、および並行性の低下を含む) の間のトレードオフ。

表の再編成の必要性を少なくする

表の再編成の必要性を少なくするには、表を作成した後に以下のタスクを行ってください。
- 表を変更して PCTFREE を追加します。
- 索引上の PCTFREE を使ってクラスタリング索引を作成します。
- データをソートします。
- データをロードします。

これらのタスクを実行した後、表およびそのクラスタリング索引と、表上の PCTFREE の設定値は、元のソート順序を保存するのに役立ちます。表ページに十分なスペースがある場合、正しいページに新しいデータを挿入することができます。これにより、索引のクラスタリング特性が保持されます。データがさらに挿入され、表のページがいっぱいになると、表の最後にレコードが追加され、上のクラスター特性は徐々に失われます。

クラスタリング索引を作成した後に REORG TABLE またはソートを実行し、LOAD を実行すると、その索引はデータの特定の順序を保持しようするので、RUNSTATS ユーティリティーによって収集された CLUSTERRATIO または CLUSTERFACTOR 統計は改善されます。

注: マルチディメンション・クラスタリング (MDC) 表を作成すると、表を再編成する必要性は少なくなります。 MDC 表のクラスタリングは、 CREATE TABLE ステートメントの ORGANIZE BY DIMENSIONS 文節の引き数として指定された列に保守されます。ただし、未使用ブロックが多すぎる、またはブロックをコンパクトにする必要があると判断される場合は、 REORGCHK は MDC 表の再編成を勧める場合があります。

関連概念:
- 287 ページの『索引の再編成』
- 290 ページの『DMS 装置に関する考慮事項』
- 16 ページの『SMS 表スペース』
- 17 ページの『DMS 表スペース』
- 20 ページの『標準の表における表および索引の管理』
- 「システム・モニター *ガイドおよびリファレンス*」の『スナップショット・モニター』

- 23 ページの『MDC 表のための表および索引管理』

関連タスク:

- 272 ページの『表の再編成時の決定』
- 275 ページの『表を再編成する方式の選択』

表の再編成時の決定

挿入、削除、および更新などの通常のデータベース・アクティビティーが繰り返されると、やがて表および索引内のデータ編成に影響が及び、データベース・パフォーマンスに悪影響を与える可能性があります。 RUNSTATS ユーティリティーは、データベース表の編成についての統計情報を収集し、これによって表サイズとデータ分散、索引のクラスター比率、索引内のリーフ・ページ数、元のページからオーバーフローする表の行数、および表内の入力されているページ数と空のページ数に関する統計情報も収集できます。またプリフェッチの効率に関する情報を収集するのにも、RUNSTATS を使用できます。

RUNSTATS が収集した情報は、システム・カタログ表に保管されます。この情報は、表および索引を再編成する時期、および必要な再編成のタイプを判別するのに役立ちます。

データベースの編成に関する情報をキャプチャーするためにあるポイント・イン・タイムで RUNSTATS を使用するだけでなく、RUNSTATS を定期的に実行して、全体的な傾向を識別するのは良いことです。こうした傾向はデータベース・パフォーマンスの変化に関連する場合もあります。

注: REORGCHK コマンドはデータ編成に関する統計情報も戻すので、特定の表で再編成が必要かどうかを判別するのに役立ちます。しかし、カタログ統計表に対する特定の照会を、決まったインターバルまたは特定の回数実行することによって提供されるパフォーマンス履歴を活用すれば、パフォーマンスを大幅に向上させる可能性のある傾向を見つけることができます。

手順:

表の再編成が必要かどうかを判別するには、カタログ統計表を照会して、以下の統計をモニターします。

1. 行のオーバーフロー

   SYSSTAT.TABLES 表の OVERFLOW 列を照会して、オーバーフロー値をモニターします。この列内の値は、元のページに収まらない行の数を示しています。行のデータのオーバーフローは、 VARCHAR 列が初期値より長い値に更新された場合に起きることがあります。この場合、行の元の位置にはポインターが保持され、実際の値はポインターが示す別の場所に保管されます。これはパフォーマンスを影響を与える可能性があります。データベース・マネージャーはポインターに従って行の内容を検出しなければならないためです。この 2 つのステップから成るプロセスにより、処理時間が長くなり、必要な入出力の回数も多くなる可能性があります。

表データを再編成すると行のオーバーフローを除去できるので、オーバーフロー行の数が増えるにつれて、表データの再編成が効果的である可能性が高くなります。

2. 統計のフェッチ

   SYSCAT.INDEXES および SYSSTAT.INDEXES カタログ統計表にある、以下の 3 つの列を照会して、索引の順序で表にアクセスする場合のプリフェッチの効果性について判別します。これらの統計では、基本表に対するプリフェッチャーの平均パフォーマンスが示されています。

   - AVERAGE_SEQUENCE_FETCH_PAGES 列には、表に順序どおりにアクセスできるページの平均数が保管されています。順序どおりにアクセスできるページは、プリフェッチが可能です。この値が小さいということは、プリフェッチャーが本来の効果性を発揮できないということです。これは、表スペースの PREFETCHSIZE 設定に指定されたページ数全体を読み取ることができないためです。数が大きい場合、プリフェッチャーの実行は効果的です。クラスター索引および表の場合、この値は NPAGES の値 (行が入っているページの数) に近づくはずです。
   - AVERAGE_RANDOM_FETCH_PAGES 列は、索引を使用して表の行をフェッチするときの、順次ページ・アクセスの間のランダム表ページの平均数を保管します。ほとんどのページが順次の場合、プリフェッチャーは数の少ないランダムのページは無視して、構成されたプリフェッチ・サイズまでプリフェッチを継続します。表のまとまりがなくなるにつれ、ランダム・フェッチ・ページ数は増加します。通常このようにまとまりがなくなるのは、表の最後かオーバーフロー・ページのいずれかで、適切でない順序で挿入が行われることにより発生します。これによって行われるフェッチでは、索引を使用して何らかの範囲の値にアクセスする際に、照会パフォーマンスが低下します。
   - AVERAGE_SEQUENCE_FETCH_GAP 列は、索引を使用してフェッチする際の、表ページ・シーケンス間の平均ギャップを保管します。各ギャップは索引リーフ・ページのスキャンにより検出され、一連の表ページの間でランダムにフェッチしなければならない表ページの平均数を表します。これらは、多くのページがランダムにアクセスされて、プリフェッチャーに割り込みが行われたときに発生します。数が大きい場合、表はまとまりがなく、索引に対するクラスター化の多重度が低いことを示しています。

3. 削除対象としてマークが付けられているものの、除去されていない RID を含む、索引リーフ・ページの数

   通常、タイプ 2 では、 RID に削除対象としてマークが付けられた時点で、その RID が物理的に削除されることはありません。これは、有用なスペースがそれらの論理的には削除済みの RID で占められる可能性があるということです。すべての RID が削除済みとしてマークされているリーフ・ページ数を検索するには、 SYSCAT.INDEXES および SYSSTAT.INDEXES 統計表の NUM_EMPTY_LEAFS 列を照会します。一部の RID に、削除対象としてマークが付けられているリーフ・ページに関しては、論理的には削除されている RID の合計数が NUMRIDS_DELETED 列に保管されます。

   この情報を使用して、 CLEANUP ALL オプションを指定して REORG INDEXES を実行することにより、レクラメーション処理できるスペースの量の見積もりを行ってください。 RID に削除対象のマークが付けられている、ペー

ジ内のスペースだけをレクラメーション処理するには、 CLEANUP ONLY PAGES オプションを指定して REORG INDEXES を実行します。

4. 索引のクラスター率およびクラスター係数の統計

   クラスター率統計は、 SYSTCAT.INDEXES カタログ表の CLUSTERRATIO 列に保管されます。この値 (0 から 100 まで) は、索引のデータ・クラスタリングの程度を表わします。 DETAILED 索引統計を収集する場合は、代わりにより細かいクラスター率統計 (0 から 1) が CLUSTERFACTOR 列に保管され、CLUSTERRATIO の値は -1 になります。これらのクラスタリング統計のいずれか 1 つだけを、 SYSCAT.INDEXES カタログ表に記録できます。CLUSTERFACTOR 値と CLUSTERRATIO 値を比較するには、CLUSTERFACTOR に 100 を乗算してパーセンテージを得ます。

   注: 一般に、高度なクラスタリングが可能なのは、1 つの表の中で 1 つの索引だけです。

   索引専用アクセスではない索引スキャンは、クラスター率が高い方がよくなることがあります。 クラスター率が低いと、この種のスキャンでは各データ・ページの最初のアクセスの後、次にアクセスが行われるまでバッファー・プール内にページが残っている可能性が少なくなるため入出力が増えます。バッファー・サイズを大きくすると、クラスター化されていない索引のパフォーマンスが向上することがあります。

   特定の索引で表データのクラスタリングが最初に行われ、クラスタリングの統計情報から、この索引に関するデータのクラスタリングが不十分であることがわかった場合、表を再編成して、データのクラスタリングを再び行うこともできます。

5. リーフ・ページの数

   索引が使用するリーフ・ページの数を調べるには、SYSCAT.INDEXES 表の NLEAF 列を照会します。その数から、索引の完全なスキャンに必要な索引ページ I/O の回数が分かります。

   理想的なのは、索引の占めるスペースが可能な限り最小で、索引スキャンに必要な I/O が少なくなることです。ランダム更新アクティビティーによりページ分割が行われ、索引のサイズが大きくなることがあります。表の再編成中に索引を再作成すると、各索引を最小のスペースで作成することができます。

   注: デフォルトでは、索引の作成を行う場合、各索引ページあたり 10% のフリー・スペースが残っています。フリー・スペースの量を増やすには、索引の作成時に PCTFREE パラメーターを指定します。索引を再編成するたびに PCTFREE 値が使用されます。フリー・スペースが 10% を超えている場合、追加のスペースによって追加の索引の挿入に対応するので、索引の再作成の頻度が減ります。

6. ファイル・ページの比較

   表の空ページの数を計算するには、 SYSCAT.TABLES の FPAGES および NPAGES 列を照会し、 FPAGES の数から NPAGES の数を引きます。 FPAGES

列には、使用中のページの合計数が保管されています。 NPAGES 列には、行を含んでいるページの数が保管されています。行の範囲が全部削除されると、空ページが生じることがあります。

空ページの数が多いほど、表を再編成する必要が大きくなります。表を再編成すると、空ページをレクラメーションして、表に使用されるスペースの量を圧縮できます。表スキャンで空ページがバッファー・プール中に読み取られるので、未使用ページのレクラメーション処理により、表スキャンのパフォーマンスが向上します。

関連概念:

- 121 ページの『カタログ統計の表』
- 270 ページの『表の再編成』
- 287 ページの『索引の再編成』

関連タスク:

- 112 ページの『カタログ統計の収集』

表を再編成する方式の選択

DB2® では、従来の方式とインプレース方式の 2 つの方法で表を再編成できます。通常は従来の表再編成の方が高速ですが、再編成中にアプリケーションに表への書き込みアクセス権限がない場合にのみ、この方式を使用すべきです。この制約事項を満たさない環境では、インプレース再編成の方が遅いものの、通常のデータ・アクセスが続行される間にバックグラウンドで実行されるという利点があります。それぞれの方式の特徴を考慮して、どちらが環境により適しているかを判別してください。

手順:

表再編成の方式を選択するために、以下の方式の特徴を考慮します。

- **従来の表再編成**

  この方式による表再編成は最も高速です (とくに、 LOB データや LONG データを再編成する必要のない場合)。しかも、表の再編成後に、索引は完全な順序で再作成されます。読み取り専用アプリケーションは、再編成の最終段階を除いて元の表にアクセスできます。再編成の最終段階では、永続する表が表のシャドー・コピーに置き換わり、索引が再作成されます。

  他方、以下のような欠点もあります。

  – 大きなスペースを必要とする

    従来の表再編成では表のシャドー・コピーが作成されるため、元の表の 2 倍のスペースが必要です。再編成後の表が元の表よりも大きい場合には、元の表の 2 倍を超えるスペースが必要となる可能性があります。

    その表の表スペースが十分に大きくない場合、シャドー・コピーを TEMPORARY 表スペースに作成することが可能です。ただし、同じ DMS 表スペースの方が置換段階がより高速に実行されます。 SMS 表スペース内の表は、シャドー・コピーを常に一時スペースに保管しなければなりません。

– 表へのアクセスが制限される

  読み取り専用アクセスでさえ制限されます (再編成の第 1 段階でのみ、読み取りが許可されます)。
– 処理を途中から再開できない

  再編成が途中で失敗した場合、失敗したノードで最初から再開する必要があります。
– 呼び出し元のアプリケーション・コントローラー内で実行される

  再編成を停止できるのは、そのアプリケーションのみ、あるいは、処理の停止方法を熟知していて、アプリケーションに対して FORCE コマンドを実行する権限を持っているユーザーのみです。

推奨: 保守ウィンドウで表を再編成できる場合には、この方式を使用してください。

- **インプレース表再編成**

  インプレース方式はより遅く、データの完全な順序も保証されませんが、再編成中にアプリケーションが表にアクセスできます。さらに、適切な権限を持つすべてのユーザーは、スキーマおよびテーブル名を使用することにより、インプレース表再編成を一時停止して後で再開することができます。

  注: インプレース表再編成は、タイプ 2 索引を含み、索引が拡張されていない表に対してのみ実行することができます。

  以下のようなトレードオフがあります。
  – 索引の再編成が不完全である

    索引のフラグメント化を削減し、索引オブジェクト・スペースをレクラメーション処理するために、後で索引を再編成する必要があるかもしれません。
  – 完了までに長い時間がかかる

    必要に応じて、インプレース再編成は同時に実行されるアプリケーションに合わせて遅延します。つまり、長時間実行されるステートメントや、長時間実行されるアプリケーションにおける RR リーダーまたは RS リーダーが、再編成の進行を遅らせる可能性があります。インプレース再編成は、小さなトランザクションが数多く発生する OLTP 環境では、より高速に動作するかもしれません。
  – より大きなログ・スペースが必要

    予期しない障害が起きた場合のリカバリーを可能にするために、インプレース表再編成はアクティビティーをログに記録します。このため、従来の再編成よりも大きなログ・スペースを必要とします。

    インプレース再編成が、再編成後の表の何倍にも及ぶログ・スペースを必要とすることもあります。必要なスペースは、移動される行数、および表の索引の数やサイズに応じて異なります。

推奨: 最小保守ウィンドウで 24x7 操作を行う場合、インプレース表再編成を使用してください。

これらの表再編成方式の実行方法について、詳しくは REORG TABLE 構文の説明を参照してください。

表再編成の進行状況のモニター

表再編成の現在の進行状況は、データベース・アクティビティーの履歴ファイルに書き込まれます。履歴ファイルには、それぞれの再編成イベントに関するレコードが含まれます。このファイルを表示するには、再編成対象の表が格納されているデータベースに対して `db2 list history` コマンドを実行してください。

このほか、表スナップショットを使用して、表再編成の進行状況をモニターすることもできます。表再編成モニター・データは、データベース・モニター表スイッチの設定値にかかわらず記録されます。

エラーが発生した場合、SQLCA ダンプが履歴ファイルに書き込まれます。インプレース表再編成の場合、状況は「PAUSED (一時停止)」と記録されます。

関連概念:

- 270 ページの『表の再編成』
- 287 ページの『索引の再編成』

関連タスク:

- 272 ページの『表の再編成時の決定』

索引管理

次のセクションでは、パフォーマンスを向上させるために行う索引の再編成について説明します。

索引の利点と欠点

オプティマイザーは表データにアクセスするために索引を使用するかどうかを決定しますが (次に挙げる場合を除き)、ユーザーはどの索引がパフォーマンスを向上させるかを判別し、それらの索引を作成しなければなりません。例外は、マルチディメンション・クラスタリング (MDC) 表の作成時に指定する各ディメンションに対して自動的に作成される、ディメンション・ブロック索引および複合ブロック索引です。

また、次のような場合には RUNSTATS ユーティリティーを実行して、索引に関する新規の統計を収集する必要もあります。

- 索引を作成した後
- プリフェッチ・サイズを変更した後

また、RUNSTATS ユーティリティーは、定期的なインターバルで実行し、統計を現行のものに保持する必要があります。索引についての最新の統計がないと、オプティマイザーは照会に対する最適のデータ・アクセス・プランを判別することができません。

注: 特定のパッケージで索引を使用するかどうかを決めるには、 SQL Explain 機能を使用します。索引を計画するには、コントロール・センターから設計アドバイザーを使用するか db2advis ツールを使用して、 1 つ以上の SQL ステートメントによって使用される可能性のある索引についてのアドバイスを入手してください。

索引を使用しない場合と比べての索引の利点

表に索引が存在しない場合、データベース照会において参照される各表ごとに、表スキャンを実行しなければなりません。表スキャンでは、各表行が順次アクセスされる必要があるため、表が大きいほど表スキャンにかかる時間も長くなります。表内のほとんどの行を必要とするような複雑な照会では表スキャンのほうが効率的かもしれませんが、いくつかの表行のみを戻す照会では、索引スキャンのほうが表行に効率的にアクセスできます。

オプティマイザーは、索引列が SELECT ステートメント内で参照され、表スキャンより索引スキャンのほうが速いと見積もった場合、索引スキャンを選択します。特に表が大きくなるにつれ、索引ファイルは一般にファイル全体に比べて小さくなり、読み取りのための時間が少なくなります。さらに、索引全体をスキャンすることが必要になることはまずありません。索引に述部を適用すると、データ・ページから読み取られる行数が減ります。

出力の配列要件が索引の列と一致する場合、列の順序で索引をスキャンすることにより、ソートをしなくても正しい順序で行を検索できます。

各索引項目は、検索キー値と、その値が入った行を指すポインターで構成されています。 CREATE INDEX ステートメントの ALLOW REVERSE SCANS パラメーターを指定する場合、その値は降順と降順の両方で検索することができます。したがって、適切な述部を使用して、検索を一まとめにすることができます。索引を使ってオーダー順序で行を入手することにより、データベース・マネージャーが行を表から読み取ってからその行をソートする必要がなくなります。

検索キーの値および行ポインターに加えて、索引には列を含めることができます。これらの列は、索引付けされた行内の索引付けされていない列です。そのような列があると、オプティマイザーは表そのものにアクセスしなくても必要な情報を索引のみから入手することができます。

注: 照会されている表に索引が存在していても、順序付けられた結果セットが得られるとは限りません。 ORDER BY 文節を使用することによってのみ、結果セットの順序付けが確実になります。

索引はアクセス時間を大幅に短縮することができますが、同時にパフォーマンスに悪い影響を与えることもあります。索引を作成する前に、複数の索引を作成することが、ディスク・スペースや処理時間に対してどんな影響を与えるかを考慮してください。

- 各索引には、ストレージおよびディスク・スペースが必要になります。正確な量は、表のサイズと索引に含まれる列の数およびサイズによって異なります。
- 表に対して実行される INSERT または DELETE の各操作では、その表の各索引を更新することもさらに必要になります。さらに、索引キーの値を変更する UPDATE 操作についても同じことが言えます。

- LOAD ユーティリティーでは既存の索引が再作成されたり、既存の索引に索引が追加されます。

  索引が作成された時に使用される索引 PCTFREE をオーバーライドするために、LOAD コマンドで indexfreespace MODIFIED BY パラメーターを指定できます。
- 各索引は、オプティマイザーが考慮する照会の代替アクセス・パスを潜在的に追加するものとなり、そのため照会のコンパイル時間が長くなります。

索引は、アプリケーション・プログラムの要件に合わせて慎重に選択してください。

関連概念:

- 「*管理ガイド: プランニング*」の『索引のスペース所要量』
- 279 ページの『索引の計画のヒント』
- 282 ページの『索引のパフォーマンスのヒント』
- 227 ページの『設計アドバイザー』
- 270 ページの『表の再編成』
- 287 ページの『索引の再編成』
- 20 ページの『標準の表における表および索引の管理』
- 23 ページの『MDC 表のための表および索引管理』
- 285 ページの『索引のクリーン・アップおよび保守』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『索引の作成』
- 112 ページの『カタログ統計の収集』
- 115 ページの『詳細索引統計の収集』

索引の計画のヒント

作成する索引は、アクセスするデータおよび照会に合わせたものにする必要があります。

コントロール・センターから設計アドバイザーを使用するか、 db2advis ツールを使用して、特定の照会またはワークロードを定義する照会の集合に最適な索引を調べてください。このツールは、パフォーマンス向上の機能を持つ索引 (ユニーク索引から継承された INCLUDE 列、および ALLOW REVERSE SCANS 索引など) を推奨します。

さまざまな目的の索引を作成する方法を判別するのに、以下の指針が役立つでしょう。

- あるソートを回避するには、可能であれば CREATE UNIQUE INDEX ソートを使用して、主キーおよびユニーク・キーを定義します。
- データ検索を向上させるには、 INCLUDE 列をユニーク索引に追加します。次のような列の場合に、追加するとよいでしょう。
  - 頻繁にアクセスされるので、索引のみのアクセスによって益が生じる場合
  - 索引スキャンの範囲を限定するために必要でない場合
  - 索引キーの順序または固有性に影響を与えない場合

- 小さな表に効率的にアクセスするには、索引を使用して、データ・ページが 2、3 ページより多い表への頻繁な照会を最適化します。このことは、SYSCAT.TABLES カタログ・ビューの NPAGES 列に記録されます。次のことを行う必要があります。
  - 表を結合するときに使用するすべての列に索引を作成します。
  - 通常、特定の値を検索する対象となる列に索引を作成します。
- 効率的に検索するには、キーの配列を昇順にするか降順にするかを、最も頻繁に使用される順序に基づいて決定します。 CREATE INDEX ステートメントの ALLOW REVERSE SCANS パラメーターを指定すれば逆方向にも値を検索できますが、指定された索引の順序のスキャンのほうが、逆方向のスキャンよりも少し良くなります。
- 索引の保守コストおよびスペースを少なくするには、以下のようにします。
  - 列上の他の索引キーの部分キーとなるような索引は作成しないようにします。たとえば、列 a、b、および c に対して 1 つの索引がある場合、列 a と b についての 2 番目の索引は通常有用ではありません。
  - 根拠もなくすべての列に索引を作成しない。必要のない索引はスペースを使用するだけでなく、準備時間を増やす原因ともなります。このことは、複合照会、ダイナミック・プログラミング結合列挙による最適化クラスを使用する場合、特に重要です。

    表に対して定義する索引の数に関する、以下の一般規則を使用してください。この数は、データベースの主な用途に基づいています。
    - オンライン・トランザクション処理 (OLTP) 環境では、 1 つまたは 2 つの索引だけを作成してください。
    - 読み取り専用照会環境では、 5 以上の索引を作成することも可能です。
    - 混合照会および OLTP 環境では、 2 ～ 5 の索引を作成できます。
- 親表での削除および更新操作のパフォーマンスを向上させるには、外部キーについて索引を作成します。
- IMMEDIATE MQT と INCREMENTAL MQT が関係する DELETE と UPDATE 操作のパフォーマンスを向上させるには、 MQT の暗黙のユニーク・キー (MQT の定義の GROUP BY 文節内の列) についてユニーク索引を作成します。
- ソート操作を速くするには、データのソート時に頻繁に使用される列について索引を作成します。
- 複数列索引で結合のパフォーマンスを向上させるには、最初のキー列に関して複数の選択の余地がある場合は、「=」(equijoin) 述部に指定されることの最も多い列を使用するか、最初のキーとしての個別の値が最も多い列を使用します。
- 新しく挿入された行がその索引に応じてクラスタリングされ続けるようにし、またページ分割を避けるには、クラスタリング索引を定義します。クラスタリング索引は、表を再編成する必要性を大幅に少なくします。

  表を定義するときには PCTFREE キーワードを使用して、ページに適切に挿入ができるようにどれだけのフリー・スペースをページに残すかを指定してください。 LOAD コマンドの pagefreespace MODIFIED BY 文節を指定することもできます。
- オンライン索引デフラグを使用可能にするには、索引の作成時に MINPCTUSED オプションを使用します。 MINPCTUSED は、索引リーフ・ページ上で使用される最小スペースの限界値を指定し、オンライン索引デフラグを使用可能にしま

す。これにより、索引ページから物理的にキーを除去する削除の場合に、パフォーマンスを低下させる再編成の必要性を低くできる可能性があります。

以下の場合に、索引の作成を考慮してください。

- 最も頻繁に処理される照会およびトランザクションの WHERE 文節の中で使用される列には、索引を作成してください。

  次のような WHERE 文節の場合、

  ```
  WHERE WORKDEPT='A01' OR WORKDEPT='E21'
  ```

  一般に、WORKDEPT 列が多くの重複値を含んでいる場合を除き、 WORKDEPT に対する索引があれば便利です。

- 必要によって要求される順序の中で行を並べ替えるには、 1 つ以上の列に索引を作成してください。 ORDER BY 文節においてだけでなく、DISTINCT 文節や GROUP BY 文節など、他の機能でも並べ替えが必要になります。

  以下の例では、DISTINCT 文節を使用しています。

  ```
  SELECT DISTINCT WORKDEPT
    FROM EMPLOYEE
  ```

  データベース・マネージャーでは、 WORKDEPT に対して昇順または降順に定義された索引を使用することによって、重複値を削除することができます。以下の GROUP BY 文節の例のようにして、この同じ索引を使用して値をグループ化することもできます。

  ```
  SELECT WORKDEPT, AVERAGE(SALARY)
    FROM EMPLOYEE
  GROUP BY WORKDEPT
  ```

- ステートメントで参照される名前を持つコンパウンド・キーについては、索引を作成します。この方法で索引を作成すると、データを索引のみから検索できるようになり、表にアクセスするよりも効率的です。

  たとえば、次の SQL ステートメントを考察してみましょう。

  ```
  SELECT LASTNAME
    FROM EMPLOYEE
    WHERE WORKDEPT IN ('A00','D11','D21')
  ```

  EMPLOYEE 表の WORKDEPT 列と LASTNAME 列に索引が定義されている場合、表全体をスキャンするよりも索引をスキャンする方が、ステートメントをより効率的に処理することができます。述部は WORKDEPT についてであるため、この列は索引の最初の列でなければなりません。

- INCLUDE 列について索引を作成すると、表での索引の使用が改善されます。前の例を使用し、次のようにしてユニーク索引を定義できます。

  ```
  CREATE UNIQUE INDEX x ON employee (workdept) INCLUDE (lastname)
  ```

  lastname を索引キーの一部としてではなく、組み込み列として指定することは、lastname が索引のリーフ・ページでのみ保管されるということを意味します。

関連概念:

- 277 ページの『索引の利点と欠点』
- 282 ページの『索引のパフォーマンスのヒント』

- 227 ページの『設計アドバイザー』
- 287 ページの『索引の再編成』
- 289 ページの『オンライン索引のデフラグ』
- 「*管理ガイド: プランニング*」の『マルチディメンション・クラスタリング (MDC) 表の作成、配置、および使用』

索引のパフォーマンスのヒント

索引の使用および管理について、以下の提案を考慮してください。

- **索引の作成時および再編成時に並列処理を指定する**

  SMP マシンがホストしている大きな表に索引を作成および再編成する場合、*intra_parallel* を YES (1) か SYSTEM (-1) に設定して、並列パフォーマンスの機能を利用することを考慮してください。

  複数のプロセッサーを使用して、データのスキャンやソートを行うことができます。

- **大規模なユーティリティー・ヒープを指定する**

  他のユーザーまたはアプリケーションからの基礎表への書き込みアクセスは、CREATE INDEX および REORG INDEXES の両方でサポートされています。索引を作成または再編成するときに基礎表に対する大量の更新アクティビティーが発生することが予期される場合は、大規模なユーティリティー・ヒープを構成することを考慮してください。大規模なユーティリティー・ヒープは、キャッチアップ・フェーズでの索引作成または索引再編成を高速化します。作成中または再編成中の索引に対するすべての書き込みアクティビティーは、 DB2® ログおよび内部メモリー・バッファー・スペースに記録されます。内部メモリー・バッファー・スペースは、ユーティリティー・ヒープからオンデマンドで割り振られる特定のメモリー域で、作成中または再編成中の索引に加えられる変更を保管します。このメモリーの使用によって、キャッチアップ・フェーズを高速化できます。割り振られたメモリーは、作成または再編成操作が完了すると解放されます。作成中または再編成中の索引に加えられる変更のすべてまたは大半を入れられる十分なユーティリティー・ヒープがあるようにすれば、キャッチアップ・フェーズでのパフォーマンスにはかなりの向上がみられる可能性があります。

- **索引には別々の表スペースを指定する**

  索引は、表データとは別の表スペースに保管することができます。これにより、索引アクセス時の読取/書込ヘッドの移動が少なくなり、ディスク装置をさらに効率的に使用できるようになります。また、より高速の物理装置に索引表スペースを作成することもできます。さらに、索引表スペースを別々のバッファー・プールに割り当てることもできます。これにより、索引ページは表データ・ページと競合することがないため、バッファーに長く保持される可能性があります。

  索引を別々の表スペースに置かない場合、データと索引ページの両方が同じエクステント・サイズとプリフェッチ数量を使用します。索引用に別の表スペースを使用する場合は、表スペースのすべての特性に対して異なる値を選択することができます。索引は一般に表より小さく、分散しているコンテナーも少ないため、

8 または 16 ページといったエクステント・サイズになるのが通常です。 SQL オプティマイザーは、アクセス・プランを選択するときに、表スペースに対する装置の速度を考慮します。

- **クラスタリングの多重度を確認する**

  SQL ステートメントが ORDER BY、GROUP BY、および DISTINCT などの順序付けを必要としているときに、たとえ索引がその順序付けを満たしているとしても、以下の場合にはオプティマイザーが索引を選択しない可能性があります。
  - 索引クラスタリングの程度が低い場合。この情報については、SYSCAT.INDEXES の CLUSTERRATIO 列および CLUSTERFACTOR 列を調べてください。
  - 表が小さいので、表のスキャンや応答セットのソートをメモリーで行う方が低コストになるような場合
  - 表へのアクセスを競合する索引がある場合

  クラスタリング索引を作成した後、 REORG TABLE をクラシック・モードで実行し、完全に編成された索引を作成するようにします。表を再クラスター化するには、代わりにソートおよび LOAD を実行することもできます。ただし、一般に 1 つの索引では 1 つの表しかクラスター化できないことに注意してください。クラスタリング索引を作成した後で、追加の索引を作成してください。RUNSTATS ユーティリティーによって集められた CLUSTERRATIO または CLUSTERFACTOR 統計を向上させて、クラスタリング索引はデータの特定順序を維持しようと試みます。

  クラスタリング率を維持するためには、ロードまたは再編成を行う前に、表の変更時において適切な PCTFREE を指定してください。 PCTFREE によって指定される各ページのフリー・スペースによって、挿入のためのスペースができ、これらの挿入を適切にクラスター化することができます。表に PCTFREE を指定しない場合、再編成によってすべての余分のスペースが除去されます。

  注: クラスタリングは、レンジ・クラスター表を使用しない限り、現状では更新の間維持されません。つまり、クラスタリング索引のキー値が変更されるようにしてレコードを更新する場合、レコードはクラスタリング順序を維持するために新規ページに必ずしも移動されるとは限りません。クラスタリングを維持するために、UPDATE の代わりに、 DELETE とそれから INSERT を使用してください。

- **表および索引の統計を最新のものにしておく**

  新規索引を作成した後は、RUNSTATS ユーティリティーを使用して索引統計を収集してください。それらの統計を使うことによって、索引の使用によってアクセス・パフォーマンスが向上するかどうかをオプティマイザーが判断することが可能になります。

- **オンライン索引デフラグを使用可能にする**

  オンライン索引デフラグは、索引に対して、MINPCTUSED 文節がゼロより大きく設定されている場合に使用可能です。オンライン索引デフラグを使用すると、あるページのフリー・スペースが指定されたレベルと同じかそれより下回った場合にリーフ・ページをマージすることによって、索引を圧縮することができます。その間も索引は使用可能なままです。

- **必要に応じて索引を再編成する**

  索引から最高のパフォーマンスを得るには、索引を周期的に再編成することを考慮してください。表に対する更新によって索引ページのプリフェッチの効率が悪くなる可能性があるからです。

  索引を再編成するには、索引をドロップするか、再作成するか、または REORG ユーティリティーを使用してください。

  頻繁に再編成を行う必要を少なくするためには、索引の作成時に適切な PCTFREE を指定して、各索引のリーフ・ページに作成時と同じパーセンテージのフリー・スペースを残しておくようにします。将来の活動で、索引ページ分割をほとんど生じさせることなくレコードを索引に挿入できます。ページ分割が生じると、索引ページは隣接したものとならず、連続したものともなりません。そのために、索引ページのプリフェッチの効率が低下します。

  注: 索引の作成時に指定する PCTFREE は、索引が再編成されるときにも保持されます。

  索引をドロップして再作成する、または索引を再編成すると、ほぼ隣接し連続した一連の新しいページも作成され、索引ページのプリフェッチが向上します。時間とリソースの面ではコストがかかりますが、 REORG TABLE ユーティリティーでも確実にデータ・ページをクラスタリングすることができます。このようにしてクラスタリングを行うと、大量のデータ・ページにアクセスする索引スキャンを行う場合に大きな利点があります。

  対称マルチプロセッサー (SMP) 環境では、 *intra_parallel* データベース・マネージャー構成パラメーターが YES または ANY である場合、「クラシック」 REORG TABLE モード (シャドー表を使用して表の再編成を高速で行う) で、複数のプロセッサーを使用して索引の再作成ができます。

- **索引使用に関する EXPLAIN 情報を分析する**

  定期的に、使用頻度が最も高い照会に対して EXPLAIN を実行して、それぞれの索引が最低でも 1 回は使用されているかどうかを検査するようにしてください。どの照会でも使用されていない索引がある場合には、その索引のドロップを検討する必要があります。

  また、EXPLAIN 情報を使用すると、大きな表の表スキャンがネスト・ループ結合の内部表として処理されているかどうかを調べることもできます。そのように処理されている場合は、結合述部列の索引が欠落しているか、またはその索引が結合述部を適用するのに効果的でないと見なされているかのいずれかです。

- **サイズが大きく異なる表には Volatile 表を使用する**

  Volatile 表は、ランタイムのサイズが空であるものから非常に大きなものまでがある表です。カーディナリティーが大きく異なるこの種類の表に対しては、オプティマイザーは索引スキャンの代わりに表スキャンを行うアクセス・プランを生成する可能性があります。

  ALTER TABLE...VOLATILE ステートメントを使用して表を「揮発性」として宣言すると、オプティマイザーは Volatile 表に対して索引スキャンを使用できるよ

うになります。オプティマイザーは、以下の状況での統計に関係なく、表スキャンの代わりに索引スキャンを使用します。

– 参照される列がすべて索引内にある場合。
– 索引が索引スキャンで述部を適用できる場合。

表がタイプ表である場合、タイプ表階層のルート表で、 ALTER TABLE...VOLATILE ステートメントの使用のみがサポートされます。

関連概念:

- 277 ページの『索引の利点と欠点』
- 279 ページの『索引の計画のヒント』
- 26 ページの『索引の構造』
- 172 ページの『索引アクセスとクラスター率』
- 270 ページの『表の再編成』
- 287 ページの『索引の再編成』
- 20 ページの『標準の表における表および索引の管理』
- 289 ページの『オンライン索引のデフラグ』
- 23 ページの『MDC 表のための表および索引管理』
- 285 ページの『索引のクリーン・アップおよび保守』

関連資料:

- 525 ページの『intra_parallel - パーティション内並列処理機能の使用可能化』

索引のクリーン・アップおよび保守

索引の作成後、索引をコンパクトに編成しなければ、パフォーマンスが低下してしまいます。以下の提案を参考にして、索引を可能な限り小さく、効率的にしてください。

- オンライン索引デフラグを使用可能にする

  MINPCTUSED 文節を使って索引を作成します。必要に応じて、既存の索引をドロップして再作成してください。

- COMMIT を頻繁に実行します。 COMMIT の頻繁な実行が不可能な場合は、明示的に、またはロック・エスカレーションによって表に対する X ロックを取得します。

  削除済みとマークされた索引キーは、 COMMIT の後に表から物理的に除去されます。表に対する X ロックを使用すると、以下に説明するように、キーが削除済みとマークされた時点で物理的に除去されます。

- REORGCHK を使用すると、索引や表 (またはその両方) をいつ再編成すべきか、また CLEANUP ONLY オプションを使っていつ REORG INDEXES を実行すべきかを判別するのに役立ちます。

  再編成中に索引への読み取りおよび書き込みアクセスを可能にするには、ALLOW WRITE ACCESS オプションを使って REORG INDEXES を実行します。

注: DB2® バージョン 8.1 以降では、すべての新しい索引はタイプ 2 索引として作成されます。唯一の例外は、すでにタイプ 1 索引が存在する表に索引を追加する場合です。この場合、新しい索引もまたタイプ 1 索引になります。表にどのタイプの索引が存在するかを判別するには、 INSPECT コマンドを実行してください。タイプ 1 索引をタイプ 2 索引に変換するには、 REORG INDEXES コマンドを実行します。

タイプ 2 索引には、主に次のような利点があります。

- 長さが 255 バイトを超える列に対して、索引を作成することができます。
- ネクスト・キー・ロッキングの使用が最小限に抑えられて、並行性が改善されます。キーは次ページから除去される代わりに削除済みとマークされるので、ほとんどのネクスト・キー・ロッキングは制限されます。キー・ロッキングについての詳細は、ロッキングがパフォーマンスに与える影響に関するトピックを参照してください。

以下の状況では、削除済みとマークされた索引キーがクリーン・アップされます。

- その後の挿入、更新、または削除アクティビティー中

  キーの挿入中、クリーンアップすればページ分割の必要がなくなり、索引サイズの増加を防げるような場合には、削除済みとマークされてコミット済みと認識されたキーがクリーン・アップされます。

  キーの削除中、ページ上のすべてのキーが削除済みとマークされた場合には、すべてのキーが削除済みとマークされ、それらの削除がすべてコミット済みになった別の索引ページを見つけるよう試行されます。そのようなページが見つかると、索引ツリーから削除されます。

  キーを削除するとき、表に対する X ロックが存在すれば、キーは単に削除済みとマークされる代わりに、物理的に削除されます。この物理的削除の際、削除済みとマークされてコミット済みと認識されたキーが同じページ上に存在すれば、それらもすべて除去されます。
- CLEANUP オプションを使って REORG INDEXES コマンドを実行するとき

  CLEANUP ONLY PAGES オプションを指定すると、すべてのキーが削除済みとマークされてコミット済みと認識されている索引ページが検索されて、解放されます。

  CLEANUP ONLY ALL オプションを指定すると、すべてのキーが削除済みとマークされてコミット済みと認識されている索引ページが解放されるだけでなく、未削除の RID を含むページにおいて、削除済みとマークされてコミット済みと認識されている RID もまた解放されます。

  さらにこのオプションを指定すると、隣接するリーフ・ページをマージすれば最低でも PCTFREE のフリー・スペースがマージ後のページに確保されるような場合、リーフ・ページをマージするよう試行されます。 PCTFREE 値は、索引の作成時に定義されるフリー・スペースのパーセントです。デフォルトの PCTFREE は 10 % です。 2 つのページをマージできる場合、いずれかのページが解放されます。
- 索引の再作成時

索引を再作成するユーティリティーには、以下のものがあります。

- REORG INDEXES (いずれの CLEANUP オプションも使用しない場合)
- REORG TABLE (INPLACE オプションを使用しない場合)
- IMPORT (REPLACE オプションを使用する場合)
- LOAD (INDEXING MODE REBUILD オプションを使用する場合)

関連概念:

- 277 ページの『索引の利点と欠点』
- 279 ページの『索引の計画のヒント』
- 26 ページの『索引の構造』
- 270 ページの『表の再編成』
- 287 ページの『索引の再編成』

索引の再編成

表は、削除や挿入によって更新されるので、索引パフォーマンスは次のような場合に低下します。

- リーフ・ページのフラグメント化

  リーフ・ページがフラグメント化されると、表ページを取り出すためにさらに多くのリーフ・ページを読み取らなければならないので、入出力のコストは増えます。
- 物理的な索引ページ順序がそのページ上のキーの順序と一致しなくなった (これを、*不良クラスター* 索引という)。

  リーフ・ページが適切にクラスター化されないと、順次プリフェッチの効率が下がり、入出力の待ち時間が長くなります。
- 索引が、最も効率的なレベル数を超えています。

  この場合、索引を再編成する必要があります。

索引を作成するときに MINPCTUSED パラメーターを設定する場合、キーが削除され、フリー・スペースが、指定されたパーセント未満になっていると、データベース・サーバーは自動的に索引リーフ・ページをマージします。このプロセスを、*オンライン索引デフラグ* といいます。ただし、索引クラスタリングをリストアし、スペースを解放し、そしてリーフ・レベルを下げるには、以下に挙げるいずれかの方法を取ってください。

- 索引をドロップした後再作成します。
- REORG INDEXES コマンドを使用してオンラインで索引を再編成します。

  この方法は、実稼働環境で選択できる場合があります。なぜなら、この方法では、索引の再作成中に表の読み取り/書き込みが行えるからです。
- 表の再編成およびオフラインでの索引の再編成の両方を可能にするオプションを指定して、 REORG TABLE コマンドを使用します。

オンライン索引再編成

ALLOW WRITE ACCESS オプションを指定して REORG INDEXES コマンドを使用すると、表への読み取り/書き込みアクセスが許可されている間、指定された表の索引すべてが再作成されます。再編成の進行中に、基礎表に加えられ、索引に影響を与える変更は、DB2® ログに記録されます。さらに、使用可能なメモリー・スペースがある場合は、その変更内容は内部メモリー・バッファー・スペースにも記録されます。再編成によって、ログに記録された変更は処理され、索引の再作成時に現行の書き込みアクティビティーにキャッチアップします。内部メモリー・バッファー・スペースは、ユーティリティー・ヒープからオンデマンドで割り振られる特定のメモリー域で、作成中または再編成中の索引に加えられる変更を保管します。メモリー・バッファー・スペースの使用によって、索引の再編成は、まずメモリーから変更を直接読み取ることによって処理し、次いで必要であれば、もっと後にログ全体を読み取ることによって実行できます。割り振られたメモリーは、再編成操作が完了すると解放されます。再編成の完了後には、再作成された索引は完全にクラスター化されない場合があります。索引に対して PCTFREE が指定される場合、再編成時にそのスペースの比率が各ページに保存されます。

注: REORG INDEXES コマンドの CLEANUP ONLY オプションによって索引を完全に再編成することはできません。 CLEANUP ONLY ALL オプションは、削除済みとしてマークされ、コミット済みとして認識されるキーを除去します。また、このオプションは、削除済みとしてマークされ、コミット済みとして認識されるページの解放も行います。ページが解放されると、隣接するリーフ・ページがマージされます (ただし、これを行って、マージされたページに少なくとも PCTFREE フリー・スペースが残る場合)。 PCTFREE は、索引の作成時に定義されたフリー・スペースのパーセンテージです。 CLEANUP ONLY PAGES オプションは、削除済みとしてマークされ、コミット済みとして認識されるページだけを削除します。

以下は、REORG INDEXES を実行する際の要件です。

- 索引および表に関する SYSADM、SYSMAINT、SYSCTRL、または DBADM 権限か、CONTROL 特権
- 索引の保管先表スペースのフリー・スペースの容量が、索引の現行サイズと等しい

  CREATE TABLE ステートメントを発行する際は、再編成対象の索引を LARGE 表スペースに置くようにしてください。
- 追加ログ・スペース

  REORG INDEXES は、そのアクティビティーをログに記録します。その結果、特にシステムがビジーであり、他の並行アクティビティーがログに記録される場合、再編成は失敗することがあります。

  注: ALLOW NO ACCESS オプションを指定した REORG INDEXES ALL が失敗する場合、索引は不正とマークされ、操作は未完了となります。ただし、ALLOW READ ACCESS オプションまたは ALLOW WRITE ACCESS を指定した REORG が失敗する場合、元の索引オブジェクトはリストアされます。

関連概念:

- 277 ページの『索引の利点と欠点』

- 279 ページの『索引の計画のヒント』
- 282 ページの『索引のパフォーマンスのヒント』
- 289 ページの『オンライン索引のデフラグ』
- 285 ページの『索引のクリーン・アップおよび保守』

関連タスク:

- 275 ページの『表を再編成する方式の選択』

オンライン索引のデフラグ

索引のリーフ・ページにある使用済みスペースの最小量としてユーザー定義可能な限界値により、オンライン索引デフラグが使用可能になります。リーフ・ページから索引キーが削除された場合にこの限界値を超えていると、索引の隣接するリーフ・ページがチェックされ、2 つのリーフ・ページをマージできるかどうか判別されます。 2 つの隣接するページをマージするのに十分なスペースがあれば、バックグラウンドで即時にマージが行われます。

索引のオンラインでのデフラグは、バージョン 6 以後のリリースで作成した索引でのみ行うことができます。既存の索引が、オンラインでのマージ機能を必要とする場合、その索引をいったんドロップしてから MINPCTUSED 文節を使って再作成することが必要です。 MINPCTUSED 値は、100 未満の値に設定します。隣接する索引のリーフ・ページをマージすることが目標なので、 MINPCTUSED の値は 50 より小さくすることをお勧めします。 MINPCTUSED の値をゼロ (デフォルト) にすると、オンラインでのデフラグは使用できません。

ページの最後の索引キーが除去されると、索引内のページが解放されます。ただし、CREATE INDEX ステートメントで MINPCTUSED 文節を指定した場合は例外です。 MINPCTUSED 文節は、索引リーフ・ページのスペースのパーセントを指定します。索引キーの削除時に、ページ上でいっぱいになったスペースのパーセントが、指定された値以下である場合には、データベース・マネージャーは残りのキーを隣接するページのキーにマージしようとします。隣接ページにスペースが十分にある場合には、マージが実行され、リーフ・ページが削除されます。

索引のノンリーフ・ページは、オンライン索引デフラグ中にマージされません。しかし、空のノンリーフ・ページは削除され、同じ表上の他の索引による再利用が可能になるようにされます。これらのノンリーフ・ページを DMS ストレージ・モデルの他のオブジェクト用に解放したり、 SMS ストレージ・モデルのディスク・スペースを開放したりするには、表または索引の完全な再編成を実行することが必要です。表および索引を完全に再編成すると、可能な限り索引を小さくすることができます。索引のノンリーフ・ページは、オンライン索引デフラグ中にマージされませんが、空になったときに削除され、再利用のために解放されます。索引中のレベルの数と、リーフ・ページおよびノンリーフ・ページの数が削減されることもあります。

タイプ 2 の索引の場合、キーは表に X ロックがある場合のみ、キー削除中にページから除去されます。そのような操作中は、オンラインでの索引デフラグが効果的です。しかし、キー削除中に表に X ロックがない場合、キーは削除のマークを付けられますが、物理的には索引ページから除去されません。その結果、デフラグは行われません。

キーに削除のマークが付けられたのに物理的には索引ページに残っているタイプ 2 の索引でデフラグを実行するには、CLEANUP ONLY ALL オプションを指定した REORG INDEXES コマンドを実行します。 CLEANUP ONLY ALL オプションは、MINPCTUSED の値にかかわらず索引のデフラグを実行します。 CLEANUP ONLY ALL を指定して REORG INDEXES を実行すると、そのようなマージによってマージされたページに少なくとも PCTFREE フリー・スペースを残す場合、 2 つの近隣のリーフ・ページがマージされます。 PCTFREE は、索引作成時に指定され、デフォルトは 10 % です。

関連概念:

- 277 ページの『索引の利点と欠点』
- 282 ページの『索引のパフォーマンスのヒント』
- 26 ページの『索引の構造』
- 287 ページの『索引の再編成』

DMS 装置に関する考慮事項

表スペースでデータベース管理ストレージ (DMS) デバイス・コンテナーを使用している場合、効率的な管理に関する以下の要因を検討してください。

- **ファイル・システム・キャッシュ**

  ファイル・システム・キャッシュは次のように実行されます。

  – DMS ファイル・コンテナー (およびすべての SMS コンテナー) の場合、オペレーティング・システムはファイル・システム・キャッシュ中のページをキャッシュすることがある

  – DMS デバイス・コンテナーの場合、オペレーティング・システムはファイル・システム・キャッシュ中のページをキャッシュしない

  注: Windows® NT の場合、レジストリー変数 DB2NTNOCACHE は、 DB2® が NOCACHE オプション付きでデータベース・ファイルをオープンするか否かを指定します。 DB2NTNOCACHE=ON の場合は、ファイル・システムのキャッシュは除去されます。 DB2NTNOCACHE=OFF の場合、オペレーティング・システムは DB2 ファイルをキャッシュに入れます。これは、LONG FIELDS または LOBS を含んでいるファイルを除くすべてのデータに適用されます。システム・キャッシュを除去すると、より多くのメモリーがデータベースに利用できるようになるため、バッファー・プールやソート・ヒープの量を増やすことができます。

- **データのバッファリング**

  ディスクから読み取られる表データは通常、データベースのバッファー・プールで使用可能です。アプリケーションが実際にページを使用してしまうよりも前に、特に別のデータ・ページでバッファー・プール・スペースが必要な場合に、データ・ページをバッファー・プールから解放できることもあります。システム管理ストレージ (SMS) またはデータベース管理ストレージ (DMS) ファイル・コンテナーを使用する表スペースについては、上記のファイル・システム・キャッシングにより入出力は必要なくなることもあります (ファイル・システム・キャッシングが使用されない場合、入出力は必要です)。

データベース管理ストレージ (DMS) を使用している表スペースは、ファイル・システムもキャッシュも使用しません。このような場合には、データベース・バッファー・プールのサイズを大きくし、ファイル・システム・キャッシュのサイズを小さくすることによって、デバイス・コンテナーを使用する DMS 表スペースは二重バッファリングを使用しないという事実を相殺することができます。

デバイス・コンテナーを使用する DMS 表スペースの入出力が同等の SMS 表スペースの入出力と比較して大きくなっていることを、システム・レベルのモニター・ツールが示す場合には、この差は、二重バッファリングが原因である可能性があります。

- **LOB データまたは LONG データの使用**

  アプリケーションが LOB データまたは LONG データのいずれかをリトリーブする場合には、データベース・マネージャーは、データをそのバッファーにキャッシュしません。アプリケーションがこれらのページの 1 つを要求するたびに、データベース・マネージャーはディスクからリトリーブしなければなりません。LOB または LONG データが SMS または DMS ファイル・コンテナーに格納される場合、ファイル・システム・キャッシュでバッファリングが行われ、結果としてパフォーマンスが向上することがあります。

  システム・カタログにはいくつかの LOB 列が含まれているので、システム・カタログは SMS 表スペースまたは DMS ファイル表スペースに入れておく必要があります。

関連概念:

- 13 ページの『データベース・ディレクトリーとファイル』
- 16 ページの『SMS 表スペース』
- 17 ページの『DMS 表スペース』

エージェント管理

このセクションでは、データベース・マネージャーがどのようにエージェントを使用するか、およびエージェントを管理してよいパフォーマンスを得る方法について説明します。

データベース・エージェント

アプリケーションがアクセスするそれぞれのデータベースごとに、各種プロセスまたはスレッドは、様々なアプリケーション・タスクの実行を開始します。これらのタスクには、ロギング、通信、プリフェッチなどが含まれます。

データベース・エージェントとは、エンジン・ディスパッチ可能単位 (EDU) プロセスまたはスレッドのことです。データベース・エージェントは、アプリケーションが要求するデータベース・マネージャー内で作業を行います。 UNIX® 環境では、これらのエージェントはプロセスとして実行し、 Windows® などの Intel ベースのオペレーティング・システムでは、エージェントはスレッドとして実行します。

アプリケーション接続の最大数は、 *max_connections* データベース・マネージャー構成パラメーターにより制御されます。各アプリケーション接続の作業は、1 つの作業エージェントによって調整されます。

作業エージェント は、アプリケーション要求を実行しますが、特定のアプリケーションに永続的にアタッチされるものではありません。コーディネーター作業エージェントには、アプリケーションが要求したデータベース・マネージャー内のアクションを完了するのに必要なすべての情報と制御ブロックがあります。

以下の 4 種類の作業エージェントがあります。

- アイドル・エージェント
- 非アクティブ・エージェント
- アクティブ・コーディネーター・エージェント
- サブエージェント

アイドル・エージェント

これは、最も単純な形式のエージェントです。このエージェントにはアウトバウンド接続がなく、またローカル・データベース接続もインスタンス接続もありません。

非アクティブ・エージェント

非アクティブ・エージェントは作業エージェントの一形式であり、アクティブ・トランザクションにありません。このエージェントにはアウトバウンド接続がなく、ローカル・データベース接続も、インスタンス接続もありません。非アクティブは、自由にアプリケーション接続の作業を開始できます。

アクティブ・コーディネーター・エージェント

クライアント・アプリケーションの各プロセス/スレッドにはそれぞれ、データベース上の作業を調整するアクティブ・エージェントが 1 つあります。コーディネーター・エージェントが作成された後、そのエージェントが、アプリケーションに代わって、すべてのデータベース要求を実行し、さらに、プロセス間通信 (IPC) および遠隔通信のプロトコルを使用して、他のエージェントとコミュニケーションします。各エージェント・プロセスは自らの専用メモリーを使って操作を行いますが、データベース・マネージャーおよびデータベース・グローバル・リソース (バッファー・プールなど) は他のエージェントと共用します。トランザクションが完了すると、アクティブ・コーディネーター・エージェントは非アクティブ・エージェントになります。

クライアントがデータベースから切断されるか、またはインスタンスから切り離されると、そのコーディネーター・エージェント状態は次のようになります。

- アクティブ・エージェント。他の接続が待機状態の場合は、作業エージェントがアクティブ・コーディネーター・エージェントになります。
- 他の接続が待機状態になく、プール・エージェントが最大数に達していない場合は、空き状態になり、アイドル中であることを示すマークが付けられます。
- 他の接続が待機状態になく、プール・エージェントが最大数に達した場合は、終了して、ストレージが解放されます。

サブエージェント

パーティション・データベース環境、およびパーティション内並列処理が使用可能になっている環境では、コーディネーター・エージェントはデータベース要求をサブエージェントに分配し、それらのサブエージェントがアプリケーションの要求を実行します。コーディネーター・エージェントが作成された後、このエージェントは、データベースへの要求を実行するサブエージェントの調整を行うことによって、アプリケーションに代わってすべてのデータベース要求の処理を行います。

どのアプリケーションの作業も実行せず、割り当てられるのを待っているエージェントは、アイドル・エージェントと見なされ、エージェント・プール に常駐します。これらのエージェントは、クライアント・プログラムの作業を行うコーディネーター・エージェントからの要求のために、あるいは、既存のコーディネーター・エージェントの作業を行うサブエージェントのために、使用することができます。使用可能なエージェントの数は、データベース・マネージャー構成パラメーター *maxagents* および *num_poolagents* によって変わります。

エージェントが作業を完了したときにまだデータベース接続が残っている場合、その接続はエージェント・プールに置かれます。接続コンセントレーターがデータベースで使用可能になっているかどうかに関係なく、エージェントが起動していないために一定の期間新しい要求を実行することができない場合、またアクティブ・エージェントおよびプール・エージェントの現行の数が *num_poolagents* より多い場合、エージェントは終了します。

次の種類のアプリケーションの場合に、エージェント・プール (*num_poolagents*) のエージェントは、コーディネーター・エージェントとして再使用されます。

- リモート TCP/IP ベースのアプリケーション
- UNIX ベースのオペレーティング・システム上のローカル・アプリケーション
- Windows オペレーティング・システム上のローカル/リモート・アプリケーション

その他の種類のリモート・アプリケーションは常に新規エージェントを作成します。エージェントが必要なときにアイドル・エージェントがない場合には、新規エージェントが動的に作成されます。新規エージェントを作成するには一定のオーバーヘッドが必要なので、アイドル・エージェントをクライアントに対してアクティブにできる場合、 CONNECT および ATTACH のパフォーマンスは向上します。

あるサブエージェントがアプリケーションの作業を行うとき、そのサブエージェントはアプリケーションに*関連付けられ*ます。割り当てられた作業が完了すると、サブエージェントはエージェント・プールに入れられますが、元のアプリケーションとの関連付けはそのまま残されます。そのアプリケーションが追加の作業を要求した場合、データベース・マネージャーは新規エージェントを作成する前に、まずアイドル・プール内にそのアプリケーションと関連するエージェントがないか検査します。

関連概念:

- 294 ページの『データベース・エージェント管理』
- 298 ページの『パーティション・データベースにおけるエージェント』
- 295 ページの『クライアント接続用の接続コンセントレーターの改善』

- 294 ページの『エージェントの数に影響を与える構成パラメーター』

データベース・エージェント管理

ほとんどのアプリケーションは、接続済みアプリケーションの数と、データベースによる処理が可能なアプリケーション要求の数の間の 1 対 1 の関係を確立します。ただし、作業環境によっては、接続済みアプリケーションの数と処理可能アプリケーション要求の数の間で多数対 1 の関係が必要です。

これらの要因を別々に制御する機能は、以下の 2 つのデータベース・マネージャー構成パラメーターによって提供されます。

- *max_connections* パラメーター。これは、接続済みアプリケーションの数を指定します。
- *max_coordagents* パラメーター。これは、処理可能なアプリケーション要求の数を指定します。

接続コンセントレーターは、 *max_connections* 値が *max_coordagents* 値より大きい場合に使用可能になります。

各アクティブ・コーディネーター・エージェントはグローバル・リソースのオーバーヘッドを必要とするので、このエージェントの数が多ければ、使用可能なデータベース・グローバル・リソースの上限に達する可能性も高くなります。使用可能なデータベース・グローバル・リソースの上限に達するのを避けるには、*max_connections* に、 *max_coordagents* より高い値を設定することができます。

関連概念:

- 298 ページの『パーティション・データベースにおけるエージェント』
- 295 ページの『クライアント接続用の接続コンセントレーターの改善』
- 294 ページの『エージェントの数に影響を与える構成パラメーター』

エージェントの数に影響を与える構成パラメーター

以下のデータベース・マネージャー構成パラメーターは、作成されるデータベース・エージェントの数、およびそれらが管理される方法を決定します。

- 「エージェントの最大数」(*maxagents*): 任意の時点で作動できるエージェントの数。この値は、すべてのアプリケーションで作動するエージェント (コーディネーター・エージェント、サブエージェント、非アクティブなエージェント、およびアイドル・エージェントを含む) の合計数に関して適用されます。
- 「エージェント・プール・サイズ」(*num_poolagents*): システム内で使用可能な状態に保たれるエージェントの合計数 (アクティブなエージェントと、エージェント・プール内のエージェントを含む)。このパラメーターのデフォルト値は、*maxagents* で指定された数値の半分です。
- 「プール内エージェントの初期数」(*num_initagents*): データベース・マネージャーの開始時に、この値に基づいて作業エージェントのプールが作成されます。これによって、初期の照会のパフォーマンスが速くなります。作業エージェントはすべてアイドル・エージェントとして開始します。

- 「最大接続数」(*max_connections*): 各パーティションごとにデータベース・マネージャー・システムに許容される接続の最大数を指定します。
- 「コーディネーター・エージェントの最大数」(*max_coordagents*): パーティション・データベース環境、および接続コーディネーター使用時にパーティション内並列処理が使用可能になっている環境では、この値によってコーディネーター・エージェント数が制限されます。
- 「最大並行エージェント数」(*maxcagents*): この値は、データベース・マネージャーが許可するトークン の数を制御します。クライアントがデータベースに接続しているときにデータベース・トランザクション (作業単位) が生じるたびに、コーディネーター・エージェントはそのトランザクションを処理する許可をデータベース・マネージャーから得る必要があります。この許可を、*処理トークン* といいます。データベース・マネージャーは、処理トークンを持っているエージェントだけに、データベースに対する作業単位の実行を許可します。トークンが使用不可である場合、エージェントがトランザクションを処理するためには、トークンが利用可能になるまで待機する必要があります。

  このパラメーターは、ピーク時の使用要件がシステム・リソース (メモリー、CPU、およびディスク) を超えるような環境で役立ちます。たとえばそのような環境では、負荷がピークのとき、ページングによってパフォーマンスが低下する場合があります。このパラメーターを使用して、負荷の制御やパフォーマンス低下を防ぐことができます。ただし、並行性または待ち時間、あるいはその両方に影響する可能性があります。

関連概念:

- 291 ページの『データベース・エージェント』
- 294 ページの『データベース・エージェント管理』
- 298 ページの『パーティション・データベースにおけるエージェント』

クライアント接続用の接続コンセントレーターの改善

比較的一過性の接続が多いインターネット・アプリケーション、およびそれに類するアプリケーションに対して、接続コンセントレーターのパフォーマンスが改善され、より多くのクライアント接続を効率的に処理できるようになりました。さらに、それぞれの接続のメモリー使用量も削減され、コンテキスト切り替えの数が減りました。

注: 接続コンセントレーターは、 *max_connections* 値が *max_coordagents* 値より大きい場合に使用可能になります。

多数のユーザーが同時接続する必要のある環境では、システム・リソースをより効率的に使用するために、接続コンセントレーターを使用可能にすることができます。この機能は、これまで DB2 Connect 接続プールでのみ利用できた機能を取り入れたものです。接続プールと接続コンセントレーターは、どちらも「DB2 Connect ユーザーズ・ガイド」で説明されています。最初の接続の後、接続コンセントレーターはホストへの接続時間を削減します。ホストからの切断が要求されると、インバウンド接続はドロップされますが、ホストへのアウトバウンド接続はプール内に保持されます。ホストへの新しい接続が要求されると、 DB2® は既存のアウトバウンド接続をプールから再使用することを試みます。

注: アプリケーションで接続プールまたは接続コンセントレーターを使用する場合、パフォーマンス最適化のために、キャッシュされるデータ・ブロック・サイズを制御するパラメーターを調整してください。詳細については、「DB2 Connect ユーザーズ・ガイド」を参照してください。

DB2Connect 接続プールおよび接続コンセントレーターを使用すると、アクティブなエージェントはクライアント接続の切断後にアウトバウンド接続を閉じないで、アプリケーション用のエージェント・プールに置かれます。ここでエージェントは、リモート・ホストとのアクティブな接続を保持する*論理サブエージェント* となります (これは、*論理コーディネーター・エージェント* によって制御されます)。

接続プールを使用する場合、 DB2 Connect™ はインバウンド TCP/IP 接続、アウトバウンド TCP/IP 接続、および SNA 接続だけを使用できます。 SNA を使って作業している場合は、接続がプールに入れられるよう、セキュリティー・タイプを NONE にする必要があります。接続プールでは、このようなアイドル状態のエージェントを*非アクティブなエージェント* といいます。非アクティブ・エージェントのプールは、アウトバウンド接続プールと同じものです。接続コンセントレーターもまた、アプリケーション固有のプールで後で使用するために、同じような方法で非アクティブ・エージェントを保持します。

使用例:

1. ある ESE 環境に 1 つのデータベース・パーティションがあり、平均して 1,000 ユーザーがデータベースに接続するとします。並行トランザクションの数は時々 200 に達しますが、 250 を超えることは決してありません。トランザクションは短期間です。

   このワークロードを処理するために、管理者は以下のようなデータベース・マネージャー構成パラメーターを設定します。

   - *max_connections* を 1000 に設定して、平均接続数を確実にサポートできるようにします。
   - *max_coordagents* を 250 に設定して、並行トランザクションの最大数をサポートします。
   - *maxagents* を十分な大きさに設定します。これは、ノードでトランザクションを実行するのに必要とされるすべてのコーディネーター・エージェントとサブエージェント (該当する場合) をサポートするためです。

     intra_parallel が OFF であれば、このような環境にはサブエージェントが存在しないので、 *maxagents* を 250 に設定します。 *intra_parallel* が ON であれば、ノード上のデータにアクセスする各トランザクションで必要とされるコーディネーター・エージェントとサブエージェントをサポートするために、 *maxagents* を十分に大きく設定する必要があります。たとえば、各トランザクションが 4 つのサブエージェントを必要とする場合、 *maxagents* を (4+1) x 250 = 1250 に設定します。 *maxagents* をさらに微調整するには、データベース・マネージャーのモニター・スナップショットを取ります。 *maxagents* の適切な設定値は、エージェントの最高水準点によって示されます。

- *num_poolagents* は、 *maxagents* の値に応じて、少なくとも 250、最大で 1250 に設定します。これによって、新規作成によるオーバーヘッドを避けながら、十分な数のデータベース・エージェントが着信するクライアント要求にサービスを提供できるようにします。

  ただし、あまり使用されない期間のリソース使用量を削減するために、この数値を小さくすることもできます。この値を低く設定しすぎると、エージェントはエージェント・プールに入れられる代わりに、割り振り解除されてしまいます。その場合、サーバーが平均的なワークロードを処理するために、新しいエージェントを作成する必要が生じます。

- *num_init_agents* は *num_poolagents* と同じ値に設定します。これは、アクティブなエージェントの数がわかっているためです。こうすれば、データベースは特定の要求を処理する前にエージェントを作成するのではなく、始動時に適切な数のエージェントを作成します。

ハードウェアがワークロードを処理する能力については、ここでは考慮しません。仮に、ハードウェアが X 個のエージェントを同時に処理できない場合、ハードウェアの処理能力の最大値までこの数値を小さくする必要があります。たとえば、この最大値がわずか 1500 エージェントであれば、並行処理されるトランザクションの最大数も制限されます。ある時点で他のノードに送られる要求の数を正確に判別することは常に可能とは限らないため、このようなパフォーマンス関連の設定値をよく観察する必要があります。

2. 例 1 と同じ数のユーザーが接続するものの、ワークロードが最大で 100 の並行トランザクションに制限されるようなシステムでは、データベース・マネージャー構成パラメーターを以下のように設定することができます。

   - *max_coordagents* を 100 に設定します
   - *num_poolagents* を 100 に設定します

   このように設定すれば、トランザクションを並行して実行できるクライアントの最大数は 100 になります。すべてのクライアントが切断したとき、 100 個のエージェントが新しいクライアント接続にサービスを提供するよう待機します。ただし *maxagents* の値は、ワークロードのタイプ、照会内並列処理の設定値、データベース・パーティション数、および基礎となるハードウェアに応じて設定する必要があります。

3. 次のケースとして、別の ESE 環境に 5 つのデータベース・パーティションがあり、各パーティションには平均 1000 ユーザーが接続し、並行トランザクションは 200 に達するものの 250 を超えることはないとします。この場合、データベース構成パラメーターを次のように設定します。

   - *max_coordagents* を 250 に設定します。これは、例 1 と同様に、最大で 250 クライアントが並行してトランザクションを実行するためです。
   - *maxagents* を 1500 に設定します。データが 5 つのパーティションに分散されることを想定すると、各トランザクションごとに、システム内の各ノードの少なくとも 1 つのサブセクションが実行される可能性があります。 ((1 コーディネーター・エージェント + 5 サブエージェント) x 250 = 1500)
   - *num_poolagents* を 1200 に設定します。 (平均して 200 の並行トランザクション、最大で 250 を想定しています。その結果、必要とされるエージェントの平均数は (1+5) x 200 = 1200 です。)

- *num_init_agents* は、例 1 と同様に、 *num_poolagents* と同じ値に設定します。

4. 接続コンセントレーターを使用可能にしないものの、同時に 250 ユーザー接続を処理する必要のあるシステムでは、データベース・マネージャー構成パラメーターを以下のように設定します。
   - *max_connections* を 250 に設定します。
   - *max_coordagents* を 250 に設定します。

関連概念:

- 291 ページの『データベース・エージェント』
- 294 ページの『データベース・エージェント管理』
- 9 ページの『DB2 アーキテクチャーおよびプロセスの概要』
- 37 ページの『メモリー管理』

パーティション・データベースにおけるエージェント

パーティション・データベース環境、およびパーティション内並列処理が使用可能になっている環境の場合には、各パーティション (つまり、各データベース・サーバーまたはノード) が独自のエージェント・プールを持っていて、そこからサブエージェントを引き出すことができます。このプールがあるので、必要になったり作業を終了したりするたびに、サブエージェントを作成したり破棄したりする必要がありません。サブエージェントはプール内に関連エージェントとして残されるので、それらが関連付けされたアプリケーションから新しい要求が出された場合には、データベース・マネージャーによってそれらのサブエージェントが使用されます。

注: 接続コンセントレーターが使用可能な場合、サブエージェントは必ずしもアプリケーションと関連付けられているとは限りません。

パーティション・データベース環境およびパーティション内並列処理が使用可能になっている環境の場合、システム内のパフォーマンスとメモリー・コストへの影響は、以下に示すエージェント・プールのチューニング方法に強く関係しています。

- エージェント・プール・サイズに関するデータベース・マネージャー構成パラメーター (*num_poolagents*) は、 1 つのパーティション (ノードとも呼ばれる) でアプリケーションとの関連付けを保持できるサブエージェントの数に影響します。プール・サイズが小さすぎるので、プールが満杯になった場合には、サブエージェントは作業を行っているアプリケーションと自分自身との関連付けを切り離し、終了します。サブエージェントを作成してアプリケーションと再度関連付けをするということを常に行わなければならないため、パフォーマンスが低下します。

  さらに、*num_poolagents* の値が小さすぎた場合には、ある 1 つのアプリケーションが関連サブエージェントによってプールを満杯にしてしまう場合があります。そして、他のアプリケーションが新しいサブエージェントを要求したときに、関連エージェント・プール内にサブエージェントがないとすると、そのアプリケーションは、他のアプリケーションのエージェント・プールからサブエージェントを「スチール」します。この状況はコストが高くつくので、パフォーマンス低下を招きます。

- エージェントを少ししか持たないことと、任意の時点で多数のエージェントをアクティブにする場合のリソース・コストと比較してください。

  たとえば、*num_poolagents* の値が大きすぎる場合には、関連するサブエージェントは、長い間未使用のままプール内に置かれ、他のタスクでは使用可能になっていないデータベース・マネージャー・リソースが使用される可能性があります。

  注: 接続コンセントレーターが使用可能な際は、 *num_poolagents* によって指定されたエージェント数のみを使用するようお勧めします。それ以上のエージェントは、任意の時点でエージェント・プールに置かれる場合があります。

その他の非同期プロセスおよび非同期スレッド

データベース・マネージャーが独自のプロセスまたはスレッドとして実行する非同期アクティビティーは、データベース・エージェント以外にもあります。たとえば、次のようなアクティビティーがあります。

- データベース入出力サーバーまたは入出力プリフェッチャー
- データベース非同期ページ・クリーナー
- データベース・ロガー
- データベース・デッドロック検出機能
- イベント・モニター
- 通信および IPC listener
- 表スペース・コンテナー再平衡機能

関連概念:

- 263 ページの『プリフェッチと並列処理のための入出力サーバー構成』
- 264 ページの『並列入出力を使用したプリフェッチの図』
- 291 ページの『データベース・エージェント』
- 294 ページの『データベース・エージェント管理』
- 294 ページの『エージェントの数に影響を与える構成パラメーター』

データベース・システム・モニター情報

DB2® データベース・マネージャーは、 DB2® データベース・マネージャーの操作、パフォーマンス、および DB2 データベース・マネージャーを使用するアプリケーションに関するデータの保守を行います。 こういったデータは、データベース・マネージャーの実行時に保守され、そこから重要なパフォーマンス情報やトラブルシューティング情報を得ることができます。たとえば、得ることができる情報には以下のようなものがあります。

- データベースに接続しているアプリケーションの数、それらの状況、および各アプリケーションが実行している SQL ステートメント。
- データベース・マネージャーおよびデータベースの構成がどのくらい適切であるかを示して、それらの構成をチューニングする際に役立つ情報。
- 指定されたデータベースでデッドロックが生じた場合、関係していたアプリケーションはどれか、および競合していたロックはどれか。

- あるアプリケーションまたはデータベースが保持しているロックのリスト。アプリケーションがあるロックを待機しているために先に進めないという場合には、そのロックに関する追加情報 (どのアプリケーションがロックを保持しているのかなど) も示されます。

これらのデータの中には収集すると DB2 の操作にオーバーヘッドをかけるものがあるので、どの情報を収集するのかを制御する**モニター・スイッチ**が用意されています。明示的にモニター・スイッチを設定するには、UPDATE MONITOR SWITCHES コマンドか sqlmon() API を使用します。 (使用には、 SYSADM、 SYSCTRL、 SYSMAINT のいずれかの権限が必要です。)

スナップショットをとるか、イベント・モニターを使用するかのいずれかの方法により、データベース・マネージャーが保守するデータにアクセスすることができます。

スナップショットをとる

以下の 2 つのうちのいずれかの方法によってスナップショットをとることができます。

- コマンド行から GET SNAPSHOT コマンドを使用します。
- sqlmonss() API 呼び出しを使用して独自のアプリケーションを作成します。

イベント・モニターの使用法

イベント・モニターは、トランザクションの終了、ステートメントの終了、デッドロックの検出などの特定のイベントが起きた後に、システム・モニター情報を収集します。この情報は、ファイルまたは名前付きパイプに書き込まれます。

イベント・モニターを使用するには、以下のことを行ってください。

1. コントロール・センターまたは SQL ステートメントの CREATE EVENT MONITOR を使用して、イベント・モニターの定義を作成します。このステートメントを使用すると、定義はデータベース・システム・カタログに保管されます。
2. コントロール・センター、または次の SQL 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 evname STATE 1
   ```

   名前付きパイプに書き込む場合には、イベント・モニターを活動化する前に、名前付きパイプからの読み取りを行うアプリケーションを開始してください。これを行うには、ユーザー独自のアプリケーションを作成するか、または **db2evmon** を使用することができます。イベント・モニターがアクティブ化されてイベントのパイプへの書き込みが開始された後に **db2evmon** を使用すると、生成されるイベントを読み取って、それらを標準出力へ書き出します。
3. トレースを読み取ります。ファイル・イベント・モニターを使用している場合には、以下のどちらかの方法で、モニターが作成する バイナリー・トレースを見ることができます。
   - **db2evmon** ツールを使用して、トレースを標準出力に形式設定する方法。
   - Windows® ベースのオペレーティング・システムのコントロール・センターの**「イベント解析プログラム (Event Analyzer)」**アイコンをクリックして、グ

ラフィカル・インターフェースを使用し、トレースの表示、キーワードの探索、および不必要な情報を除いた表示を行う方法。

注: モニターのデータベース・システムが、コントロール・センターと同じマシンで実行していない場合、トレースを表示する前にコントロール・センターと同じマシンに、イベント・モニター・ファイルをコピーする必要があります。別の方法としては、両方のマシンにアクセス可能なファイル共用システムにファイルを配置します。

関連概念:

- 7 ページの『パフォーマンス・チューニングのためのクイック・スタート・ヒント』

第 9 章 ガバナー・プログラムの使用

この章では、ガバナー・プログラム・ツールをセットアップして実行し、データベース・アクティビティーをモニターおよび制御する方法を説明します。

ガバナー・ユーティリティー

ガバナーは、データベースに対して実行しているアプリケーションの動作をモニターし、ガバナー構成ファイルで指定した規則によって、特定の動作を変更することができます。

ガバナー・インスタンスは、フロントエンド・ユーティリティーおよび 1 つ以上のデーモンで構成されています。開始するガバナーの各インスタンスは、データベース・マネージャーのインスタンスに特定のものです。デフォルトでは、ガバナーを開始すると、パーティション・データベースの各パーティションでガバナー・デーモンが開始します。ただし、モニターしたい単一のパーティションでデーモンが開始するように指定することもできます。

注: ガバナーがアクティブになると、そのスナップショット要求によって、データベース・マネージャーのパフォーマンスに影響が出る可能性があります。パフォーマンスを向上させるには、ガバナー・ウェイクアップ・インターバルを大きくすることにより CPU の使用を削減してください。

それぞれのガバナー・デーモンは、データベースに対して実行しているアプリケーションについての情報を収集します。そしてその情報を、このデータベースについてガバナー構成ファイルで指定した規則と比較して検査します。

ガバナーは、構成ファイルの規則によって指定されたとおりにアプリケーション・トランザクションを管理します。たとえば、規則を適用すると、アプリケーションは特定のリソースを過剰に使用していることが示されたとします。規則は、アプリケーションの優先順位を変更する、またはそのアプリケーションをデータベースから強制切断するなどの取るべき処置を指定します。

規則に関連した処置がアプリケーションの優先順位を変更する場合、ガバナーはリソース違反が起こったデータベース・パーティションに対するエージェントの優先順位を変更します。パーティション・データベースでは、アプリケーションがデータベースから強制切断される場合、違反を検出したデーモンがそのアプリケーションのコーディネーター・ノードで実行されていても、その処置が行われます。

ガバナーは、ガバナーが行った処置のログをすべて記録します。処置について検討するには、ログ・ファイルを照会してください。

関連概念:

- 305 ページの『ガバナー・デーモン』
- 307 ページの『ガバナー構成ファイル』
- 316 ページの『ガバナー・ログ・ファイル』

関連タスク:

- 304 ページの『ガバナーの開始と停止』
- 306 ページの『ガバナーの構成』

関連資料:

- 「コマンド・リファレンス」の『db2gov - DB2 管理プログラム・コマンド』

ガバナー・の始動およびシャットダウン

このセクションでは、ガバナー・ツールの始動と停止の方法、およびガバナー・デーモンのアクティビティーについて説明します。

ガバナーの開始と停止

ガバナー・ユーティリティーは、データベースに接続しているアプリケーションをモニターし、そのデータベースに対するガバナー構成ファイルで指定した規則に従って、それらのアプリケーションの動作を変更します。

前提条件:

ガバナーを開始する前に、構成ファイルを作成する必要があります。

制約事項:

ガバナーを開始または停止するには、 *sysadm* または *sysctrl* 権限を持っていなければなりません。

手順:

ガバナーを開始または停止するには、次のようにします。

1. ガバナーを開始するには、DB2 コマンド行で *db2gov* コマンドを実行します。以下の必要パラメーターを入力してください。
   - START *database_name*

     指定するデータベース名は、指定する構成ファイルでのデータベース名と同じものにしなければなりません。名前が異なると、エラーが戻されます。ガバナーが複数のデータベースに対して実行している場合、それぞれのデータベースごとにデーモンが開始されることに注意してください。
   - *config_file_name*

     そのデータベースに対するガバナーの構成ファイル名。ファイルがデフォルトのロケーション (`sqllib` ディレクトリー) にない場合、ファイル名と共にパスも含める必要があります。
   - *log_file_name*

     このガバナーに対するログ・ファイルの基本名。パーティション・データベースでは、ガバナーのこのインスタンスに対してデーモンが実行されるパーティションごとに、パーティション番号が付加されます。

パーティション・データベースの単一のパーティションでガバナーを開始するには、 *nodenum* オプションを追加してください。たとえば、*sales* というデータベースに対するガバナーを、パーティション・データベースのノード 3 のみで、*salescfg* という構成ファイルおよび *saleslog* というログ・ファイルを使用して開始するには、以下のコマンドを入力してください。

```
db2gov START sales nodenum 3 salescfg saleslog
```

sales データベースのすべてのパーティションでガバナーを開始するには、以下のコマンドを入力してください。

```
db2gov START sales  salescfg saleslog
```

2. ガバナーを停止するには、 STOP オプションを指定して *db2gov* コマンドを入力します。

   たとえば、 *sales* データベースのすべてのパーティションでガバナーを停止するには、以下のコマンドを入力してください。

   ```
   db2gov STOP sales
   ```

   パーティション 3 でのみガバナーを停止するには、以下のコマンドを入力してください。

   ```
   db2gov START sales nodenum 3
   ```

関連概念:

- 303 ページの『ガバナー・ユーティリティー』
- 305 ページの『ガバナー・デーモン』

関連資料:

- 「コマンド・リファレンス」の『db2gov - DB2 管理プログラム・コマンド』

ガバナー・デーモン

ガバナー・デーモンは、db2gov ユーティリティーによって実行するか、またはウェイクアップするかのいずれかで開始すると、以下のタスク・ループを実行します。

1. ガバナー構成ファイルが変更されたか、またはファイルがまだ読み取られていないかどうかを検査します。いずれかの条件が真である場合には、デーモンはファイル内の規則を読み取ります。これによって、ガバナー・デーモンの実行中にその動作を変更することができるようになります。
2. データベース上で作動しているアプリケーションおよびエージェントごとに、リソース使用統計についてのスナップショット情報を要求します。

   注: 一部のプラットフォームでは、 CPU 統計を DB2® モニターから取得することはできません。その場合には、会計規則や CPU 制限も使用できません。
3. 統計を各アプリケーションごとに、ガバナー構成ファイル内の規則に照らして検査します。規則がアプリケーションに適用される場合、ガバナーは指定された処置を実行します。

   注: ガバナーは、累積された情報と構成ファイルで定義された値とを比較します。これは、アプリケーションが既にブリーチしている可能性がある新規の

値で構成ファイルが更新されている場合、そのブリーチに関係しているガバナーが、次のガバナーのインターバルに即時に適用されることを意味します。

4. これは実行するすべての処置について、ガバナー・ログ・ファイルにレコードを書き込みます。

注: ガバナーは、 *agentpri* データベース・マネージャー構成パラメーターがシステム・デフォルト以外である場合には、エージェントの優先順位を調整するために使用することはできません。 (この注は、Windows® NT プラットフォームには適用されません。)

ガバナーは、タスクを終えると、構成ファイル内で指定されたインターバルの間はスリープします。そのインターバルが経過すると、ガバナーはウェイクアップして、タスク・ループを再度実行します。

ガバナーがエラーまたはストップ信号を検出した場合、クリーンアップ処理を行ってから終了します。クリーンアップ処理では、優先順位が設定してあるアプリケーションのリストを使用して、すべてのアプリケーション・エージェントの優先順位がリセットされます。続いて、すでにアプリケーション上で処理を行っていないエージェントの優先順位もすべてリセットされます。こうすることにより、ガバナーの終了後には、デフォルトでない優先順位で実行されているエージェントがないようにします。エラーが起きた場合、ガバナーは、異常終了したことを示すメッセージを管理通知ログに書き込みます。

注: ガバナー・デーモンはデータベース・アプリケーションではなく、そのためデータベースへの接続は維持しませんが、インスタンスの接続はあります。スナップショット要求を出すことができるので、ガバナー・デーモンは、データベース・マネージャーが終了した時点を検出することができます。

関連概念:

- 303 ページの『ガバナー・ユーティリティー』

関連タスク:

- 304 ページの『ガバナーの開始と停止』

ガバナーの構成

このセクションでは、ガバナーを構成し、データベース・アクティビティーをモニターおよび制御する方法を説明します。

ガバナーの構成

ガバナーを構成するには、ガバナーのインスタンスがモニターするデータベース、および照会の管理方法を決定する、構成ファイルを作成します。

構成ファイルは、規則のセットで構成されています。最初の 3 つの規則は、モニターするデータベース、ログ・レコードを書き込むインターバル、およびモニターのためにウェイクアップするインターバルを指定します。残りの規則は、データベース・サーバーのモニター方法、および特定の状況でどのような処置を取るかを指定します。

手順:

ガバナー構成ファイルを作成するには、次のようにします。

1. すべてのデータベース・マネージャー・パーティションに取り付けられていて使用可能なディレクトリーに、記述名を持った ASCII ファイルを作成します。たとえば、 **sales** データベースをモニターするガバナー・インスタンスの構成ファイルに、 *govcfgsales* といった名前を付けることができます。
2. 任意のテキスト・エディターでそのファイルを開き、構成情報および処置の条件を入力します。

   各規則の終わりには、セミコロン (;) を置きます。以下の構成情報が推奨されています。

   - dbname: モニター対象となるデータベースの名前または別名。
   - account: ガバナー・インスタンスが CPU 使用統計をログ・ファイルに書き込む、分単位の間隔。このオプションは、Windows NT では使用できません。
   - interval: ガバナー・デーモンがウェイクアップしてアクティビティーをモニターする、秒単位のインターバル。インターバルを指定しない場合、デフォルト値の 120 秒が使用されます。

   たとえば、構成ファイル内の最初の 3 つの規則は次のようになります。

   ```
   { Wake up once a second, the database name is sales,
    do accounting every 30 minutes.  }
   interval 1; dbname sales; account 30;
   ```

   モニターする状態、およびその規則が真と評価されたときに取る処置を指定する規則を追加します。たとえば、次のようにして、作業単位 (UOW) が実行する時間を 1 時間に制限し、その後はデータベースから強制切断するという規則を追加することができます。

   ```
   setlimit uowtime 3600 action force;
   ```

3. ファイルを保管します。

関連概念:

- 307 ページの『ガバナー構成ファイル』
- 310 ページの『ガバナーの規則のエレメント』
- 315 ページの『ガバナー構成ファイルの例』
- 316 ページの『ガバナー・ログ・ファイル』

関連資料:

- 「コマンド・リファレンス」の『db2gov - DB2 管理プログラム・コマンド』

ガバナー構成ファイル

ガバナーを開始するときには、データベースに対して実行されるアプリケーションを管理する規則を含んだ構成ファイルを指定します。ガバナーはそれぞれの規則を評価し、規則が真と評価されたときに、指定されたとおりの処置を行います。

規則要件が変更になった場合には、ガバナーを停止しないで構成ファイルを編集することができます。各ガバナー・デーモンは、構成ファイルが変更されたことを検出し、ファイルを再度読み取ります。

構成ファイルは、各パーティション上のガバナー・デーモンが同一の構成ファイルを読み取れるように、すべてのデータベース・パーティションに取り付けられるディレクトリーに作成する必要があります。

構成ファイルは、モニターするデータベース、ログ・レコードを書き込むインターバル、およびガバナー・デーモンのスリープ・インターバルを識別する 3 つの必須規則で構成されています。これらのパラメーターに続いて、構成ファイルには、オプションのアプリケーション・モニターに関する規則および処置が含まれています。次のコメントは、すべての規則に適用されます。

- 注釈は、中括弧 { } に入れて区切ります。
- 大部分の項目は、英大文字、英小文字、または英大文字小文字混合文字で指定することができます。例外はアプリケーション名で (`applname` 規則の引き数として指定されます)、これは大文字小文字の区別をします。
- 各規則は、セミコロン (;) で終わります。

必須規則

以下の規則によって、モニターするデータベース、およびデーモンが各活動のループ後にウェイクアップするインターバルを指定します。これらの規則はそれぞれ、ファイルに 1 回のみ指定されます。

dbname
モニター対象となるデータベースの名前または別名。

account *nnn*
各接続ごとの CPU 使用率統計を含む会計レコードが、指定された数の分ごとに書き出されます。

注: このオプションは、Windows NT では使用できません。

短い接続セッションが全体的にアカウント・インターバル内で発生する場合、ログ・レコードは作成されません。ログ・レコードが作成される場合、そこには前の接続に関するログ・レコード以来の CPU 使用量を反映する CPU 統計が含まれます。ガバナーが停止してから再始動された場合、 CPU 使用量は 2 つのログ・レコードで反映される場合があります。これらはログ・レコードのアプリケーション ID を介して識別できます。

interval
デーモンがウェイクアップする時間間隔 (秒単位)。インターバルを指定しない場合、デフォルト値の 120 秒が使用されます。

処置を管理する規則

必須規則に続けて、アプリケーションの管理方法を指定する規則を追加することができます。これらの規則は、規則文節と呼ばれるより小さなコンポーネントから成ります。それらを使用する場合、文節は次のように、規則ステートメントに特定の順序で入力しなければなりません。

1. **desc** (オプション): 規則に関する注釈。引用符で囲みます。
2. **time** (オプション): 規則が評価される時間帯。
3. **authid** (オプション): アプリケーションがステートメントを実行する、 1 つ以上の許可 ID。

4. **applname** (オプション): データベースに接続する実行可能アプリケーション、またはオブジェクト・ファイルの名前。この名前には、大文字と小文字の区別があります。アプリケーション名にスペースが含まれる場合には、その名前を二重引用符で囲む必要があります。
5. **setlimit**: ガバナーが検査する制限。たとえば、これらには CPU 時間、戻される行の数、またはアイドル時間などがあります。
6. **action** (オプション): 制限に達した場合に実行する処置。処置が指定されていない場合、制限に達すると、ガバナーはアプリケーションに対して作動しているエージェントの優先順位を 10 低くします。アプリケーションに対する処置には、エージェントの優先順位を下げる、データベースから強制切断する、または運用についてのスケジューリング・オプションを設定することが含まれます。

規則文節を組み合わせて、1 つの規則を作ります。次の例に示すように、各文節は規則ごとに 1 回のみ使用し、規則の終わりにセミコロンを置きます。

```
desc "Allow no UOW to run for more than an hour"
setlimit uowtime 3600 action force;

desc "Slow down the use of db2 CLP by the novice user"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;
```

複数の規則がアプリケーションに適用される場合、そのすべてが適用されます。通常、最初に検出された規則制限に関連付けられた処置が最初に適用される処置となります。例外は、規則内の文節に -1 を指定した場合です。この場合には、後続する規則の文節の値は、それより前に*同じ* 文節に指定された値のみをオーバーライドします。このとき、前に置かれた規則の他の文節は、有効なままになります。たとえば、ある規則が rowssel 100000 uowtime 3600 文節を使用して、経過時間が 1 時間を超えるか、または選択された行数が 100 000 行を超えるかした場合には、そのアプリケーションの優先順位を低くするように指定しているとします。 また、後続する規則では uowtime -1 文節を使用して、同じアプリケーションに無制限の経過時間を許可するように指定しているとします。この場合、アプリケーションが 1 時間を超えて実行されたとしても、その優先順位は変更されません。つまり、uowtime -1 が uowtime 3600 をオーバーライドするということです。ただし、アプリケーションが 100 000 行を超える行を選択した場合は、 rowssel 100000 が有効なままであるために、優先順位が下げられます。

規則適用の順序

ガバナーは、構成ファイル内の規則をファイルの最初から最後まで処理します。ただし、後の規則の **setlimit** 文節が前の規則よりも緩やかな場合は、より制限的な規則が適用されます。たとえば、以下の構成ファイルでは、後の規則にかかわらず、admin は 5000 行に制限されます。最初の規則の方がより制限的であるためです。

```
desc "Force anyone selecting 5000 or more rows"
setlimit rowssel 5000 action force;

desc "Allow user admin to select more rows"
authid admin
setlimit rowssel 10000 action force;
```

緩やかな規則で、ファイル内の先の部分に出てくるより制限的な規則をオーバーライドするには、-1 オプションを指定して、新しい規則を適用する前に、前の規則を

クリアします。たとえば、以下の構成ファイルでは、最初の規則がすべてのユーザーを 5000 行に制限します。 2 番目の規則は admin に対するこの規則をクリアし、 3 番目の規則は admin の制限を 10000 行にリセットします。

```
desc "Force anyone selecting 5000 or more rows"
setlimit rowssel 5000 action force;

desc "Clear the rowssel limit for admin"
authid admin
setlimit rowssel -1;

desc "Now set the higher rowssel limit for admin"
authid admin
setlimit rowssel 10000 action force;
```

関連概念:

- 310 ページの『ガバナーの規則のエレメント』
- 315 ページの『ガバナー構成ファイルの例』

ガバナーの規則のエレメント

ガバナー構成ファイル内の各規則は、規則の適用に関する条件、および規則が真と評価される場合に取られる処置を指定する文節から構成されています。文節は、以下に説明する順序で指定する必要があります。文節の記述において、[ ] はオプションの文節であることを示します。

オプションの先頭エレメント

[desc] 規則に関するテキスト記述を指定します。記述は、単一引用符か二重引用符のいずれかで囲む必要があります。

[time] 規則が適用される時間帯を指定します。

時間帯は、time hh:mm hh:mm (たとえば、 time 8:00 18:00) という形式で指定する必要があります。この文節が指定されない場合は、規則は全日 (24 時間) 有効になります。

[authid]

アプリケーションを実行する許可 ID (authid) を 1 つまたは複数指定します。複数の authid を指定する場合は、たとえば authid gene, michael, james のように、コンマ (,) で区切る必要があります。この文節が規則になかった場合には、規則はすべての authid に適用されます。

[applname]

データベースへの接続を行う実行可能アプリケーション (または、オブジェクト・ファイル) の名前を指定します。

複数のアプリケーション名を指定する場合は、たとえば applname db2bp, batch, geneprog のように、コンマ (,) で区切る必要があります。この文節が規則になかった場合には、規則はすべてのアプリケーション名に適用されます。

注:

1. アプリケーション名は、大文字小文字を区別する必要があります。
2. データベース・マネージャーは、すべてのアプリケーション名を 20 文字で切り捨てます。管理したいアプリケーションがアプリケーション名

の最初の 20 文字で固有に識別可能であることを確認しておく必要があります。確認を怠ると、望まないアプリケーションを管理対象としてしまう可能性があります。

ガバナー構成ファイルに指定されたアプリケーション名は 20 文字で切り捨てられて、構成ファイルの内部表記に一致させられます。

Limit clauses

setlimit

ガバナーが検査する制限を 1 つまたは複数指定します。制限値は、-1 か、さもなければ 0 より大きい値にしなければなりません (たとえば、 `cpu -1 locks 1000 rowssel 10000`)。制限 (cpu、 locks、 rowsread、 uowtime) は、最低でも 1 つは指定する必要があり、規則によって指定されていない制限は、その特定の規則によって制限されません。ガバナーが検査できるのは、以下に示す制限です。

cpu *nnn*
: アプリケーションが使用可能な CPU の秒数を指定します。 -1 を指定すると、ガバナーはアプリケーションの CPU 使用の制限は行いません。

  注: このオプションは、Windows® NT 環境では使用できません。

locks *nnn*
: アプリケーションが保持できるロック数を指定します。 -1 を指定すると、ガバナーはアプリケーションが保持するロック数の制限は行いません。

rowssel *nnn*
: アプリケーションに戻される行数を指定します。この値は、コーディネーター・ノードでは非ゼロとなります。 -1 を指定すると、ガバナーは選択できる行数の制限は行いません。

uowtime *nnn*
: 作業単位 (UOW) が最初にアクティブになった時刻から経過可能な秒数を指定します。 -1 を指定すると、経過時間は制限されません。

  注: `sqlmon` (データベース・システム・モニター・スイッチ) API を使用して作業単位スイッチを非活動化した場合には、ガバナーが作業単位経過時間に基づいてアプリケーションを管理する機能に影響を与えます。ガバナーはモニターを使って、システムについての情報を収集します。データベース・マネージャー構成ファイルでスイッチをオフにすると、インスタンス全体がオフになり、ガバナーはそれ以上この情報を受け取りません。

idle *nnn*
: 接続において、指定された処置が行われる前に許されるアイドル状態の秒数を指定します。 -1 を指定すると、接続のアイドル時間は制限されません。

rowsread *nnn*

アプリケーションが選択できる行数を指定します。 -1 を指定すると、アプリケーションが選択できる行数は制限されません。

注: この制限値は、rowssel と同じものではありません。異なるのは、rowsread が結果セットを戻すために読み取りする必要のあった行数のカウントである点です。読み取りされた行数にはエンジンによるカタログ表の読み取りが含まれるので、索引使用時には行数が少なくなる可能性があります。

処置文節

[action]

指定された制限の 1 つまたは複数を超えた場合に取る処置を指定します。次のように処置を指定することができます。

注: 制限を超えたときに action 文節が指定されていなかった場合には、ガバナーは、アプリケーションのために処理を行っているエージェントの優先順位を 10 倍低くします。

nice *nnn*

アプリケーションのために処理を行っているエージェントの優先順位の変更を指定します。有効な値は -20 ～ +20 です。

このパラメーターを有効に使用するには、以下に注意してください。

- UNIX® ベースのプラットフォーム上では、 *agentpri* データベース・マネージャー・パラメーターをデフォルト値に設定する必要があります。デフォルト値にしないと、このパラメーターが優先順位の値をオーバーライドしてしまいます。
- Windows プラットフォームでは、 *agentpri* データベース・マネージャー・パラメーターと *優先順位* の処置とを一緒に使用することができます。

force アプリケーションにサービスを提供しているエージェントの強制停止を指定します。 (コーディネーター・エージェントを終了する場合は、FORCE APPLICATION を出します。)

schedule [class]

スケジューリングによって、すべてのアプリケーションにおける公平性を確保しながら同時に平均応答時間を最小化するという目的に向けて、アプリケーション上で処理を行っているエージェントの優先順位の調整が行われます。

ガバナーは、次の 3 つの基準に基づいて、スケジューリングの優先度が高いアプリケーションを選択します。

- 最も多くロックを保持しているアプリケーション

  これは、ロック待機を削減しようとするために選択されます。
- 経過時間の最も長いアプリケーション
- 見積もられた残り実行時間が最も短いアプリケーション

これは、できるだけ多くの短期間のステートメントを、インターバルの間に完了させようとするために選択されます。

各規準で上位の 3 つのアプリケーションには、他のアプリケーションよりも高い優先順位が与えられます。つまり、各基準のグループで 1 位のアプリケーションには最も高い優先順位が、その次のアプリケーションには 2 番目に高い優先順位が、そして 3 位のアプリケーションには 3 番目に高い優先順位が与えられます。単一のアプリケーションが複数の基準において 3 位以内となった場合、そのアプリケーションには最も高い順位となった基準において該当する優先順位が与えられ、他の基準においては、次に高い優先順位が次に順位の高かったアプリケーションに与えられます。たとえば、アプリケーション A は最も多くロックを保持しているが、見積もりの残り実行時間は 3 番目に短いとします。この場合、このアプリケーションには 1 つめの基準において最も高い優先順位が与えられ、見積もりの残り実行時間が 4 番目に短いアプリケーションに、その基準において 3 番目に高い優先順位が与えられます。

このガバナー規則によって選択されたアプリケーションは、 3 つのクラスに分けられます。それぞれのクラスごとに、ガバナーは上にリストした基準に基づいて、各クラスからの上位 3 つである、 9 個のアプリケーションを選択します。クラス・オプションを指定した場合、この規則によって選択されたすべてのアプリケーションが単一のクラスと見なされ、 9 個のアプリケーションが選択されて上記のように他より高い優先順位を与えられます。

複数のガバナー規則で同じアプリケーションが選択された場合、最後に選択された際の規則によって管理されます。

注: sqlmon (データベース・システム・モニター・スイッチ) API を使用してステートメント・スイッチを非活動化した場合には、ガバナーがステートメント経過時間に基づいてアプリケーションを管理する機能に影響を与えます。ガバナーはモニターを使って、システムについての情報を収集します。データベース・マネージャー構成ファイルでスイッチをオフにすると、インスタンス全体がオフになり、ガバナーはそれ以上この情報を受け取りません。

スケジュール処置には次のことが含まれます。

- それぞれ異なるグループのアプリケーションが、すべてのアプリケーションに平均に時間を分割することなく確実に時間を入手するようにします。

  たとえば、14 のアプリケーション (短いアプリケーション 3 つ、中程度 5 つ、長いアプリケーション 6 つ) が同時に実行している場合、これらは CPU を分割しているので、応答時間があまりないかもしれません。 データベース管理者は、中程度の長さのアプリケーションと、長いアプリケーションの 2 つのグループを設定できます。優先順位を使用して、ガバナーはすべての短いアプリケーションの実行を許可し、大部分を占める 3 つの中程度

のアプリケーションと 3 つの長いアプリケーションを、同時に確実に実行します。これを行うために、ガバナー構成ファイルには、中程度のアプリケーションに 1 つの規則、長いアプリケーションに別の規則が入っています。

以下に、この点を例証するガバナー構成ファイルの一部を示します。

```
desc "Group together medium applications in 1 schedule class"
applname medq1, medq2, medq3, medq4, medq5
setlimit cpu -1
action schedule class;

desc "Group together long applications in 1 schedule class"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;
```

- 複数のユーザー・グループのそれぞれ (たとえば、組織上の部門) が確実に等しい優先度を獲得できるようにします。

  あるグループが多数のアプリケーションを実行している場合でも、管理者は他のグループが自分のアプリケーション用に適度な応答時間を獲得できるようにすることができます。たとえば、3 つの部門 (金融、在庫、企画) が関係している場合、すべての金融ユーザーを 1 つのグループに、すべての在庫ユーザーを 2 つ目のグループに、すべての企画ユーザーを 3 つ目のグループに入れることができます。処理能力は 3 つの部門の間でより平均に、またはその逆に分割されます。以下に、この点を例証するガバナー構成ファイルの一部を示します。

  ```
  desc "Group together Finance department users"
  authid tom, dick, harry, mo, larry, curly
  setlimit cpu -1
  action schedule class;

  desc "Group together Inventory department users"
  authid pat, chris, jack, jill
  setlimit cpu -1
  action schedule class;

  desc "Group together Planning department users"
  authid tara, dianne, henrietta, maureen, linda, candy
  setlimit cpu -1
  action schedule class;
  ```

- ガバナーにすべてのアプリケーションをスケジュールさせます。

  クラス・オプションが処置に含まれていない場合、ガバナーはスケジュール処理の下に落とされるアクティブ・アプリケーションの数に基づいた、独自のクラスを作成し、アプリケーションが実行している照会に DB2 照会コンパイラーのコスト見積もりに基づいてアプリケーションを別のクラスに入れます。 管理者は、選択されるアプリケーションを限定しないことによって、すべてのアプリケーションをスケジュールするように選択できます。つまり、*applname* または *authid* 文節は提供されず、 *setlimit* 文節も制限を課しません。

注: 制限を超えたときに action 文節が指定されていなかった場合には、ガバナーは、アプリケーションのために処理を行っているエージェントの優先順位を低くします。

関連概念:

- 307 ページの『ガバナー構成ファイル』
- 315 ページの『ガバナー構成ファイルの例』

関連タスク:

- 306 ページの『ガバナーの構成』

ガバナー構成ファイルの例

以下の例は、アクションについていくつかの規則を設定するガバナー構成ファイルを示しています。

```
{ Wake up once a second, the database name is ibmsampl,
 do accounting every 30 minutes.  }
interval 1; dbname ibmsampl; account 30;

desc "CPU restrictions apply 24 hours a day to everyone"
setlimit cpu 600 rowssel 1000000 rowsread 5000000;

desc "Allow no UOW to run for more than an hour"
setlimit uowtime 3600 action force;

desc 'Slow down a subset of applications'
applname jointA, jointB, jointC, quryA
setlimit cpu 3 locks 1000 rowssel 500 rowsread 5000;

desc "Have governor prioritize these 6 long apps in 1 class"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;

desc "Schedule all applications run by the planning dept"
authid planid1, planid2, planid3, planid4, planid5
setlimit cpu -1
action schedule;

desc "Schedule all CPU hogs in one class which will control consumption"
setlimit cpu 3600
action schedule class;

desc "Slow down the use of db2 CLP by the novice user"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;

desc "During day hours do not let anyone run for more than 10 seconds"
time 8:30 17:00 setlimit cpu 10 action force;

desc "Allow users doing performance tuning to run some of
     their applications during lunch hour"
time 12:00 13:00 authid ming, geoffrey, john, bill
applname tpcc1, tpcc2, tpcA, tpvG setlimit cpu 600 rowssel 120000 action force;

desc "Some people should not be limited -- database administrator
  and a few others.  As this is the last specification in the
  file, it will override what came before."
authid gene, hershel, janet setlimit cpu -1 locks -1 rowssel -1 uowtime -1;
```

```
desc "Increase the priority of an important application so it always
      completes quickly"
applname V1app setlimit cpu 1 locks 1 rowssel 1 action priority -20;
```

関連概念:

- 307 ページの『ガバナー構成ファイル』
- 310 ページの『ガバナーの規則のエレメント』

関連タスク:

- 306 ページの『ガバナーの構成』

ガバナー・ログ・ファイルの使用

このセクションでは、ガバナー・ログ・ファイルについて、およびそれらを照会して情報を検索する方法を説明します。

ガバナー・ログ・ファイル

ガバナー・デーモンは処置を実行する度に、ログ・ファイルにレコードを書き込みます。処置には以下のものが含まれます。

- アプリケーションの強制停止
- ガバナー構成ファイルの読み取り
- アプリケーション優先順位の変更
- エラーまたは警告の検出
- 開始または終了

ガバナー・デーモンには、それぞれに別個のログ・ファイルがあります。ログ・ファイルが別になっていることで、多くのガバナー・デーモンが同一のファイルに同時に書き込みを行うことによって起こる可能性のある、ファイル・ロッキングのボトルネックを防ぐことができます。ログ・ファイルを 1 つにマージして照会するには、 `db2govlg` ユーティリティーを使用します。

ログ・ファイルは、 `sqllib` ディレクトリーの `log` サブディレクトリーに保管されます。ただし、 Windows® NT の場合、 `log` サブディレクトリーはインスタンス・ディレクトリーの下にあります。 `db2gov` コマンドを使用してガバナーを開始するときには、ログ・ファイルの基底名を指定します。管理対象となる各データベースの各パーティション用のログ・ファイルを区別するため、ログ・ファイル名には必ずデータベース名を含めてください。パーティション・データベース環境においては、各ガバナーごとにファイル名がユニークになるように、ガバナー・デーモンが実行されているパーティションのパーティション番号が、自動的にログ・ファイル名の後に付加されます。

ログ・ファイル・レコード・フォーマット

ログ・ファイルの各レコードの形式は、次のとおりです。

```
Date Time NodeNum RecType Message
```

注: *Date* および *Time* フィールドの形式は、 yyyy-mm-dd hh.mm.ss です。このフィールドでソートを行うことによって各データベース・パーティションごとのログ・ファイルをマージすることができます。

NodeNum フィールドは、ガバナーが実行されているデータベース・パーティションの番号を示します。

RecType フィールドには、ログに書き込まれるログ・レコードのタイプによって異なる値が入ります。フィールドに入れることができる値は、以下のとおりです。

- START: ガバナーが開始された
- STOP: ガバナーが停止された
- FORCE: アプリケーションが強制された
- NICE: アプリケーションの優先順位が変更された
- ERROR: エラーが起きた
- WARNING: 警告が起きた
- READCFG: ガバナーが構成ファイルの読み取りを行った
- ACCOUNT: アプリケーションの会計統計
- SCHEDGRP: エージェントの優先順位に変更が生じた

これらの値の一部について、以下で詳細に説明します。

START

START レコードは、ガバナーが始動するときに書き込まれます。 START レコードは、以下のようなフォーマットになります。

```
Database = <database_name>
```

STOP STOP レコードは、ガバナーが停止するときに書き込まれます。 STOP レコードは、以下のようなフォーマットになります。

```
Database = <database_name>
```

FORCE

FORCE レコードは、ガバナー構成ファイル内の規則の要求に従ってアプリケーションを強制することをガバナーが判別したときに書き出されます。 FORCE レコードは、以下のようなフォーマットになります。

```
<appl_name> <auth_id> <appl_id> <coord_partition> <cfg_line>
<restriction_exceeded>
```

各パラメーターの意味は以下のとおりです。

<coord_partition>
アプリケーションの調整パーティションの番号を指定します。

<cfg_line>
アプリケーションを強制する規則が位置する、ガバナー構成ファイル内の行番号を指定します。

<restriction_exceeded>
規則超過の詳細を提供します。以下の値が有効です。

- CPU: アプリケーション USR cpu と SYS cpu の合計時間 (秒単位)
- Locks: アプリケーションが保持したロックの合計数
- Rowssel: アプリケーションが選択した行の合計数

- Rowsread: アプリケーションが読み取った行の合計数
- Idle: アプリケーションがアイドルだった合計時間
- ET (経過時間): アプリケーションの現行作業単位が開始した (作業単位設定限度を超えた) ときからの経過時間

NICE NICE レコードは、ガバナー構成ファイル内で指定された優先順位アクションによって、アプリケーションの優先順位が変更されるときに書き込まれます。 NICE レコードは、以下のようなフォーマットになります。

```
<appl_name> <auth_id> <appl_id> <nice value> (<cfg_line>)
<restriction_exceeded>
```

各パラメーターの意味は以下のとおりです。

<nice value>
アプリケーションのエージェント・プロセス用の優先度の値の増分 (または減分) を指定します。

<cfg_line>
アプリケーションの優先順位を変更する規則が位置する、ガバナー構成ファイル内の行番号を指定します。

<restriction_exceeded>
規則超過の詳細を提供します。以下の値が有効です。

- CPU: アプリケーション USR cpu と SYS cpu の合計時間 (秒単位)
- Locks: アプリケーションが保持したロックの合計数
- Rowssel: アプリケーションが選択した行の合計数
- Rowsread: アプリケーションが読み取った行の合計数
- Idle: アプリケーションがアイドルだった合計時間
- ET (経過時間): アプリケーションの現行作業単位が開始した (作業単位設定限度を超えた) ときからの経過時間

ERROR
ERROR レコードは、ガバナーがシャットダウンするときに書き込まれます。

WARNING
WARNING レコードは、以下の状況でガバナー・ログに書き込まれます。

- アプリケーションを強制するために sqlefrce API が呼び出されたが、正の SQLCODE が戻された。
- スナップショット呼び出しが 1611 以外の正の SQLCODE を戻した (″SQL1661 No data was returned″)。
- スナップショットが -1224 (″SQL 1224N A database agent could not be started to service a request, or was terminated as a result of a database system shutdown or a force command″) または -1032 (″SQL1032N No start database manager command was issued″) 以外の負の SQLCODE を戻した。これらの戻りコードは、以前にアクティブだったインスタンスがダウンしたときに生じます。
- UNIX® 環境で、シグナル・ハンドラーをインストールする試みが行われたが失敗した。

ACCOUNT

ACCOUNT レコードは、以下の状況でガバナー・ログに書き込まれます。

- このアプリケーションの最後の ACCOUNT レコードが書き込まれたときから、このアプリケーションの agent_usr_cpu または agent_sys_cpu の値が変更された。
- アプリケーションがもはやアクティブでないことが分かった。

ACCOUNT レコードは、以下のようなフォーマットになります。

<auth_id> <appl_id> <applname> <connect time> <agent_usr_cpu delta> <agent_sys_cpu delta>

SCHEDGRP

SCHEDGRP レコードは、以下の場合に書き込まれます。

- アプリケーションがスケジューリング・グループに追加される場合
- アプリケーションがあるスケジューリング・グループから別のスケジューリング・グループへ移動する場合

SCHEDGRP レコードは、以下のようなフォーマットになります。

```
<appl_name> <auth_id> <appl_id> <cfg_line> <restriction_exceeded>
```

各パラメーターの意味は以下のとおりです。

<cfg_line>
アプリケーションをスケジュールする規則が位置する、ガバナー構成ファイル内の行番号を指定します。

<restriction_exceeded>
規則超過の詳細を提供します。以下の値が有効です。

- CPU: アプリケーション USR cpu と SYS cpu の合計時間 (秒単位)
- Locks: アプリケーションが保持したロックの合計数
- Rowssel: アプリケーションが選択した行の合計数
- Rowsread: アプリケーションが読み取った行の合計数
- Idle: アプリケーションがアイドルだった合計時間
- ET (経過時間): アプリケーションの現行作業単位が開始した (作業単位設定限度を超えた) ときからの経過時間

ここには標準値が書き込まれるので、ログ・ファイルを照会してさまざまなタイプの処置を見ることができます。それに対し *Message* フィールドには、 *RecType* フィールドの値によって変わるその他の非標準情報が入ります。たとえば、FORCE レコードまたは NICE レコードには *Message* フィールドのアプリケーション情報が示され、 ERROR レコードにはエラー・メッセージが入れられます。

ログ・ファイルの例を、次に示します。

```
1995-12-11 14.54.52    0 START      Database = TQTEST
1995-12-11 14.54.52    0 READCFG    Config = /u/db2instance/sqllib/tqtest.cfg
1995-12-11 14.54.53    0 ERROR      SQLMON Error: SQLCode = -1032
1995-12-11 14.54.54    0 ERROR      SQLMONSZ Error: SQLCode = -1032
```

関連概念:

- 303 ページの『ガバナー・ユーティリティー』
- 320 ページの『ガバナー・ログ・ファイルの照会』

ガバナー・ログ・ファイルの照会

各ガバナー・デーモンは、それぞれ固有のログ・ファイルに書き込みを行います。 db2govlg ユーティリティーを使用すると、ログ・ファイルの照会を行うことができます。単一パーティションあるいは全データベース・パーティションのログ・ファイルを、日時でソートしてリストすることができます。また *RecType* ログ・フィールドに基づいて、照会することも可能です。 db2govlg の構文は、次のとおりです。

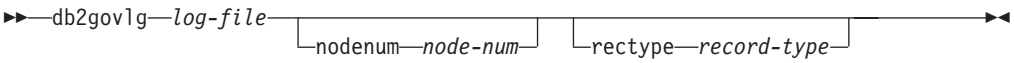


図 25. db2govlg の構文

パラメーターは、以下のとおりです。

log-file
照会したいログ・ファイル (複数も可) の基底名。

nodenum **node-num**
ガバナーが実行されているデータベース・パーティションのノード番号。

rectype **record-type**
照会したいレコードのタイプ。レコード・タイプは、次のとおりです。
- START
- READCFG
- STOP
- FORCE
- NICE
- ERROR
- WARNING
- ACCOUNT

このユーティリティーを使用するには、許可に関する制約事項はありません。したがって、すべてのユーザーが、ガバナーがユーザーのアプリケーションに影響を与えていないかどうかを照会することができます。このユーティリティーへのアクセスに制約を加えたい場合には、 db2govlg ファイルのグループ許可を変更することができます。

関連概念:
- 303 ページの『ガバナー・ユーティリティー』
- 316 ページの『ガバナー・ログ・ファイル』

第 10 章 構成のスケーリング

この章では、主にデータベース・パーティションを追加およびドロップすることによって、データベース容量を管理する方法を説明します。容量を大きくするには、他に CPU やメモリーを追加するなどの方法があります。

データベース・サーバー容量の管理

データベース・マネージャーの容量が、現在または将来の必要を満たさない場合、次のような方法でその容量を拡張することができます。

- ディスク・スペースを追加し、追加のコンテナーを作成します。
- メモリーを追加します。

これらの単純なストラテジーで、必要な容量を追加できない場合は、次の方法を考慮してください。

- プロセッサーを追加します。

  1 台のプロセッサーを単一パーティション構成で使用しており、しかもそれを最大限まで使用してしまっている場合、プロセッサーを追加するか、パーティションを追加したほうがよいでしょう。プロセッサーを追加することの利点は、処理力が増すことです。 SMP システムでは、プロセッサーはメモリーおよびストレージ・システム・リソースを共用します。すべてのプロセッサーが 1 つのシステムに収まっているため、システム間の通信回線、およびシステム間のタスクの調整などについての、オーバーヘッドとなる考慮事項が加わることがありません。 LOAD、バックアップおよびリストアなどの DB2® のユーティリティーは、追加のプロセッサーを利用することができます。 DB2 Universal Database™ はこの環境をサポートします。

  注: Solaris オペレーティング環境のように、オペレーティング・システムによっては、プロセッサーを動的にオンラインまたはオフラインに調整することができるものがあります。

  プロセッサーを追加する場合、使用されるプロセッサーの数を左右する、いくつかのデータベース構成パラメーターを検討し、変更してください。次のデータベース構成パラメーターは、使用するプロセッサー数を判別するもので、更新の必要の可能性があります。

  – デフォルトのパーティション内並行度 (dft_degree)
  – 最大並列処理の多重度 (max_querydegree)
  – パーティション内並列処理機能の使用可能化 (intra_parallel)

  また、アプリケーションが並列処理を実行する方法を決定するパラメーターを評価することも必要です。

  通信に TCP/IP が使用されている環境では、 DB2TCPCONNMGRS レジストリー変数の値を考慮する必要があります。

- 物理ノードを追加します。

  データベース・マネージャーが現在パーティション化されている場合、個別の単一プロセッサーまたはマルチプロセッサーの物理ノードを追加することによって、データ・ストレージ・スペースを広げ、処理能力を高めることができます。各ノード上のメモリーおよびストレージ・システム・リソースは、他のノードと共用されません。ノードの追加の結果、通信およびタスク調整の問題が発生することがありますが、この選択には、複数のシステム間でデータおよびユーザー・アクセスのバランスを取ることができる、という利点があります。 DB2 Universal Database はこの環境をサポートします。

  ノードの追加は、データベース・マネージャー・システムの実行中でも停止中でも行えます。ただし、システムの実行中にノードを追加すると、データベースがその新しいノードに移行する前にシステムを一度停止し、再始動することが必要です。

環境を変更してシステムを拡大または縮小するときは、そのような変更が、データのロード、データベースのバックアップおよびリストアなどのデータベース手順に与える影響をよく知っている必要があります。

新しいデータベース・パーティションを追加するときは、その処理が完了し、新しいサーバーが正常にシステムに組み込まれるまでは、新規パーティションを活用するデータベースのドロップまたは作成を行うことはできません。

関連概念:

- 322 ページの『パーティション・データベースのパーティション』

パーティション・データベースのパーティション

システムの稼働中、またはシステムの停止時に、パーティション・データベース・システムにデータベース・パーティションを追加することができます。新しいサーバーを追加するのは時間のかかる作業なので、これはデータベース・マネージャーがすでに実行しているときに行うこともできます。

ADD DBPARTITIONNUM コマンドを使用すると、システムにデータベース・パーティションを追加することができます。このコマンドは以下のようにして呼び出すことができます。

- `db2start` のオプションとして
- コマンド行プロセッサー ADD DBPARTITIONNUM コマンドを使って
- API 関数 `sqleaddn` を使って
- API 関数 `sqlepstart` を使って

システムが停止している場合、`db2start` を使用します。実行中の場合は、その他の選択肢のどれでも使用できます。

新規のデータベース・パーティションを ADD DBPARTITIONNUM コマンドを使用してシステムに追加すると、すでにインスタンスに入っているすべてのデータベースはその新規データベース・パーティションに拡張されます。システム管理者は、

データベース用の TEMPORARY 表スペースに使用するためのコンテナーを指定することもできます。このコンテナーは、以下のようにすることができます。

- 各データベースのカタログ・ノードに定義されるものと同じ (これはデフォルト)。
- 他のデータベース・パーティションに定義されているものと同じです。
- まったく作成しない。 ALTER TABLESPACE ステートメントを使用して、各データベースに TEMPORARY 表スペース・コンテナーを追加してからでないと、データベースを使用できません。

新規パーティション上のデータベースは、 1 つ以上のデータベース・パーティション・グループを変更して新規のデータベース・パーティションを含めるようにするまでは、データを入れるために使用できません。

単にご使用のシステムにパーティションを追加するだけでは、単一パーティション・システムを複数パーティション・システムに変更することはできません。パーティション間でデータを再分散するには、関係する各表にパーティション・キーが必要だからです。パーティション・キーは、複数パーティション環境で表が作成されたときに自動的に生成されます。単一パーティション環境の場合、パーティション・キーは CREATE TABLE または ALTER TABLE SQL ステートメントによって明示的に作成できます。

注: システムにデータベースが定義されておらず、 UNIX® ベースのシステムで Enterprise Server Edition を実行している場合、 `db2nodes.cfg` ファイルを編集して、新規データベース・パーティション定義を追加します。ここで説明されている手順はどれも使用してはなりません。適用されるのは、データベースが存在する場合だけです。

Windows NT についての考慮事項: Windows NT 上の Enterprise Server Edition を使用している場合、インスタンスにデータベースがなければ、 DB2NCRT コマンドを使ってデータベース・システムを拡大または縮小してください。しかし、データベースがあれば、DB2START ADDNODE コマンドを使用し、システムのサイズ変更時に既存のデータベースごとにデータベース・パーティションを確実に作成してください。 Windows NT では、ノード構成ファイル (`db2nodes.cfg`) を手作業で編集することは避けてください。そのようなことをすれば、ファイル内での整合性が失われるおそれがあります。

関連タスク:

- 324 ページの『実行中のデータベース・システムへのパーティションの追加』
- 325 ページの『Windows NT 上で停止中のデータベース・システムへのパーティションの追加』
- 330 ページの『データベース・パーティションのドロップ』

実行中のデータベース・システムへのパーティションの追加

システムが稼働していて、アプリケーションがデータベースに接続されている間に、パーティション・データベースのシステムに新しいデータベース・パーティションを追加することができます。しかし、新規に追加されたサーバーがすべてのデータベースに使用できるようになるのは、データベース・マネージャーがシャットダウンされて再び始動された後です。

手順:

データベース・パーティションを実行中のデータベース・マネージャーに追加するためには、次のようにします。

1. 既存のデータベース・パーティションで、DB2START コマンドを実行します。

   すべてのプラットフォームで、DBPARTITIONNUM、ADD DBPARTITIONNUM、HOSTNAME、PORT、および NETNAME パラメーターに対して新パーティション値を指定します。 Windows NT プラットフォームでは、COMPUTER、USER、および PASSWORD パラメーターも指定します。

   また、データベースに作成する必要のある任意の TEMPORARY 表スペース・コンテナー定義のためのソースを指定することもできます。表スペース情報を提供しないと、 TEMPORARY 表スペース・コンテナー定義は各データベースのカタログ・ノードから検索されます。

   この DB2START コマンドが完了すると、新しいサーバーは停止します。
2. DB2STOP コマンドを実行することによって、すべてのパーティション上のデータベース・マネージャーを停止します。

   システム内のすべてのデータベース・パーティションを停止すると、ノード構成ファイルが更新されて新規データベース・パーティションが組み込まれます。DB2STOP が実行されるまでは、ノード構成ファイルがこの新しいサーバー情報によって更新されることはありません。このため、ADD DBPARTITIONNUM コマンド (DB2START コマンドに ADDNODE パラメーターが指定されたときに呼ばれる) は確実に正しいデータベース・パーティションで実行されます。このユーティリティーが終了すると、新しいサーバー・パーティションは停止します。
3. DB2START コマンドを実行してデータベース・マネージャーを開始します。

   これで、新規に追加されたデータベース・パーティションが残りのシステムと共に開始されます。

   システム内のすべてのデータベース・パーティションが実行中になると、データベースの作成またはドロップなどのシステム全般にわたる活動を行うことができます。

   注: 新しい `db2nodes.cfg` ファイルにアクセスするには、すべてのデータベース・パーティション・サーバーに関して DB2START コマンドを 2 回発行しなければならない場合があります。

4. 新規データベース・パーティションのすべてのデータベースのバックアップをとる。 (オプション)
5. 新規データベース・パーティションにデータを再分散します。 (オプション)

関連概念:

- 322 ページの『パーティション・データベースのパーティション』

関連タスク:

- 325 ページの『Windows NT 上で停止中のデータベース・システムへのパーティションの追加』
- 327 ページの『UNIX 上で停止中のデータベース・システムへのパーティションの追加』

Windows NT 上で停止中のデータベース・システムへのパーティションの追加

パーティション・データベース・システムの停止時に、新しいデータベース・パーティションを追加することができます。新規に追加されたデータベース・パーティションがすべてのデータベースに利用可能になるのは、データベース・マネージャーを再び始動したときです。

前提条件:

パーティションを作成する前に、新しいサーバーをインストールすることが必要です。

手順:

停止中のパーティション・データベース・サーバーにパーティションを追加する方法は以下のとおりです。

1. DB2STOP を出して、すべてのデータベース・パーティションを停止します。
2. ADD DBPARTITIONNUM コマンドを新規サーバーで実行します。

   データベース・パーティションは、システムにすでにある各データベースに対してローカルに作成されます。新規データベース・パーティションのためのデータベース・パラメーターは、デフォルト値に設定されます。各データベース・パーティションは、ユーザーがそこにデータを移すまでは空のままです。データベース構成パラメーター値を更新して、他のデータベース・パーティション上の値と一致させてください。
3. DB2START コマンドを実行して、データベース・システムを開始します。ノード構成ファイル (.cfg) は、すでに新規のサーバーのインストール時に更新されており、その新規のサーバーは組み込まれています。
4. 新しいパーティションで構成ファイルを更新する方法は次のとおりです。
   a. 既存のデータベース・パーティションで、DB2START コマンドを実行します。

DBPARTITIONNUM、ADDDB2PARTITIONNUM、HOSTNAME、PORT、および NETNAME パラメーターに加え、COMPUTER、USER、および PASSWORD パラメーターに新パーティション値を指定します。

また、データベースに作成する必要のある任意の TEMPORARY 表スペース・コンテナー定義のためのソースを指定することもできます。表スペース情報を提供しないと、 TEMPORARY 表スペース・コンテナー定義は各データベースのカタログ・ノードから検索されます。

この DB2START コマンドが完了すると、新しいサーバーは停止します。

b. DB2STOP コマンドを実行することによってデータベース・マネージャー全体を停止します。

システム内のすべてのデータベース・パーティションを停止すると、ノード構成ファイルが更新されて新規データベース・パーティションが組み込まれます。 DB2STOP が実行されるまでは、ノード構成ファイルがこの新しいサーバー情報によって更新されることはありません。このため、ADD DB2PARTITIONNUM コマンド (DB2START コマンドに ADDDB2PARTITIONNUM パラメーターが指定されたときに呼ばれる) は確実に正しいデータベース・パーティションで実行されます。このユーティリティーが終了すると、新しいサーバー・パーティションは停止します。

5. DB2START コマンドを実行してデータベース・マネージャーを開始します。

これで、新規に追加されたデータベース・パーティションが残りのシステムと共に開始されます。

システム内のすべてのデータベース・パーティションが実行中になると、データベースの作成またはドロップなどのシステム全般にわたる活動を行うことができます。

注: 新しい `db2nodes.cfg` ファイルにアクセスするには、すべてのデータベース・パーティション・サーバーに関して DB2START コマンドを 2 回発行しなければならない場合があります。

6. 新規データベース・パーティションのすべてのデータベースのバックアップをとる。 (オプション)
7. 新規データベース・パーティションにデータを再分散します。 (オプション)

関連概念:

- 322 ページの『パーティション・データベースのパーティション』
- 329 ページの『ノード追加エラーのリカバリー』

関連タスク:

- 324 ページの『実行中のデータベース・システムへのパーティションの追加』
- 327 ページの『UNIX 上で停止中のデータベース・システムへのパーティションの追加』

UNIX 上で停止中のデータベース・システムへのパーティションの追加

パーティション・データベース・システムの停止時に、新しいデータベース・パーティションを追加することができます。新規に追加されたデータベース・パーティションがすべてのデータベースに利用可能になるのは、データベース・マネージャーを再び始動したときです。

前提条件:

サーバーが存在しない場合、以下のタスクも含め、新しいサーバーをインストールする必要があります。

- 実行可能モジュールをアクセス可能にする (共用ファイル・システム・マウントまたはローカル・コピーを使用する)
- オペレーティング・システム・ファイルを既存のプロセッサー上のシステム・ファイルと同期化する
- sqllib ディレクトリーがファイル共用システムとしてアクセス可能であることを確認する
- 関連するオペレーティング・システム・パラメーター (プロセスの最大数など) が適切な値に設定されていることを確認する

また、すべてのデータベース・パーティションにおいて、 etc ディレクトリーの hosts ファイルのネーム・サーバーにホスト名を登録することも必要です。

手順:

停止中のパーティション・データベース・サーバーにパーティションを追加する方法は以下のとおりです。

1. DB2STOP を出して、すべてのデータベース・パーティションを停止します。
2. ADD DB2PARTITIONNUM コマンドを新規サーバーで実行します。

   データベース・パーティションは、システムにすでにある各データベースに対してローカルに作成されます。新規データベース・パーティションのためのデータベース・パラメーターは、デフォルト値に設定されます。各データベース・パーティションは、ユーザーがそこにデータを移すまでは空のままです。データベース構成パラメーター値を更新して、他のデータベース・パーティション上の値と一致させてください。
3. DB2START コマンドを実行して、データベース・システムを開始します。ノード構成ファイル (.cfg) は、すでに新規のサーバーのインストール時に更新されており、その新規のサーバーは組み込まれています。
4. 新しいパーティションで構成ファイルを更新する方法は次のとおりです。
   a. 既存のデータベース・パーティションで、DB2START コマンドを実行します。

      DB2PARTITIONNUM、ADDDB2PARTITIONNUM、HOSTNAME、PORT、および NETNAME パラメーターに加え、COMPUTER、USER、および PASSWORD パラメーターに新パーティション値を指定します。

また、データベースに作成する必要のある任意の TEMPORARY 表スペース・コンテナー定義のためのソースを指定することもできます。表スペース情報を提供しないと、 TEMPORARY 表スペース・コンテナー定義は各データベースのカタログ・ノードから検索されます。

この DB2START コマンドが完了すると、新しいサーバーは停止します。

b. DB2STOP コマンドを実行することによってデータベース・マネージャー全体を停止します。

システム内のすべてのデータベース・パーティションを停止すると、ノード構成ファイルが更新されて新規データベース・パーティションが組み込まれます。 DB2STOP が実行されるまでは、ノード構成ファイルがこの新しいサーバー情報によって更新されることはありません。このため、ADD DB2PARTITIONNUM コマンド (DB2START コマンドに ADDDB2PARTITIONNUM パラメーターが指定されたときに呼ばれる) は確実に正しいデータベース・パーティションで実行されます。このユーティリティーが終了すると、新しいサーバー・パーティションは停止します。

5. DB2START コマンドを実行してデータベース・マネージャーを開始します。

これで、新規に追加されたデータベース・パーティションが残りのシステムと共に開始されます。

システム内のすべてのデータベース・パーティションが実行中になると、データベースの作成またはドロップなどのシステム全般にわたる活動を行うことができます。

注: 新しい db2nodes.cfg ファイルにアクセスするには、すべてのデータベース・パーティション・サーバーに関して DB2START コマンドを 2 回発行しなければならない場合があります。

6. 新規データベース・パーティションのすべてのデータベースのバックアップをとる。 (オプション)
7. 新規データベース・パーティションにデータを再分散します。 (オプション)

また、次のように構成ファイルを手動で更新することもできます。

1. 1. db2nodes.cfg ファイルを編集し、新規データベース・パーティションをそこに追加します。
2. 次のコマンドを出して、新しいノードを開始します。 `DB2START DB2PARTITIONNUM partitionnum`

新しいデータベース・パーティションに割り当てる番号を nodenum の値として指定します。

3. この新規サーバーを論理データベース・パーティション (すなわち、ノード 0 ではない) にする場合は、 db2set コマンドを使用して、DB2PARTITIONNUM レジストリー変数を更新します。追加するデータベース・パーティションの数を指定します。
4. ADD NODE コマンドを新規データベース・パーティションで実行します。

このコマンドは、システムにすでにある各データベースに対してローカルにデータベース・パーティションを作成します。新規データベース・パーティションの

ためのデータベース・パラメーターは、デフォルト値に設定されます。各データベース・パーティションは、ユーザーがそこにデータを移すまでは空のままです。データベース構成パラメーター値を更新して、他のデータベース・パーティション上の値と一致させてください。

5. ADD DB2PARTITIONNUM コマンドが完了したら、DB2START コマンドを出して、システム内の他のデータベース・パーティションを開始します。

   すべてのデータベース・パーティションが正常に開始するまでは、データベースの作成またはドロップなどのシステム全般にわたる活動を行わないでください。

関連概念:

- 329 ページの『ノード追加エラーのリカバリー』

関連タスク:

- 324 ページの『実行中のデータベース・システムへのパーティションの追加』
- 325 ページの『Windows NT 上で停止中のデータベース・システムへのパーティションの追加』
- 330 ページの『データベース・パーティションのドロップ』

ノード追加エラーのリカバリー

バージョン 8.1 以降では、DB2® は「隠れた」バッファー・プールを作成して、すべてのバッファー・プールのページ・サイズにデフォルト自動サポートを提供するので、ノードの追加がバッファー・プールが存在しないために失敗することはありません。しかし、これらの「隠れた」バッファー・プールのうち 1 つが使用されると、隠れたバッファー・プールが非常に小さいため、パフォーマンスが大きな影響を受けることがあります。隠れたバッファー・プールが使用される場合、管理通知ログにメッセージが書き込まれます。

隠れたバッファー・プールは、次の状況で、ノード追加のシナリオにおいて使用されます。

- デフォルト (4KB) と異なるページ・サイズで、 1 つ以上のシステム TEMPORARY 表スペースを持つ分割したデータベースにノードを追加する場合。ノードの作成時には IBMDEFAULTDP バッファー・プールだけしか存在せず、このバッファー・プールのページ・サイズが 4KB です。

  次の例を考慮してください。

  1. 現在の分割したデータベースにノードを追加するため、db2start コマンドを使用します。

     ```
     DB2START DB2PARTITIONNUM 2 ADD DB2PARTITIONNUM HOSTNAME newhost PORT 2
     ```

  2. 新しいノード記述で、db2nodes.cfg ファイルを手動で更新した後に、 ADD DB2PARTITIONNUM コマンドを使用します。

  これらの問題を予防する 1 つの方法としては、ADD NODE コマンドや **db2start** コマンドで WITHOUT TABLESPACES 文節を指定する方法があります。コマンド実行後、CREATE BUFFERPOOL ステートメントを使用して、バッファー・プールを作成し、ALTER TABLESPACE ステートメントを使用して、バッファー・プールにシステム TEMPORARY 表スペースを作成する必要があります。

- デフォルト・ページ・サイズ (4KB) と異なるページ・サイズで、 1 つ以上の表スペースを持つ既存のデータベース・パーティション・グループにノードを追加する場合。これは、デフォルトでないページ・サイズ・バッファー・プールを、表スペースとしてアクティブになっていない、新規のノード上に作成した場合に発生します。

  注: DB2 の旧バージョンでは、このコマンドは、DATABASE PARTITION GROUP キーワードではなく NODEGROUP キーワードを使用していました。

  次の例を考慮してください。

  – データベース・パーティション・グループにノードを追加するために ALTER DATABASE PARTITION GROUP ステートメントを次のように使用します。

    ```
    DB2START
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
    ```

    この問題を予防する 1 つの方法としては、それぞれのページ・サイズのバッファー・プールを作成し、その後、ALTER DATABASE PARTITION GROUP ステートメントを発行する前にデータベースに再接続する方法があります。

    ```
    DB2START
    CONNECT TO mpp1
    CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
    CONNECT RESET
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
    ```

注: デフォルトのページ・サイズの表スペースを持つデータベース・パーティション・グループの場合は以下のようなメッセージが返されます。

```
SQL1759W Redistribute nodegroup is required to change data
      positioning for objects in nodegroup "ng1" to include some
      added nodes or exclude some drop nodes.
```

関連タスク:

- 324 ページの『実行中のデータベース・システムへのパーティションの追加』
- 325 ページの『Windows NT 上で停止中のデータベース・システムへのパーティションの追加』

データベース・パーティションのドロップ

どのデータベースにも使用されていないデータベース・パーティションをドロップして、他の使用のためにコンピューターを解放することができます。

前提条件:

DROP NODE VERIFY コマンドまたは *sqledrpn* API を使用して、そのパーティションが使用中ではないことを確認します。

- メッセージ SQL6034W (ノードはほかのデータベースによって使用されていません - Node not used in any database) が出た場合は、パーティションをドロップできます。

- メッセージ SQL6035W (ノードはデータベースによって使用中です - Node in use by database) が出た場合は、 REDISTRIBUTE NODEGROUP コマンドを使用して、ドロップするデータベース・パーティションのデータを、データベース別名から他のデータベース・パーティションへ再分散します。

また、このデータベース・パーティションをコーディネーターとしていたすべてのトランザクションがすべて正常にコミットされている、またはロールバックされていることを確認してください。このためには、他のサーバーでクラッシュ・リカバリーを行う必要があることがあります。たとえば、(コーディネーター・ノードのような) コーディネーター・データベース・パーティションをドロップすると、そのコーディネーター・ノードがドロップされる前にトランザクションに関連する他のデータベース・パーティションが破損した場合、破損したデータベース・パーティションはどの未確定トランザクションの結果についてもコーディネーター・ノードを照会することができなくなります。

手順:

データベース・パーティションをドロップする方法は次のとおりです。

1. DROP NODENUM パラメーターを指定した DB2STOP コマンドを出して、データベース・パーティションをドロップします。このコマンドが正常に終了すると、システムは停止します。
2. DB2START コマンドを実行してデータベース・マネージャーを開始します。

関連概念:

- 321 ページの『データベース・サーバー容量の管理』
- 322 ページの『パーティション・データベースのパーティション』

第 11 章 データベース・パーティション間でのデータの再分散

この章では、パーティション間のデータの再分散時の決定、再分散の実行方法、および再分散エラーからのリカバリー方法について説明します。

データの再分散

パーティション・データベース内のパーティション間で表データを再分散するには、 REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用します。

注: DB2® の以前のバージョンでは、このコマンドは DATABASE PARTITION GROUP キーワードではなく NODEGROUP キーワードを使用していました。

パーティション・データベースでは、次の理由でデータの再分散を行うことができます。

- データベース・パーティション間のデータ量と処理負荷のバランスを調整します。

  データ・アクセスを複数のパーティションに広げることができると、パフォーマンスが向上します。
- データベース・パーティション間のデータ分散にスキューを導入します。

  アクセスの頻度が多い表のデータを再分散して、アクセスの頻度が少ないデータはデータベース・パーティション・グループ内の少数のデータベース・パーティションに、また、アクセスの頻度が多いデータは多数のパーティションにわたって分散するようにすると、アクセスおよびスループット・パフォーマンスが向上する可能性があります。こうすることによって、実行頻度が最も多いアプリケーションにおけるアクセス・パフォーマンスとスループットが向上します。

表コロケーションを保存するには、 REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用して、データベース・パーティション・グループ・レベルでデータを再分散します。単一の操作ですべての表が再分散されます。指定されたデータ再分散を実行するために、 REDISTRIBUTE DATABASE PARTITION GROUP コマンドは、データベース・パーティション間で行を移動し、表を分割します。指定したオプションによって、ユーティリティーはターゲットパーティション・マップを生成する、または既存のパーティション・マップを入力として使用することができます。

データベース・パーティション間でのデータの再分散方法

データの再分散は、データベースの指定されたデータベース・パーティション・グループ内の表集合に対して実行されます。 REDISTRIBUTE DATABASE PARTITION GROUP コマンドを実行してデータ再分散ユーティリティーを呼び出す前に、カタログ・データベース・パーティションでデータベースに接続する必要があります。ユーティリティーはソースパーティション・マップとターゲットパーティション・マップの両方を使用して、どのハッシュ・パーティションが新しいロケ

ーション (新しいデータベース・パーティション番号) に割り当てられているのかを識別します。新しいロケーションに割り当てられたパーティションに対応する行はすべて、ソースパーティション・マップに指定されたデータベース・パーティションからターゲットパーティション・マップに指定されたデータベース・パーティションに移動されます。

データ再分散ユーティリティーは、以下のステップを実行します。

1. ターゲットパーティション・マップの新しいパーティション・マップ ID を取得し、その ID を SYSCAT.PARTITIONMAP カタログ表に挿入します。
2. データベース・パーティション・グループの SYSCAT.DBPARTITIONGROUPS カタログ・ビュー内の REBALANCE_PMAP_ID 列を、新しいパーティション・マップ ID で更新します。
3. 任意の新しいデータベース・パーティションを SYSCAT.DBPARTITIONGROUPDEF カタログ・ビューに追加します。
4. ドロップしたいデータベース・パーティションがあればそのパーティションすべてについて、 SYSCAT.DBPARTITIONGROUPDEF カタログ・ビュー内の IN_USE 列を D に設定します。
5. カタログの更新を COMMIT します。
6. 新しいデータベース・パーティションすべてに対して、データベース・ファイルを作成します。
7. 次のステップ中のすべての表に対してそれぞれの表を基本とするデータを再分散します。
   a. SYSTABLES カタログ表にあるその表の行をロックします。
   b. この表に関連するパッケージすべてを無効にします。表に関連付けられたパーティション・マップ ID は、表行が再分散されるので変更されます。パッケージが無効にされたので、コンパイラーは表に関する新しいパーティション情報を取得して、それに応じてパッケージを生成する必要があります。
   c. 排他モードで表をロックします。
   d. DELETE および INSERT を使用して、表のデータを再分散します。
   e. 再分散操作が成功すると、表に対して COMMIT を発行し、データベース・パーティション・グループ内の次の表について操作を継続します。表が完全に再分散される前に操作が失敗した場合は、ユーティリティーは表への更新に対して ROLLBACK を発行し、再分散操作全体を終了してエラーを戻す。
8. データベース・ファイルを削除し、 SYSCAT.NODEGROUPDEF カタログ表の中から前にドロップするものとしてマークを付けたデータベース・パーティションの項目を削除します。
9. SYSCAT.NODEGROUPS カタログ・ビュー内のデータベース・パーティション・グループ・レコードを更新して、 PMAP_ID を REBALANCE_PMAP_ID の値に、REBALANCE_PMAP_ID を NULL に設定します。
10. SYSCAT.PARTITIONMAPS カタログ・ビューから古いパーティション・マップを削除します。
11. すべての変更を COMMIT します。

関連概念:

- 338 ページの『データ再分散のログ・スペース所要量』
- 339 ページの『再分散エラー・リカバリー』

関連タスク:

- 336 ページの『パーティション間でのデータの再分散』
- 335 ページの『データ再分散の決定』

データ再分散の決定

データを再分散することを決定する前に、データがパーティション間で不均等に分散しているかを確認してください。現行の分散情報を入手したなら、その情報を使用してカスタム再分散ファイルまたはパーティション・マップを作成することができます。

手順:

データベース・パーティション・グループ内のパーティション間での、現行のデータ分散について情報を得るには、次のようにします。

1. 行が不均等に分散されているデータベース・パーティションがあるか判別します。

   最大の表に対して、該当するパーティション列を使用して次のような照会を入力します。

   ```
   SELECT PARTITION(column_name), COUNT(*) FROM table_name
      GROUP BY PARTITION(column_name)
      ORDER BY PARTITION(column_name) DESC
      FETCH FIRST 100 ROWS ONLY
   ```

   PARTITION および DBPARTITIONNUM SQL 関数は、ハッシュ・パーティション間またはデータベース・パーティション間での現行のデータ分散を判別します。 PARTITION 関数は、表の行ごとについてパーティション・マップ索引を戻します。 DBPARTITIONNUM 関数は、行のパーティション番号を戻します。
2. データベース・パーティション・グループにわたってパーティション化されている他の大規模な表に対して、この照会を実行します。
3. この情報を使用して、分散ファイルおよびターゲットパーティション・マップの両方を作成します。

注: また、 ANALYZE オプションを指定した AutoLoader ユーティリティーを使用しても、データ分散ファイルを作成することができます。このファイルは、データ再分散ユーティリティーへの入力として使用することが可能です。

関連概念:

- 333 ページの『データの再分散』
- 338 ページの『データ再分散のログ・スペース所要量』

関連タスク:

- 336 ページの『パーティション間でのデータの再分散』

パーティション間でのデータの再分散

パーティション・データベースでは、以下のような場合にパーティション間でデータを再分散し、データ・アクセスの平衡を取ることができます。

- いくつかのパーティションが他のパーティションより多くのデータを含んでいる場合
- いくつかのパーティションが他のパーティションよりアクセスの頻度が高い場合

前提条件:

ログ・ファイルのサイズ: ログ・ファイルがデータ再分散操作に応じた十分な大きさであることを確認してください。影響を受けるパーティションごとのログ・ファイルは、そこで実行される INSERT および DELETE 操作を受け入れるのに十分な大きさでなければなりません。

複製されたマテリアライズ照会表: データベース・パーティション・グループ内のデータに複製されたマテリアライズ照会表が含まれている場合、データを再分散する前にこれらの表をドロップする必要があります。データの再分散が完了した後に、マテリアライズ照会表を再作成することができます。

制約事項:

ユーティリティーの実行中に、データベース・パーティション・グループのオブジェクトに対して以下に示す操作を行うことができます。ただし、再分散中の表に対してはそれらの操作を実行することはできません。行うことができる操作は、以下のとおりです。

- 他の表に対する索引の作成。 CREATE INDEX ステートメントは、影響される表のパーティション・マップを使用します。
- 他の表のドロップ。 DROP TABLE ステートメントは、影響される表のパーティション・マップを使用します。
- 他の表に対する索引のドロップ。 DROP INDEX ステートメントは、影響される表のパーティション・マップを使用します。
- 他の表の照会。
- 他の表の更新。
- データベース・パーティション・グループで定義された表スペースで新しい表を作成します。 CREATE TABLE ステートメントは、ターゲットパーティション・マップを使用します。
- データベース・パーティション・グループでの表スペースの作成。

ユーティリティーの実行中に行うことができない操作は、以下のとおりです。

- そのデータベース・パーティション・グループ上での別の再分散操作の開始
- そのデータベース・パーティション・グループ内の任意の表に対する ALTER TABLE ステートメントの実行
- データベース・パーティション・グループのドロップ
- データベース・パーティション・グループの変更

影響を受けるすべての表がパーティション・キーを持っているのでない限り、パーティションを単一パーティション・システムに追加した後、この手順を使用してデ

ータを再分散することはできません。 REDISTRIBUTE DATABASE PARTITION GROUP コマンドは、データを再分散するためのパーティション・キーに基づいています。パーティション・キーは、マルチパーティション・データベースのパーティション・グループ内に表が作成されるときに自動的に生成するか、あるいは CREATE TABLE または ALTER TABLE SQL ステートメントを使用して明示的に定義することができます。単一パーティションのパーティション・グループ内に表が作成されており、 CREATE TABLE SQL ステートメント内にパーティション・キーを定義しなかった場合、定義されるパーティション・キーはありません。データを再分散する前に、ALTER TABLE SQL ステートメントを使用して、関係するそれぞれの表のパーティション・キーを作成しなければなりません。

手順:

データベース・パーティション・グループ内のパーティション間でデータを再分散するには、次のようにします。

1. システム・カタログ表を含むデータベース・パーティションに接続します。
2. 必要に応じて、前提条件タスクを実行します。
3. REDISTRIBUTE DATABASE PARTITION GROUP コマンドを出す。

   注: DB2 の旧バージョンでは、このコマンドは、DATABASE PARTITION GROUP キーワードではなく NODEGROUP キーワードを使用していました。

   以下の引き数を指定します。

   データベース・パーティション・グループ名
   データをその中に再分散する、データベース・パーティション・グループを指定する必要があります。

   UNIFORM
   データを均等に分散し、その後も均等に分散されたままにしておく場合は、 UNIFORM を指定するか、分散タイプの引き数をいずれも省略します。 UNIFORM がデフォルトです。

   USING DISTFILE distfile-name
   データ・スキューを訂正または作成するカスタム分散を指定するには、分散ファイル名を含めます。データ再分散ユーティリティーは、このファイルを使用してターゲット・パーティション・マップを構成します。

   USING TARGETMAP targetmap-name
   データ再分散ユーティリティーは、指定されたターゲット・マップを直接使用します。

   詳細については、 REDISTRIBUTE DATABASE PARTITION GROUP コマンド行ユーティリティーの情報を参照してください。
4. 再分散が完了した後は、次のことを行ってください。
   - 再分散の前にドロップした、複製されたマテリアライズ照会表を再作成します。
   - RUNSTATS コマンドを実行してデータ分散統計を収集し、 SQL コンパイラーおよびオプティマイザーが照会のデータ・アクセス・プランを選択するときにそれを使用できるようにします。

注: Explain 表には、データの再分散に使用されるパーティション・マップについての情報が含まれています。

関連概念:

- 333 ページの『データの再分散』
- 338 ページの『データ再分散のログ・スペース所要量』
- 339 ページの『再分散エラー・リカバリー』

関連タスク:

- 335 ページの『データ再分散の決定』

データ再分散のログ・スペース所要量

パーティション間でデータを再分散する前に、ログ・スペース所要量を考慮してください。

ログは、データを再分散する各データベース・パーティションごとの INSERT 操作および DELETE 操作を受け入れるのに十分の大きさでなければなりません。ロギング所要量は、最多のデータを失うデータベース・パーティション、または最多のデータを獲得するデータベース・パーティションで最も大きくなります。

より多くのデータベース・パーティションに移動しようとしている場合は、現行データベース・パーティションの、新しいデータベース・パーティション数に対する比率を使用して、 INSERT 操作および DELETE 操作の数を見積もってください。たとえば、再分散前に均等に分散されているデータを再分散することについて考えてください。 4 つのデータベース・パーティションから 5 つのデータベース・パーティションへの移動の場合、元の 4 つのデータベース・パーティションの約 20 % のデータが、新規データベース・パーティションに移動します。これはつまり、DELETE 操作の 20 % が元のデータベース・パーティションごとに対して行われ、INSERT 操作のすべてが新規データベース・パーティションで行われることを意味します。

パーティション・キーに多くの NULL 値が含まれているといった、データが不均等に分散されている場合について考えてください。この場合、パーティション・キーに NULL 値を含むすべての行が、以前のパーティション方式によるあるデータベース・パーティションから、新規のパーティション方式による別のデータベース・パーティションに移動します。その結果、これら 2 つのデータベース・パーティションで必要とされるログ・スペースの量が増え、おそらく、均等な分散を想定した場合の計算量を超えることになります。

各表の再分散は単一トランザクションです。そのため、ログ・スペースを見積もるときは、 20 % などの変更のパーセントに最大の表のサイズを乗算します。ただし、最大の表は均等に分散されているが、たとえば 2 番目に大きな表には 1 つ以上の膨張したデータベース・パーティションが存在する場合もあることを考慮してください。そのような場合には、最大の表の代わりに、不均等に分散されている表を使用することを考慮してください。

注: データベース・パーティションで挿入および削除されるデータの最大量を見積もった後、その見積もりを 2 倍して、アクティブ・ログのピーク・サイズを判

別します。この見積もりがアクティブ・ログ限界の 256 GB を超える場合、複数のステップでデータ再分散を行わなければなりません。「makepmap」ユーティリティーを使用して、各ステップに対して 1 つの、一連のターゲット・パーティション・マップを生成してください。また、 *logsecond* データベース構成パラメーターを -1 に設定して、大部分のログ・スペースの問題を回避することもできます。

関連概念:

- 333 ページの『データの再分散』

再分散エラー・リカバリー

再分散操作の実行が開始されると、ファイルが sqllib ディレクトリーの redist サブディレクトリーに書き込まれます。この状況ファイルには、データベース・パーティション上で行われた操作すべて、再分散された表の名前 (複数の場合もある)、および操作の完了状況がリストされます。ある表が再分散できなかった場合には、その表の名前と該当する SQLCODE がファイルにリストされます。入力パラメーターが正しくないために再分散操作が開始できない場合には、ファイルは作成されず、SQLCODE が戻されます。

このファイルの命名規則は、次に示すとおりです。

UNIX® プラットフォーム:
データベース名.データベース・パーティション・グループ名.タイムスタンプ
UNIX 以外のプラットフォーム:
データベース名¥データベース・パーティション・グループ名¥日付¥時刻

注: UNIX 以外のプラットフォームでは、データベース・パーティション・グループ名の最初の 8 バイトのみが使用されます。

データの再分散操作が失敗した場合には、一部の表は再分散されたが、一部の表は再分散されなかったということが起こる可能性があります。これは、データの再分散は一度に 1 つの表に対してのみ行われるために起こります。リカバリーするには、次に示す 2 つの方法を取ることができます。

- CONTINUE オプションを使用して操作を続行して、残りの表の再分散を行う。
- ROLLBACK オプションを使用して再分散を取り消して、再分散された表を元の状態に設定し直す。ロールバック操作には、再分散操作でかかった時間とほぼ同じくらいの時間が必要になります。

いずれのオプションを使用する場合であっても、使用する前に、SYSCAT.DBPARTITIONGROUPS 表の REBALANCE_PMID 列が非 NULL 値に設定されていることをチェックして、直前に行ったデータ再分散操作が確かに失敗していることを確認する必要があります。

このファイルを誤って削除してしまった場合であっても、 CONTINUE 操作を試行することができます。

関連概念:

- 333 ページの『データの再分散』

関連タスク:

- 336 ページの『パーティション間でのデータの再分散』

再分散ストアード・プロシージャーおよび関数

ステップ単位の再分散ストアード・プロシージャーは、データベース・パーティション・グループを、いくつかのステップで安全に再分散するのに使用できます。

1. ログ・スペースの可用性およびデータ・スキューに関してデータベース・パーティション・グループを分析します。
2. 指定した表のデータ分散ファイルを作成します。
3. データベース・パーティション・グループのステップ単位再分散プランの内容を作成し、報告します。
4. プランに従って、データベース・パーティション・グループを再分散します。

以下のプロシージャーで、個別のパラメーターを扱っている場合、値を取得できないときは、パラメーターの出力値に「-1」が使用されます。

注: 再分散ストアード・プロシージャーおよび関数は、それぞれの表にパーティション・キーが定義されているパーティション・データベースでのみ使用可能です。

get_swrd_settings ストアード・プロシージャー

get_swrd_settings 関数は、指定したデータベース・パーティション・グループの既存の再分散レジストリー・レコードを読み取ります。

表 31. get_swrd_settings 入力パラメーター

名前	データ・タイプ	説明
dbpgName	VARCHAR(128)	再分散プロセスが実行される、データベース・パーティション・グループ名。
matchingSpec	SMALLINT	表 32 のビット単位フィールド ID で、 341 ページの表 33 の出力パラメーターによって返されるターゲット・フィールドを示します。これらの必要ない出力パラメーターは、NULL 設定できます。 たとえば、matchingSpec が (REDIST_STAGE_SIZE | REDIST_NEXT_STEP) の整数値である 96 に設定されている場合、この関数の呼び出し元は、値を受け取るために stageSize および nextStep を提供する必要があるだけで、出力パラメーターの残りは NULL にできます。

表 32. ビット単位フィールド ID

フィールド名	16 進値	10 進値
REDIST_METHOD	0x0001<<0	1
REDIST_PMAP_FILE	0x0001<<1	2
REDIST_DIST_FILE	0x0001<<2	4
REDIST_STEP_SIZE	0x0001<<3	8
REDIST_NUM_STEPS	0x0001<<4	16
REDIST_STAGE_SIZE	0x0001<<5	32
REDIST_NEXT_STEP	0x0001<<6	64

表 32. ビット単位フィールド ID (続き)

フィールド名	16 進値	10 進値
REDIST_PROCESS_STATE	0x0001<<7	128
REDIST_PWEIGHT_START_NODE	0x0001<<8	256
REDIST_PWEIGHT	0x0001<<9	512

表 33. get_swrd_settings 出力パラメーター

名前	データ・タイプ	説明
redistMethod	SMALLINT	分散ファイルまたはターゲット・パーティション・マップを使用して再分散が実行されることを示す数。
pMapFile	VARCHAR(255)	ターゲット・パーティション・マップの絶対パス・ファイル名。
distFile	VARCHAR(255)	データ分散ファイルの絶対パス・ファイル名。
stepSize	BIGINT	ログがフルになる状態を避けるために、コミットが呼び出される前に移動できる、行の最大数。数は、各分散ステップで移動できます。
totalSteps	SMALLINT	指定したデータベース・パーティション・グループを、完全に再分散するのに必要なステップの数。
stageSize	SMALLINT	連続して実行されるステップの数。
nextStep	SMALLINT	完了したステップおよび、実行する必要のあるステップを区切る索引。
processState	SMALLINT	nextStep で再分散ステージを停止するのにユーザーが設定できるフラグ。
pNumber	VARCHAR(6000)	パーティション重みに対応するデータベース・パーティション・グループ内のすべてのパーティション番号で移植される事前割り振りストリング。ストリング内のパーティション番号は「,」で区切られています。
pWeight	VARCHAR(6000)	SET_SWRD_SETTINGS ストアード・プロシージャーで指定される各パーティション上のボリュームのすべての相対重みで移植される事前割り振りストリング。ストリング内のパーティション重みは「,」で区切られています。

set_swrd_settings ストアード・プロシージャー

set_swrd_settings 関数は、再分散レジストリーを作成または変更します。レジストリーが存在しない場合は作成され、レコードが追加されます。レジストリーがすでに存在する場合は、overwriteSpec を使用して上書きされる必要のあるフィールド値を識別します。 overwriteSpec フィールドはこの関数を使用可能にして、更新される必要のないフィールドに NULL 入力を指定します。

このプロシージャーでは、数が無制限であることを示すのに、stepSize および totalSteps に値「-2」を使用することができます。

表 34. set_swrd_settings 入力パラメーター

名前	データ・タイプ	説明
dbpgName	VARCHAR(128)	再分散プロセスが実行される、データベース・パーティション・グループ名。

表 34. set_swrd_settings 入力パラメーター (続き)

名前	データ・タイプ	説明
overwriteSpec	SMALLINT	340 ページの表 32 のビット単位フィールド ID で、再分散上書きレジストリーに書き込み、または上書きされるターゲット・フィールドを示します。
redistMethod	SMALLINT	分散ファイルまたはターゲット・パーティション・マップを使用して再分散が実行されることを示す数。
pMapFile	VARCHAR(255)	ターゲット・パーティション・マップの絶対パス・ファイル名。
distFile	VARCHAR(255)	データ分散ファイルの絶対パス・ファイル名。
stepSize	BIGINT	ログがフルになる状態を避けるために、コミットが呼び出される前に移動できる、行の最大数。数は、各分散ステップで移動できます。
totalSteps	SMALLINT	指定したデータベース・パーティション・グループを、完全に再分散するのに必要なステップの数。
stageSize	SMALLINT	連続して実行されるステップの数。
nextStep	SMALLINT	完了したステップおよび、実行する必要のあるステップを区切る索引。
processState	SMALLINT	nextStep で再分散ステージを停止するのにユーザーが設定できるフラグ。
pNumber	VARCHAR(6000)	パーティション重みに対応したすべてのパーティション番号を含むストリング。各パーティション番号は 0 から 999 で、番号は「,」で区切られています。ストリング内にスペースは使用できません。
pWeight	VARCHAR(6000)	pNumber ストリング内のパーティション番号に対応する、ユーザーが指定したすべてのパーティション重みを含むストリング。各パーティション重みは 0 から SQL_MAXSMALLVAL の番号で、番号は「,」で区切られています。ストリング内にスペースは使用できません。

analyze_log_space ストアード・プロシージャー

analyze_log_space 関数は、ログ・スペース分析の結果セット (オープン・カーソル) を返します。この中には、指定されたデータベース・パーティション・グループの各データベース・パーティションについて、以下のフィールドが含まれます。

表 35. analyze_log_space フィールド

列名	列タイプ	説明
PARTITION_NUM	SMALLINT	ログ・スペース分析のパーティション番号
TOTAL_LOG_SIZE	BIGINT	割り振られたログ・スペースの合計 (バイト単位)。-1 はサイズが無制限であることを示します。
AVAIL_LOG_SPACE	BIGINT	空いていて、再分散プロセスに使用できるログ・スペースの量 (バイト単位)
DATA_SKEW	BIGINT	ターゲット・レベルからはずれたデータのサイズの絶対値 (バイト単位)
REQ_LOG_SPACE	BIGINT	所要のデータ分散に達するのに必要なスペースの量 (バイト単位)
NUM_OF_STEPS	SMALLINT	データ・スキューをゼロまで減らすのに必要なステップの数

表 35. analyze_log_space フィールド (続き)

列名	列タイプ	説明
MAX_STEP_SIZE	BIGINT	ログ・フル・エラーを起こさずに一度に移動できる、データの最大量 (バイト単位)

表 36. analyze_log_space 入力パラメーター

名前	データ・タイプ	説明
inDBPGroup	VARCHAR(128)	データベース・パーティション・グループ名
inMainTbSchema	VARCHAR(128)	メイン表のスキーマ
inMainTable	VARCHAR(128)	データベース・パーティション・グループ内のメイン表 (dbpg 内最大の表である場合が多い)
useTbType	SMALLINT	分析タイプのインディケーター: SWRD_USE_STMG_TABLE 1: パーティションごとの行数を検出する、ストレージ管理表の情報の使用を示す。 SWRD_USE_REALTIME_ANALYSIS 2: パーティションごとの行数を検出する、選択照会の使用を示す。
inStmgTime	VARCHAR(26)	ストレージ管理レコードのタイム・スタンプ。このパラメーターは、 analysisType が SWRD_USE_REALTIME_ANALYSIS に設定されている場合は無視されます。
addDropOption	CHAR(1)	パーティションの追加またはドロップのインディケーター: 'A' パーティションの追加 'D' パーティションのドロップ 'N' 追加またはドロップなし
addDropList	VARCHAR(6000)	追加またはドロップされるパーティションのリスト (コンマで区切られたストリング)
pNumber	VARCHAR(6000)	パーティション重みに対応したすべてのパーティション番号を含むストリング。各パーティション番号は 0 から 999 で、番号は「,」で区切られています。ストリング内にスペースは使用できません。
pWeight	VARCHAR(6000)	pNumber ストリング内のパーティション番号に対応する、ユーザーが指定したすべてのパーティション重みを含むストリング。各パーティション重みは 0 から SQL_MAXSMALLVAL の番号で、番号は「,」で区切られています。ストリング内にスペースは使用できません。

generate_Distfile ストアード・プロシージャー

generate_Distfile 関数は、指定された表のデータ分散ファイルを生成し、指定された fileName の下に保管します。

表 37. generate_Distfile、入力パラメーター

名前	データ・タイプ	説明
inTbSchema	VARCHAR(128)	表スキーマ名
inTbName	VARCHAR(128)	表名

表 38. *generate_Distfile*、入出力パラメーター

名前	データ・タイプ	説明
fileName	VARCHAR(255)	データ分散ファイル名。指定されたファイル名が、通常のファイル名の場合、ファイルは instance/tmp ディレクトリーに保管され、完全ファイル・パス名がパラメーターによって返されます。

stepwise_redistribute_dbpg ストアード・プロシージャー

stepwise_redistribute_dbpg 関数は、入力および設定ファイルに従って、データベース・パーティション・グループの一部を再分散します。

ステップ 1. 設定レジストリーは、inDbPGroup 名を使用して検索されます。

- レジストリーが検出されなかった場合、エラーが返されます。
- レジストリーが検出された場合、以下の値が読み取られます。
  - 現行ステップ
  - ステップの数
  - パーティション・マップ名
  - 最大データ・サイズ (各ステップで移動可能)
  - プロセスの状態
  - パーティションの重み

  inNumSteps に -1 (SWRD_UNLIMITED_STEPS) を入力することで、データベース・パーティション・グループをすべて同時に再分散できます。

ステップ 2. レジストリー処理状態が SWRD_STOP の場合、処理は停止され、警告メッセージが返されます。処理状態が SWRD_CONTINUE の場合、処理は継続されます。

ステップ 3. inStartingPoint が NULL でなく有効な場合、対応するレジストリー値は上書きされます。そうでない場合、現行ステップのレジストリー値が読み取られ、このステップの開始点として使用されます。

ステップ 4. 次に現行ステップのパーティション・マップが生成され、パーティション重みは設定レジストリーから検索されます。既存の完全な パーティション・マップまたは distfile がある場合は、それに合わせてパーティション・マップが生成されます。

ステップ 5. 再分散 API は、パーティション・マップ・オプションを使用して呼び出されます。プロセスが完了すると、次のステップのレジストリー値が、増加します。

ステップ 1 から 5 は、レジストリーに指定されたステップ数繰り返されます。

このプロシージャーでは、数が無制限であることを示すのに、値「-2」を使用することができます。

表 39. *stepwise_redistribute_dbpg* 入力パラメーター

名前	データ・タイプ	説明
inDBPGroup	VARCHAR(128)	ターゲット・データベース・パーティション・グループの名前

表 39. *stepwise_redistribute_dbpg* 入力パラメーター (続き)

名前	データ・タイプ	説明
inStartingPoint	SMALLINT	このパラメーターは、NULL にすることができます。NULL でなく、正数を指している場合は、swrd 設定レジストリーに指定された「nextStep」値を上書きします。これは、特定のステップから SWRD を再実行したい場合に便利なオプションです。
inNumSteps	SMALLINT	実行するステップの数。NULL でなく、正数を指している場合は、swrd 設定レジストリーに指定された「numSteps」値を上書きします。これは、SWRD を設定に指定されているのと違うステップ数で再実行したい場合に、便利なオプションです。たとえば、スケジュールされたステージに 5 つのステップがあり、 SWRD プロセスがステップ 3 で失敗した場合、エラー状態を訂正したあとで、残りの 3 つのステップを実行するのに SWRD を呼び出すことができます。

db_partitions UDF

db_partitions ユーザー定義関数は db2nodes.cfg ファイルを構文解析し、検出されたパーティションごとに 1 行を返します。

入力パラメーター: なし

表 40. *db_partitions* 出力パラメーター

名前	データ・タイプ	説明
PARTITION_NUMBER	SMALLINT	パーティション番号
HOST_NAME	VARCHAR(128)	ホスト名 (Intel プラットフォームでは、これはマシン名です)
PORT_NUMBER	SMALLINT	論理ポート番号
SWITCH_NAME	VARCHAR(128)	ネット・スイッチ名 (NULL でも可)

使用例

以下に示すのは、AIX での CLP スクリプトの例です。

```
# ------------------------------------------------------------------------------
# Set the database you wish to connect to
# ------------------------------------------------------------------------------
dbName="SAMPLE"

# ------------------------------------------------------------------------------
# Set the target database partition group name
# ------------------------------------------------------------------------------
dbpgName="IBMDEFAULTGROUP"

# ------------------------------------------------------------------------------
# Specify the table name and schema
# ------------------------------------------------------------------------------
tbSchema="$USER"
tbName="STAFF"

# ------------------------------------------------------------------------------
# Specify the name of the data distribution file
# ------------------------------------------------------------------------------
distFile="$HOME/sqllib/function/$dbName.IBMDEFAULTGROUP_swrdData.dst"

export DB2INSTANCE=$USER
```

```
export DB2COMM=TCPIP

# ------------------------------------------------------------------------------------
# Invoke call statements in clp
# ------------------------------------------------------------------------------------
db2start
db2 -v "connect to $dbName"

# ------------------------------------------------------------------------------------
# Analysing the effect of adding a partition without applying the changes - a 'what if'
# hypothetical analysis
#
# - In the following case, the hypothesis is adding partition 40, 50 and 60 to the
#   database partition group, and for partitions 10,20,30,40,50,60, using a respective
#   target ratio of 1:2:1:2:1:2.
#
# NOTE: in this example only partitions 10, 20 and 30 actually exist in the database
#       partition group
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'A', '40,50,60', '10,20,30,40,50,60', '1,2,1,2,1,2')"

# ------------------------------------------------------------------------------------
# Analysing the effect of droping a partition without applying the changes
#
# - In the following case, the hypothesis is dropping partition 30 from the database
#   partition group, and redistributing the data in partitions 10 and 20 using a
#   respective target ratio of 1 : 1
#
# NOTE: In this example all partitions 10, 20 and 30 should exist in the database
#       partition group
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'D', '30', '10,20','1,1')"

# ------------------------------------------------------------------------------------
# Generate a data distribution file to be used by the redistribute process
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.generate_distfile('$tbSchema', '$tbName', '$distFile')"

# ------------------------------------------------------------------------------------
# Write a step wise redistribution plan into a registry
#
# Setting the 10th parameter to 1, may cause a currently running step wise redistribute
# stored procedure to complete the current step and stop, until this parameter is reset
# to 0, and the redistribute stored procedure is called again.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.set_swrd_settings('$dbpgName', 255, 0, ' ', '$distFile', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')"

# ------------------------------------------------------------------------------------
# Report the content of the step wise redistribution plan for the given database
# partition group.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.get_swrd_settings('$dbpgName', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

# ------------------------------------------------------------------------------------
# Redistribute the database partition group "dbpgName" according to the redistribution
# plan stored in the registry by set_swrd_settings. It starting with step 3 and
# redistributes the data until 2 steps in the redistribution plan are completed.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.stepwise_redistribute_dbpg('$dbpgName', 3, 2)"
```

関連概念:

- 333 ページの『データの再分散』

第 12 章 ベンチマーク・テスト

この章では、ベンチマークのプロセス、およびデータベース・ワークロードのベンチマーク・テストを実行するために *db2batch* ユーティリティーを使用する方法について説明します。

ベンチマーク・テスト

ベンチマーク・テストは、アプリケーション開発ライフ・サイクルの通常の部分のうちの 1 つです。これはアプリケーション開発者とデータベース管理者 (DBA) の両方が関係するチームの作業であり、現行のパフォーマンスを調べてそれを向上させるためにアプリケーションに対して行うものです。アプリケーション・コードが可能な限り効率的に作成されている場合、データベースおよびデータベース・マネージャーの構成パラメーターをチューニングすることにより、さらにパフォーマンスの改善を実現できます。また、アプリケーション・パラメーターもチューニングして、アプリケーションの要件をさらに満たすこともできます。

特定の種類の情報を得るには、さまざまなタイプのベンチマーク・テストを実行します。

- *1 秒当たりのトランザクション* ・ベンチマークは、特定の限定された実験条件の下でデータベース・マネージャーのスループット能力を調べるものです。
- *アプリケーション* ・ベンチマークは、同じスループット能力をより実動状態に近い条件下でテストするものです。

構成パラメーターをチューニングするベンチマークは、これらの「現実環境」条件に基づくものであり、アプリケーションが可能な限り効率的に実行されるまで、そのアプリケーションから取った SQL をパラメーター値を変えて繰り返し実行することが必要です。

ここで説明されるベンチマーク方式は構成パラメーターのチューニングを目的としています。しかし、同じ基本的な技法を、パフォーマンスに影響を与える次のような他の要因を調整するために使用することもできます。

- SQL ステートメント
- 索引
- 表スペース構成
- アプリケーション・コード
- ハードウェア構成

ベンチマークは、データベース・マネージャーが各種の条件下でどのように応答するかを調べるためのよい方法でもあります。デッドロック処理、ユーティリティー・パフォーマンス、データをデータベースに迅速にロードする異なる方式、ユーザーを追加する場合のトランザクション率、および新規リリースの製品を実動状態のときのアプリケーションへの影響をテストするよう、シナリオを作成することができます。

ベンチマーク・テスト方式

ベンチマーク・テストは、反復可能環境に基づいています。これにより、適切に比較できる結果を、同じ条件で実行した同じテストから得ることができます。

さらに、通常の環境でテスト・アプリケーションを実行することによってベンチマークを開始することもできます。パフォーマンス上の問題を追及していく際に、テストしている機能の有効範囲を制限する特殊化したテスト・ケースを作成することができます。つまり、特殊化テスト・ケースで、価値のある情報を得るためにアプリケーション全体をエミュレートする必要はありません。単純な測定から開始し、必要のある場合のみ複雑化させてください。

良いベンチマークまたは測定の特性には、次のものがあります。
- テストは繰り返しが可能です。
- テストの各繰り返しは、同じシステム状態で開始します。
- 他のアクティブがある程度そのシステムに含まれているシナリオの場合は除き、他にアクティブな関数やアプリケーションがシステムにありません。

  注: 開始されるアプリケーションは、最小化された状態またはアイドル状態であっても、メモリーを使用します。そのために、ページングがベンチマークの結果スキューしたり、反復可能度規則に違反したりする可能性が大きくなります。
- ベンチマークに使用されるハードウェアおよびソフトウェアは、インストール先の実稼働環境に適合しています。

ベンチマークには、シナリオを作成し、その後そのシナリオで数回アプリケーションを実行し、実行ごとにキー情報を把握してください。実行ごとにキー情報を把握することは、パフォーマンスとデータベースの両方のパフォーマンスを向上させる可能性のある変更を判別するために特に重要です。

関連概念:
- 348 ページの『ベンチマークの準備』
- 350 ページの『ベンチマーク・テストの作成』
- 356 ページの『ベンチマーク・テストの実行』
- 358 ページの『ベンチマーク・テストの分析例』

ベンチマークの準備

パフォーマンスのベンチマークを開始する前に、アプリケーションがそれに対して実行するデータベースの論理設計を完了しておいてください。表、ビュー、および索引をセットアップし、データを入れておきます。表を正規化し、アプリケーション・パッケージをバインドし、表に現実的なデータを入れておきます。

データベースの最終的な物理設計も決めておいてください。データベース・マネージャー・オブジェクトを最終ディスク・ロケーションに置き、作業ファイルとバックアップのロケーションを判別してログ・ファイルをサイズ変更し、バックアップ手順をテストします。さらに、パッケージをチェックして、可能な場合には行ブロッキングなどのパフォーマンス・オプションを使用可能にします。

アプリケーションのプログラミングと単体テストの中で、ベンチマーク・プログラムを作成できるようなフェーズに達していなければなりません。アプリケーション

の実際上の限界はベンチマーク・テスト時に分かりますが、ここで説明するベンチマークの目的はパフォーマンスを測定することであり、障害や異常終了を検出することではありません。

ベンチマーク・テスト・プログラムは、最終的な実稼働環境を可能な限り正確に表す状態で実行する必要があります。ベンチマークは、同じメモリー構成、同じディスク構成の同じモデル・サーバー上で実行するのが理想的です。最終的にアプリケーションに、たくさんのユーザーと大量のデータが関係してくる場合、このことは特に重要になります。ベンチマークで直接使用するオペレーティング・システム自体および通信機能またはファイル機能も、やはり調整済みでなければなりません。

ベンチマーク・テストは実動サイズ・データベースで実行するようにしてください。個々の SQL ステートメントが戻すデータは、実動での場合と同じ量で、同じ程度にソートされるものでなければなりません。この規則により、アプリケーションが典型的なメモリー所要量をテストすることになります。

ベンチマークの対象の SQL ステートメントは、*典型的* か*最悪のケース* のいずれかです。これらは以下で説明します。

典型的な SQL

典型的な SQL には、ベンチマークの対象のアプリケーションの一般的操作で実行されるステートメントが含まれます。選択するステートメントは、アプリケーションの性質によって異なります。たとえば、データ入力アプリケーションでは INSERT ステートメントをテストしますが、銀行業務トランザクションでは、FETCH、UPDATE、およびいくつかの INSERT をテストします。実行の頻度および選択されたステートメントによって処理されるデータの量は、平均と見なしてください。量が多過ぎる場合、典型的な SQL ステートメントであっても、*最悪のケース* のカテゴリーに入るものと見なします。

最悪のケースの SQL

このカテゴリーに該当するステートメントには、次のものがあります。

- 頻繁に実行されるステートメント。
- 大量のデータを処理するステートメント。
- 時間が重要なステートメント。

  たとえば、ある顧客から電話がかかったときに実行されるアプリケーションとそのステートメントとは、顧客を待たせている間に、顧客の情報の検索や更新を行わなければなりません。
- 結合される表の数が多いステートメント、またはアプリケーションの中でも最も複雑な SQL が含まれるステートメント。

  たとえば、口座の異なる種類すべての月ごとの活動を行うための組み合わされた顧客ステートメントを生成する銀行業務アプリケーション。共通表には顧客の住所と口座番号のリストを入れ、他のいくつかの表は結合して、すべての必要な口座トランザクション情報を処理および統合するようにしなければなりません。 1 つの口座に必要な作業量に、同じ期間内に処理しなければならない何千もの口座の数を乗算し、潜在的な時間の節約を考慮すると、パフォーマンス要件が導き出されます。

- アクセス・パスが不適切なステートメント。たとえば、あまり実行されないステートメントや、関与する表に関して作成された索引によってサポートされないステートメント。
- 経過時間の長いステートメント。
- アプリケーション初期設定時にのみ実行されるが、リソース要件が不均衡なステートメント。

  たとえば、その日のうちに処理されなければならない会計作業のリストを生成するアプリケーション。このアプリケーションが開始されると、最初の主要 SQL ステートメントにより 7 とおりの結合が生成され、それらによりこのアプリケーションのユーザーが担当する全会計の大きなリストが作成されます。このステートメントは 1 日のうち数回実行されるだけですが、正しく調整されていないと実行に数分間もかかってしまいます。

関連概念:

- 347 ページの『ベンチマーク・テスト』
- 350 ページの『ベンチマーク・テストの作成』

ベンチマーク・テストの作成

ベンチマーク・プログラムをインプリメントする場合は、さまざまな要因を考慮に入れてください。このプログラムの主な目的はユーザー・アプリケーションをシミュレートするということですから、プログラムの全体的な構造は異なります。アプリケーション全体をベンチマークとして使用し、単に複数の SQL ステートメントを分析するタイミングを合わせる手段として用いることもできます。大きかったり複雑であったりするアプリケーションの場合は、重要なステートメントを含むブロックのみを組み込んでおくほうがより実際的であることもあります。

特定の SQL ステートメントのパフォーマンスをテストする場合、ベンチマーク・プログラムにそのステートメントだけを組み込み、あとは
CONNECT、PREPARE、OPEN など他の必要なステートメントとタイミング機構を加えることもできます。

考慮すべき別の要因は、使用するベンチマークのタイプです。 1 組の SQL ステートメントを、一定の時間間隔をおいて繰り返し実行するという方法があります。実行するステートメントの数と時間間隔の比率によって、アプリケーションのスループットが決まります。あるいは、単に SQL ステートメントを個別に実行するのに必要とされる時間を判別するということもできます。

すべてのベンチマーク・テストにおいて、個別の SQL ステートメントまたはアプリケーション全体であれ、経過時間を算定するには効率的なタイミング・システムが必要です。個々の SQL ステートメントが別個に実行されるアプリケーションをシミュレートする場合、CONNECT、 PREPARE、および COMMIT ステートメントのための時間を追跡することが重要です。しかし、多くの異なるステートメントを処理するプログラムの場合、おそらく単一の CONNECT または COMMIT しか必要ではなく、個々のステートメントの実行時間に焦点を絞ることが優先です。

パフォーマンス分析においては各照会ごとの経過時間が重要な要因ですが、他の潜在的な障害が必ずしも明示されるわけではありません。たとえば、CPU 使用率、ロ

ッキング、およびバッファー・プール入出力に関する情報は、アプリケーションが CPU をフルに使用するのではなく入出力制約であることを示しているかもしれません。ベンチマーク・プログラムを行うことによって、必要に応じてより詳細な分析のためにこのようなデータを入手することができます。

必ずしもすべてのアプリケーションで、照会によって取り出した一連の行全体を特定の出力装置に送信しません。たとえば、応答セット全体を別のプログラムへの入力とし、最初のアプリケーションから 1 行も出力として送信されないようにすることもできます。画面出力のデータを形式制御すると通常は CPU の消費の多重度が高くなり、ユーザーの希望どおりには行かないこともあります。 正確にシミュレーションするには、ベンチマーク・プログラムが特定のアプリケーションの行処理を反映していなければなりません。行が出力装置に送信された場合に形式制御が不十分だと、 CPU の処理時間の大部分が消費され、 SQL ステートメント自体の実際のパフォーマンスが誤って表示されます。

db2batch ベンチマーク・ツール: ベンチマーク・ツール (db2batch) は、インスタンスの sqllib ディレクトリーの bin サブディレクトリー内に提供されています。このツールは、ベンチマーク・プログラムの作成に関する指針の多くを使用します。このツールは、フラット・ファイルまたは標準入力のいずれかから SQL ステートメントを読み取り、それらステートメントを動的に記述、作成し、応答セットを返します。また、このツールを使用すると、応答セットのサイズや、その応答セットから出力装置に送信される行数をコントロールすることもできます。

さらに、経過時間、CPU とバッファー・プールの使用率、およびデータベース・モニターから収集されるその他の統計を含め、提供されるパフォーマンス関連情報のレベルを指定することができます。一連の SQL ステートメントの時間を設定している場合は、 db2batch を指定すると、パフォーマンスの結果を要約し、算術方式と幾何学方式の両方を提供します。 構文およびオプションには、コマンド行で db2batch -h と入力します。

このベンチマーク・ツールにも CLI オプションがあります。このオプションで、キャッシュ・サイズを指定することができます。次の例では、キャッシュ・サイズが 30 ステートメントの CLI モードで db2batch が実行されます。

```
db2batch -d sample -f db2batch.sql -cli 30
```

リモートで db2batch を実行することもできます。

```
-f <filename>
```

または

```
-o <options>
```

というベンチマーク・ツールのコマンド・パラメーターのいずれかを使用する場合、次のようになります。

- 制御オプション

  ```
  perf_detail
  ```

  および

  ```
  -p <perf_detail>
  ```

(戻されるパフォーマンス情報のレベルを指定するために) 1 より大きい値が設定される場合、リモートでの実行中にはサポートされません。

これら 2 項目以外の場合、

```
perf_detail
```

および

```
-p <perf_detail>
```

制御オプション値はサポートされ、全 DB2® Universal Database プラットフォームで有効です。

db2batch テストの例

以下の例は、どのように db2batch を入力ファイル db2batch.sql と使用することができるかを示しています。

```
-- db2batch.sql
-- ------------
--#SET PERF_DETAIL 3 ROWS_OUT 5

-- This query lists employees, the name of their department
-- and the number of activities to which they are assigned for
-- employees who are assigned to more than one activity less than
-- full-time.
--#COMMENT Query 1
select lastname, firstnme,
       deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
      employee.empno = emp_act.empno and
      emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2;
--#SET PERF_DETAIL 1 ROWS_OUT 5
--#COMMENT Query 2
select lastname, firstnme,
       deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
      employee.empno = emp_act.empno and
      emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2;
```

図 26. ベンチマーク入力ファイルのサンプル: db2batch.sql

次のベンチマーク・ツールの呼び出しを使用すると、

```
db2batch -d sample -f db2batch.sql
```

次の出力が生成されます。

```
--#SET PERF_DETAIL 3 ROWS_OUT 5
Query 1

Statement number: 1

select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2
```

図 27. db2batch からの出力例 (第 1 部)

```
LASTNAME         FIRSTNME      DEPTNAME                       NUM_ACT
-------------------------------------------------------------------------
JEFFERSON        JAMES         ADMINISTRATION SYSTEMS                3
JOHNSON          SYBIL         ADMINISTRATION SYSTEMS                4
NICHOLLS         HEATHER       INFORMATION CENTER                    4
PEREZ            MARIA         ADMINISTRATION SYSTEMS                4
SMITH            DANIEL        ADMINISTRATION SYSTEMS                7
Number of rows retrieved is:        5
Number of rows sent to output is:   5
Elapsed Time is:           0.074      seconds
Locks held currently                        = 0
Lock escalations                            = 0
Total sorts                                 = 5
Total sort time (ms)                        = 0
Sort overflows                              = 0
Buffer pool data logical reads              = 13
Buffer pool data physical reads             = 5
Buffer pool data writes                     = 0
Buffer pool index logical reads             = 3
Buffer pool index physical reads            = 0
Buffer pool index writes                    = 0
Total buffer pool read time (ms)            = 23
Total buffer pool write time (ms)           = 0
Asynchronous pool data page reads           = 0
Asynchronous pool data page writes          = 0
Asynchronous pool index page reads          = 0
Asynchronous pool index page writes         = 0
Total elapsed asynchronous read time        = 0
Total elapsed asynchronous write time       = 0
Asynchronous read requests                  = 0
LSN Gap cleaner triggers                    = 0
Dirty page steal cleaner triggers           = 0
Dirty page threshold cleaner triggers       = 0
Direct reads                                = 8
Direct writes                               = 0
Direct read requests                        = 4
Direct write requests                       = 0
Direct read elapsed time (ms)               = 0
Direct write elapsed time (ms)              = 0
Rows selected                               = 5
Log pages read                              = 0
Log pages written                           = 0
Catalog cache lookups                       = 3
Catalog cache inserts                       = 3
Buffer pool data pages copied to ext storage    = 0
Buffer pool index pages copied to ext storage   = 0
Buffer pool data pages copied from ext storage  = 0
Buffer pool index pages copied from ext storage = 0
Total Agent CPU Time (seconds)              = 0.02
Post threshold sorts                        = 0
Piped sorts requested                       = 5
Piped sorts accepted                        = 5
```

図 28. db2batch からの出力例 (第 1 部)

```
--#SET PERF_DETAIL 1 ROWS_OUT 5
Query 2
Statement number: 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2
LASTNAME         FIRSTNME      DEPTNAME                        NUM_ACT
------------------------------------------------------------------------
GEYER            JOHN          SUPPORT SERVICES                      2
GOUNOT           JASON         SOFTWARE SUPPORT                      2
HAAS             CHRISTINE     SPIFFY COMPUTER SERVICE DIV.          2
JONES            WILLIAM       MANUFACTURING SYSTEMS                 2
KWAN             SALLY         INFORMATION CENTER                    2
Number of rows retrieved is:        8
Number of rows sent to output is:   5
Elapsed Time is:           0.037      seconds
Summary of Results
==================
                Elapsed             Agent CPU         Rows      Rows
Statement #     Time (s)            Time (s)          Fetched   Printed
1                    0.074               0.020          5         5
2                    0.037       Not Collected          8         5
Arith. mean     0.055
Geom.  mean     0.052
```

図 29. db2batch からの出力例 (第 2 部)

上記のサンプル出力には、データベース・システム・モニターによって返される特定のデータ・エレメントが含まれています。

次の例 (UNIX® の場合) では、-r オプションの使用方法について説明します。

```
db2batch -d sample -f db2batch.sql -r /dev/null,
```

-r オプションを使用すると、 outfile1 は /dev/null で置き換えられ、 outfile2 (結果表のみが含まれている) は NULL になります。そのため、db2batch は次のような出力を画面に送信します。

```
Summary of Results
==================
                Elapsed             Agent CPU         Rows      Rows
Statement #     Time (s)            Time (s)          Fetched   Printed
1                    0.074               0.020          5         5
2                    0.037       Not Collected          8         5
Arith. mean     0.055
Geom.  mean     0.052
```

図 30. db2batch からの出力例 -- 結果表のみ

関連概念:

- 350 ページの『ベンチマーク・テストの作成』
- 356 ページの『ベンチマーク・テストの実行』
- 358 ページの『ベンチマーク・テストの分析例』

ベンチマーク・テストの実行

あるタイプのデータベース・ベンチマークでは、構成パラメーターを選択し、そのパラメーターに異なるさまざまな値を使用して、最大の効果を得るまでテストを実行します。単一テストでは、同じパラメーター値を使用してアプリケーションを何度か反復して (たとえば、20 または 30 回) 実行するようにし、平均時間を調べます。これにより、パラメーター変更の効果がより明確になります。

ベンチマークを実行するときは、最初の反復 (ウォームアップ実行) を、後続の反復 (通常実行) とは別個のケースであると見なすようにします。ウォームアップ実行にはバッファー・プールの初期化などのいくつかの開始アクティビティーが含まれるため、通常実行よりいくらか時間が長くかかります。ウォームアップ実行から得られる情報は、現実的に有効である可能性はありますが、統計的には無効なものです。パラメーター値の特定の集合に関して平均時間や CPU を計算するときは、通常実行の結果のみを使用してください。

ベンチマークのウォームアップ実行を作成するために構成アドバイザーの使用を考慮することができます。構成アドバイザーが尋ねる質問により、ベンチマーク・アクティビティーにおいて通常実行の環境の構成を調整するときに考慮すべき内容について考えることができます。構成アドバイザーは、コントロール・センターからも開始できますし、適切なオプションを使用した db2 autoconfigure コマンドを実行しても開始できます。

ベンチマークが個々の照会を使用する場合、バッファー・プールをフラッシュして、直前の照会の潜在的な影響が最小限になるようにしてください。バッファー・プールをフラッシュするには、照会とは関係のない多数のページを読み取り、それでバッファー・プールを満たします。

パラメーター値の単一の集合での繰り返しを完了したら、単一のパラメーターを変更することができます。しかし、繰り返しから次の繰り返しまでの間に、以下に示すタスクを実行して、ベンチマーク環境がその元の状態にリストアされるようにしてください。

- . カタログ統計がテストにより更新された場合は、その統計については同じ値が繰り返し使用されるようにしてください。
- テストで使用されるデータがテストによって更新される場合、それらは一貫性のあるものにしなければなりません。そのためには、次のようにします。
  - RESTORE ユーティリティーを使用して、データベース全体をリストアします。データベースのバックアップ・コピーはその直前の状態を含み、次回のテストのために用意が整った状態になります。
  - IMPORT または LOAD ユーティリティーを使って、エクスポートされたデータ・コピーをリストアします。この方法では、影響を受けたデータだけをリストアできます。 REORG および RUNSTATS のユーティリティーは、このデータが入った表と索引に対して実行するようにしてください。
- アプリケーションを元の状態に戻すには、それをデータベースに再バインドします。

以下に、データベース・アプリケーションのベンチマークを実行するために行う、ステップまたは反復のサマリーを示します。

ステップ 1

データベースおよびデータベース・マネージャーの調整パラメーターは、それらの**デフォルト値**のままにしておきます。ただし、次のものは例外です。

- テストのワークロードと目標にとって重要なパラメーター。 (ベンチマーク・テストを実行しても、すべてのパラメーターを調整する時間はほとんどないので、一部のパラメーターには最適と推察される値を使用して開始し、そこから出発して調整を行うことができます。)
- アプリケーションの単体テストとシステム・テストのときに決められたログ・サイズ。
- アプリケーションを実行可能にするために変更しなければならないパラメーター (つまり、ステートメント・ヒープのメモリーを超えたなどのイベントから負の SQL 戻りコードが戻されることがないようにするために必要な変更)。

この初期ケースに関して一連の繰り返しを実行し、平均時間、または CPU を計算します。

ステップ 2

テストする調整パラメーターを 1 つだけ選択し、その値を変更します。

ステップ 3

別の一連の繰り返しを実行し、その平均時間、または CPU を計算します。

ステップ 4

ベンチマーク・テストの結果にしたがって、次のいずれかを実行します。

- パフォーマンスが向上した場合は、同じパラメーターの値を変更して、ステップ 3 に戻ります。最適値が示されるまで、このパラメーターを変更し続けます。
- パフォーマンスが低下したか、または変わらなかった場合は、パラメーターを前の値に戻してステップ 2 に戻り、新しいパラメーターを選択します。すべてのパラメーターをテストするまで、この手順を繰り返します。

  注: パフォーマンス結果をグラフ化する予定であれば、曲線が上がり始めるかまたは下がり始める点を探すようにしてください。

ベンチマーク・テストのためのドライバー・プログラムを作成することもできます。そのドライバー・プログラムは、REXX などの言語を使用して作成するか、または UNIX® ベースのプラットフォームの場合は、シェル・スクリプトを使用して作成できます。

このドライバー・プログラムはベンチマーク・プログラムを実行し、プログラムに適切なパラメーターを渡し、テストを複数回繰り返し、環境を一貫した状態にリストアし、新しいパラメーター値を使って次のテストを設定し、さらにテスト結果を収集/統合します。これらのドライバー・プログラムは非常に柔軟性に富んでおり、一連のベンチマーク・テスト全体を実行し、結果を分析してから、さらに所定のテストの最終パラメーター値および最適パラメーター値のレポートを作成する、といったことも可能です。

関連概念:

- 347 ページの『ベンチマーク・テスト』
- 348 ページの『ベンチマークの準備』

• 352 ページの『db2batch テストの例』

ベンチマーク・テストの分析例

ベンチマーク・プログラムからの出力には、各テストの ID、プログラム実行の繰り返し回数、ステートメント番号、および実行時間が含まれるようにします。

一連の測定の後ベンチマーク結果のサマリーは、次のように表示されます。

```
Test     Iter.   Stmt    Timing         SQL Statement
Numbr    Numbr   Numbr   (hh:mm:ss.ss)
 002      05      01     00:00:01.34    CONNECT TO SAMPLE
 002      05      10     00:02:08.15    OPEN cursor_01
 002      05      15     00:00:00.24    FETCH cursor_01
 002      05      15     00:00:00.23    FETCH cursor_01
 002      05      15     00:00:00.28    FETCH cursor_01
 002      05      15     00:00:00.21    FETCH cursor_01
 002      05      15     00:00:00.20    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      20     00:00:00.84    CLOSE cursor_01
 002      05      99     00:00:00.03    CONNECT RESET
```

図 31. ベンチマークの結果の例

注: 上記レポートのデータは、例示目的でのみ示したものです。測定された結果を表すものでは**ありません**。

これを分析すると、 CONNECT (ステートメント 01) に 1.34 秒、 OPEN CURSOR (ステートメント 10) に 2 分 8.15 秒かかり、 FETCHES (ステートメント 15) が 7 行を戻してその最大の遅延が .28 秒であり、 CLOSE CURSOR (ステートメント 20) に .84 秒、および CONNECT RESET (ステートメント 99) に .03 秒かかったことが分かります。

使用するプログラムが区切り付き ASCII フォーマットでデータを出力できる場合、後でそれをデータベース表やスプレッドシートにインポートし、さらに統計分析を行うことができます。

ベンチマーク・レポートのサンプル出力は、次のようになります。

```
PARAMETER        VALUES FOR EACH BENCHMARK TEST
TEST NUMBER      001     002     003    004     005
locklist         63      63      63      63      63
maxappls         8       8       8       8       8
applheapsz       48      48      48      48      48
dbheap           128     128     128     128     128
sortheap         256     256     256     256     256
maxlocks         22      22      22      22      22
stmtheap         1024    1024    1024    1024    1024
SQL STMT         AVERAGE TIMINGS (seconds)
  01             01.34   01.34   01.35   01.35   01.36
  10             02.15   02.00   01.55   01.24   01.00
  15             00.22   00.22   00.22   00.22   00.22
  20             00.84   00.84   00.84   00.84   00.84
  99             00.03   00.03   00.03   00.03   00.03
```

図 32. ベンチマークのタイミング・レポートの例

注: 上記レポートのデータは、例示目的でのみ示したものです。測定された結果を表すものでは**ありません**。

関連概念:

- 347 ページの『ベンチマーク・テスト』
- 350 ページの『ベンチマーク・テストの作成』
- 356 ページの『ベンチマーク・テストの実行』

第 13 章 DB2 の構成

構成パラメーター

DB2 Universal Database™ インスタンスまたはデータベースが作成されると、デフォルトのパラメーター値を持つ、対応する構成ファイルが作成されます。これらのパラメーター値を変更して、パフォーマンスを向上させ、インスタンスまたはデータベースの他の特性を改善することができます。

構成ファイル には、 DB2 UDB 製品および個々のデータベースに割り振られているリソースや、診断レベルなどの値を定義するパラメーターが含まれています。構成ファイルには次の 2 つのタイプがあります。

- DB2 UDB インスタンスごとに存在するデータベース・マネージャー構成ファイル。
- データベースごとに存在するデータベース構成ファイル。

データベース・マネージャー構成ファイル は、 DB2 UDB インスタンスの作成時に作成されます。これに含まれるパラメーターは、インスタンスの一部であるデータベースに関係なく、インスタンス・レベルでシステム・リソースに影響します。システムの構成によっては、これらのパラメーターの多くの値をシステム・デフォルト値から変更して、パフォーマンスを向上させたり容量を増やしたりすることができます。

それぞれのクライアントごとにも 1 つのデータベース・マネージャー構成ファイルがあります。このファイルには、特定のワークステーションのクライアント・イネーブラーに関する情報が入っています。サーバーに使用可能なパラメーターのサブセットは、クライアントにも適用できます。

データベース・マネージャーの構成パラメーターは、`db2systm` というファイルに保管されます。このファイルは、データベース・マネージャーのインスタンス作成時に作成されます。 UNIX ベースの環境では、このファイルはデータベース・マネージャーのインスタンス用サブディレクトリー `sqllib`にあります。 Windows では、このファイルは、デフォルト解釈として、 `sqllib` ディレクトリーのインスタンス用サブディレクトリーに入っています。 DB2INSTPROF 変数が設定されている場合は、このファイルは、 DB2INSTPROF 変数が指定するディレクトリーの `instance` サブディレクトリーに入っています。

パーティション・データベース環境では、このファイルは、どのデータベース・パーティション・サーバーも同じファイルにアクセスすることができるように、ファイル共用システムに常駐します。データベース・マネージャーの構成は、すべてのデータベース・パーティション・サーバーで同じです。

パラメーターの多くは、データベース・マネージャーの単一のインスタンスに割り振られるシステム・リソースの量を制御したり、あるいは、環境上の考慮事項に基づいて、データベース・マネージャーのセットアップや異なる通信サブシステムを構成します。その他に、情報提供だけを目的とした、変更不能のパラメーターがあ

ります。これらのすべてのパラメーターには、データベース・マネージャーのインスタンスに保管されているシングル・データベースとは関係なくグローバルに適用されるものです。

データベース構成ファイル は、データベースの作成時に作成され、データベースが常駐している場所に保管されます。データベースごとに 1 つの構成ファイルがあります。このデータベース構成ファイルで、データベースに割り当てるリソースの量を指定します。パラメーターの多くの値を変更して、パフォーマンスを向上させたり容量を増やしたりすることができます。特定のデータベースでの活動のタイプによっては、異なった変更が必要になることがあります。

個々のデータベース用のパラメーターは、SQLDBCON という構成ファイルに保管されます。このファイルは、データベースの他の制御ファイルとともに SQLnnnnn ディレクトリーに保管されています。 nnnnn はこのデータベースの作成時に割り当てられた番号です。各データベースにはそれぞれ構成ファイルがあり、パラメーターの多くはそのデータベースに割り振られるリソースの量を指定します。ファイルには、データベースの状況を示すフラグと共に、記述情報も入っています。

パーティション・データベース環境では、データベース・パーティションごとに別個の SQLDBCON ファイルが存在します。 SQLDBCON ファイルにある値は、データベース・パーティションごとに同じ値にも異なる値にもできますが、データベース構成パラメーター値はすべてのパーティションで同じ値にすることが推奨されています。

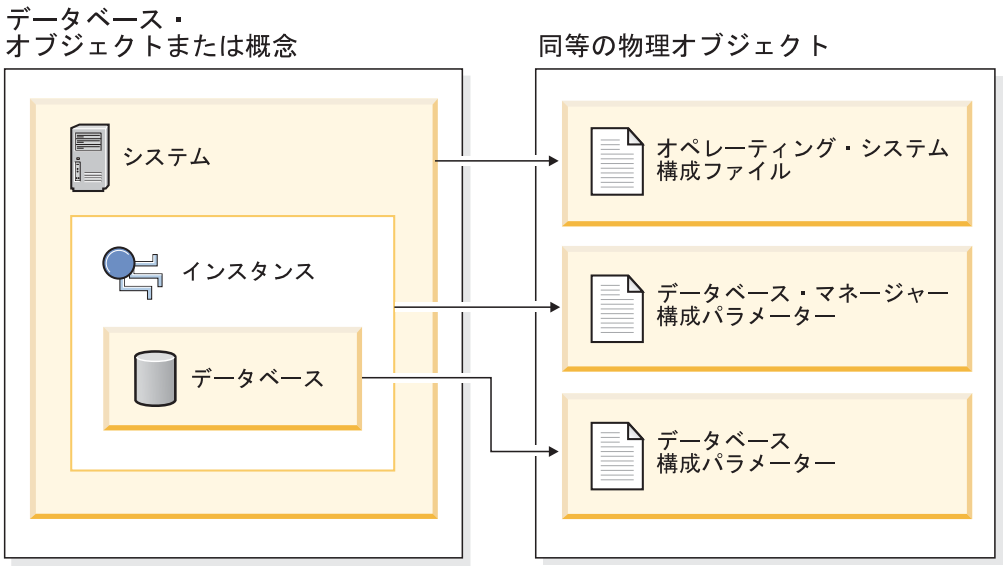


図 33. データベース・オブジェクトと構成ファイルの関係

関連概念:

- 363 ページの『構成パラメーターの調整』

関連タスク:

- 364 ページの『構成パラメーターによる DB2 の構成』

構成パラメーターの調整

データベース・マネージャーは、パラメーターのデフォルト値に基づいてディスク・スペースおよびメモリーを割り振ります。この割り振りでは、ある程度のニーズは満たすことができるかもしれませんが、デフォルト値を使用して最高のパフォーマンスを引き出すことはできない場合もあります。

デフォルト値は、比較的メモリーが小さいマシンに合わせて設定されており、データベース・サーバー専用に設定されています。したがって、次の環境では、デフォルト値を修正する必要があります。

- データベースが大きい
- 接続の数が多い
- 特定のアプリケーションでハイ・パフォーマンスが求められている
- ユニークな照会またはトランザクション・ロードまたはタイプを使用する
- さまざまなマシン構成または使用法を使う

それぞれのトランザクション処理環境は、ある 1 つの面または複数の面においてユニークです。デフォルト構成を使用した場合、このような相違点によりデータベース・マネージャーのパフォーマンスが大きな影響を受けることがあります。このため、ユーザーの環境に合わせて構成を調整するよう強くお勧めします。

ご使用の構成を調整するための最善の開始点は、「構成アドバイザー」または AUTOCONFIGURE コマンドです。

アプリケーションやユーザーのタイプが異なると、応答の所要時間や予想される結果も異なります。アプリケーションといっても、単純なデータ入力項目画面から、1 つの作業単位で複数の表にアクセスする複合 SQL ステートメントで成る戦略アプリケーションまで、幅広いタイプがあります。たとえば、電話による顧客サービス・アプリケーションと、バッチの報告書作成アプリケーションとでは応答の所要時間はかなり違ってきます。

一部の構成パラメーターを *automatic* に設定することができます。その後、DB2® は、現在のリソース要件を反映するようにこれらのパラメーターを自動的に調整します。

関連概念:

- 361 ページの『構成パラメーター』

関連タスク:

- 364 ページの『構成パラメーターによる DB2 の構成』

関連資料:

- 370 ページの『構成パラメーターのサマリー』

構成パラメーターによる DB2 の構成

データベース・マネージャーの構成パラメーターは、db2systm というファイルに保管されます。データベースの構成パラメーターは、SQLDBCON というファイルに保管されます。これらのファイルは直接編集できないので、提供されている API、または API を呼び出すツールを使って、変更や表示を行ってください。

警告: DB2 が提供している以外の方法を使用してdb2systm または SQLDBCON を編集すると、データベースが使用不能になることがあります。 DB2 で言及またはサポートされていない方法では、これらのファイルを変更しないよう**強くお勧めします。**

次の方法のいずれかを用いて、構成パラメーターのリセット、更新、または表示を行うことができます。

- コントロール・センターを使用します。インスタンスのパラメータの構成から「DBM 構成」ノートブックを使用すると、クライアントかサーバーのいずれかで、データベース・マネージャー構成パラメーターを設定することができます。データベースのパラメーター構成から「データベースの構成」ノートブックを使用すると、データベース構成パラメーターの値を変更することができます。また、DB2 コントロール・センターは、構成アドバイザーも提供しており、構成パラメーターの値をこれで変更することができます。このアドバイザーは、データベースに実行されるトランザクションのタイプやワークロードなどに関する一連の質問に対する応答に基づいて、パラメーターに値を生成します。

  パーティション・データベース環境では、 SQLDBCON ファイルは各データベース・パーティションごとに存在しています。「データベースの構成 (Configure Database)」ノートブックは、コントロール・センターのツリー・ビューにあるデータベース・オブジェクトから立ち上げられた場合に、全パーティション上の値を変更します。データベース・パーティション・オブジェクトから立ち上げられた場合は、そのパーティションの値だけを変更します。 (ただし、構成パラメーター値はすべてのパーティションで同じにするようお勧めします。)

  注: 構成アドバイザーは、パーティション・データベース環境では使用できません。
- コマンド行プロセッサーを使用します。設定を変更するためのコマンドを、す早くかつ容易に入力することができます。

  データベース・マネージャー構成パラメーターの場合:
  - GET DATABASE MANAGER CONFIGURATION (または GET DBM CFG)
  - UPDATE DATABASE MANAGER CONFIGURATION (または UPDATE DBM CFG)
  - RESET DATABASE MANAGER CONFIGURATION (または RESET DBM CFG)。すべてのデータベース・マネージャーのパラメーターをデフォルト値にリセットします。
  - AUTOCONFIGURE。

  データベース構成パラメーターの場合:
  - GET DATABASE CONFIGURATION (または GET DB CFG)

  - UPDATE DATABASE CONFIGURATION (または UPDATE DB CFG)
  - RESET DATABASE CONFIGURATION (または RESET DB CFG)。すべてのデータベースのパラメーターをデフォルト値にリセットします。
  - AUTOCONFIGURE。
- アプリケーション・プログラミング・インターフェース (API) を使用します。API は、アプリケーションまたはホスト言語プログラムから簡単に呼び出すことができます。
- 構成アシスタント (データベース・マネージャーの構成パラメーターの場合) を使用します。クライアント構成アシスタントは、構成パラメーターの設定にのみ使用できます。

一部の データベース・マネージャー 構成パラメーターの場合、新しいパラメーター値を反映させるために、データベース・マネージャーを一度停止 (db2stop) してから再始動 (db2start) する必要があります。

一部のデータベース・パラメーターの場合、データベースが再活動化されるときにのみ変更が有効になります。これらの場合、まずすべてのアプリケーションをデータベースから切断する必要があります。 (データベースが活動化している場合は、いったん非活動化し、それから活動化し直す必要があります。) その後は、データベースに最初に接続した時点で、変更が反映されます。

その他のパラメーターはオンラインで変更できます。これらのパラメーターは、*構成可能なオンライン構成パラメーター* と呼ばれます。

インスタンスにアタッチしている間に、構成可能なオンラインの データベース・マネージャー 構成パラメーターの設定を変更する場合、 UPDATE DBM CFG コマンドのデフォルトの動作は、即時に変更を適用することになっています。変更を即時に適用させたくない場合には、 UPDATE DBM CFG コマンドで DEFERRED オプションを使用します。

データベース・マネージャーの構成パラメーターをオンラインで変更するには、以下のようにします。

```
db2 attach to <instance-name>
db2 update dbm cfg using <parameter-name> <value>
db2 detach
```

クライアントの場合、データベース・マネージャー構成パラメーターの変更は、クライアントがサーバーに次回接続したときに反映されます。

接続中に、構成可能なオンラインのデータベース構成パラメーターを変更する場合には、デフォルトの動作は、可能であればどこであってもオンラインで変更を適用することになっています。パラメーターの場合、スペースの割り振りに関係したオーバーヘッドが生じるため、設定を変更しても、それが反映されるまでにかなりの時間がかかることがあります。 コマンド行プロセッサーからオンラインで構成パラメーターを変更するには、データベースへの接続が必要です。データベースの構成パラメーターをオンラインで変更するには、以下のようにします。

```
db2 connect to <dbname>
db2 update db cfg using <parameter-name> <parameter-value>
db2 connect reset
```

それぞれの構成可能なオンラインの構成パラメーターには、そのパラメーターと関連した*伝搬*クラスがあります。伝搬クラスは、構成パラメーターへの変更が有効であることを予期できるときを示します。以下の 3 つの伝搬クラスがあります。

- **immediate**: コマンドまたは API 呼び出しで即時に変更するパラメーター。たとえば、*diaglevel* には、immediate という伝搬クラスがあります。
- **Statement boundary**: ステートメントおよびステートメントなどの境界で変更するパラメーター。たとえば、*sortheap* という値を変更する場合、すべての新規 SQL 要求は、新規値を使用して開始します。
- **Transaction boundary**: トランザクション境界で変更するパラメーター。たとえば、*dl_expint* の新規値は、 COMMIT ステートメントの後に更新されます。

データベース構成パラメーターをいくつか変更しただけで、 SQL オプティマイザーが選択するアクセス・プランに影響が出ます。これらのパラメーターのいずれかを変更した場合には、 SQL ステートメントに最適なアクセス・プランが使用されるように、アプリケーションの再バインドを考慮してください。オンラインで変更 (たとえば、UPDATE DATABASE CONFIGURATION IMMEDIATE コマンドを使用して) されたパラメーターを使用すると、 SQL オプティマイザーは、新規 SQL ステートメントの新規アクセス・プランを選択することができます。ただし、一度実行された SQL ステートメントはキャッシュに保存され、パージされません。その SQL キャッシュの内容を消去するには、FLUSH PACKAGE CACHE ステートメントを使用します。

新しいパラメーター値は実際にはすぐに反映されませんが、 (GET DATABASE MANAGER CONFIGURATION または GET DATABASE CONFIGURATION コマンドを使用して) パラメーターの設定値を表示すると、必ず最新の更新内容が表示されます。これらのコマンドで SHOW DETAIL 文節を使用してパラメーターの設定値を表示すると、メモリー内の最新の更新内容と値の両方が表示されます。

注: ヘルプまたはその他の DB2 の資料では、構成パラメーター (たとえば、*userexit*) の大多数の許容値が「Yes」か「No」、あるいは「On」か「Off」であると記述されています。 混乱しないように、「Yes」は「On」と同じであり、「No」は「Off」と同じものと見なします。

関連概念:

- 361 ページの『構成パラメーター』
- 363 ページの『構成パラメーターの調整』

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 370 ページの『構成パラメーターのサマリー』
- 「*SQL リファレンス 第 2 巻*」の『FLUSH PACKAGE CACHE ステートメント』

パラメーターの動的構成

DB2 では、動的構成を活用できます。 DB2 データベースの実行中、接続を受け入れ中、またはトランザクションの処理中に、そのデータベースまたはインスタンス内の特定の構成パラメーターを変更できます。 DB2 の動的構成機能を活用する方法の例を、以下に示します。

このシナリオでは、データベース・サーバーは 4 GB のメモリーで構成され、そのうちの 3.5 GB はデータベース・マネージャーが使用可能であるものとします。搭載されているプロセッサーは 8 個です。データベース・サーバーは DB2DYN という単一のデータベース専用で、このデータベースはインスタンス DB2INST 上で実行します。データベース・ワークロードは、次のように 1 日の中また週ごとに変化します。

- 昼間のワークロード (05:00-20:00) には、多くの接続および並行トランザクションが含まれます。
- 一日の終わりのワークロード (20:00-24:00) には、接続とトランザクションの少ないサマリー報告書および意思決定支援照会が含まれます。
- 日次保守ワークロード (24:00-05:00) には、オンライン・ロード操作、増分バックアップ、索引作成などが含まれます。
- 週次保守ワークロードには、大規模な再編成操作、RUNSTATS 操作、および大規模な索引作成が含まれます。

上記のワークロード特性が指定されている場合には、次の表に示されているようにシステムを構成できます。

	昼間 (05:00 - 20:00)	**一日の終わり (20:00 - 24:00)**	**日次保守 (24:00 - 05:00)**	**週次保守 (毎日曜日)**
データベース活動	多量のトランザクション・ワークロード	意思決定支援照会	ロード、バックアップ、索引作成	REORG および RUNSTATS をバッファー・プール 1 で実行
バッファー・プール 1 (MB)	1000	500	500	2000
バッファー・プール 2 (MB)	1000	500	500	200
ソート・ヒープ (MB)	0.1	20	200	200
カタログ・キャッシュ (MB)	200	200	50	50
パッケージ・キャッシュ (MB)	800	200	200	200
ユーティリティー・ヒープ (MB)	0	0	1000	0
Diag レベル	1	3	4	4

始めにデータベース構成パラメーター *database_memory* を 3.5 GB に設定してから、ここで指定した量の使用可能なメモリーをデータベース用に予約することもできます。これは 1 回のみの操作であり、操作を完了すると、バッファー・プール作成または構成調整のために予約したメモリー 3.5 ギガバイト (または 4 キロバイト・ページ 917 504 ページ分) がデータベース用となります。

```
   db2start  db2 update db cfg for db2dyn using database_memory 917504
  db2stop
```

次いで、以下のスクリプトを使用して、データベースの構成を移行します。 (適切な時刻に実行されるよう、以下のスクリプトをスケジュールすることもできます。)

```
MorningConfiguration.sh

# This script is used to prepare
# the database server for the
# morning configuration,
# for a workload consisting of
# a large number of OLTP connections and
# concurent transactions.

db2 connect to db2dyn

db2 update db cfg using sortheap 25
db2 update db cfg using util_heap_sz 32
db2 alter bufferpool bufferpool1 size 262144
db2 commit
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 204800
db2 alter bufferpool bufferpool2 size 262144
db2 commit
db2 flush package cache dynamic
db2 get db cfg show detail
db2 connect reset

db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 1
db2 get dbm cfg show detail
db2 detach


EveningConfiguration.sh

# This script is used to prepare
# the database server for the
# evening configuration,
# for a workload consisting of
# decision-support queries.

db2 connect to db2dyn

db2 alter bufferpool bufferpool1 size 131072
db2 commit
db2 alter bufferpool bufferpool2 size 131072
db2 commit
db2 update db cfg using pkcachesz 51200
db2 update db cfg using catcachesz 51200
db2 update db cfg using util_heap_sz 32
db2 update db cfg using sortheap 5120
db2 flush package cache dynamic
db2 get db cfg show detail
db2 connect reset

db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 3
```

```
db2 get dbm cfg show detail
db2 detach
```

```
NightTimeConfiguration.sh

# This script is used to prepare
# the database server for the
# daily maintenance configuration,
# for a workload consisting of
# index creation and load and backup
# operations.

db2 connect to db2dyn

db2 alter bufferpool bufferpool1 size 131072
db2 commit
db2 alter bufferpool bufferpool2 size 131072
db2 commit
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 51200
db2 update db cfg using sortheap 51200
db2 update db cfg using util_heap_sz 262144
db2 flush package cache dynamic
db2 get db cfg show detail
db2 connect reset

db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 4
db2 get dbm cfg show detail
db2 detach
```

```
MaintenanceConfiguration.sh

# This script is used to prepare
# the database server for the
# weekly maintenance configuration,
# for a workload consisting of
# reorg and runstats operations executed
# in buffer pool 1.

db2 connect to db2dyn

db2 update db cfg using util_heap_sz 32
db2 alter bufferpool bufferpool2 size 51200
db2 commit
db2 update db cfg using sortheap 5120
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 51200
db2 alter bufferpool bufferpool1 size 524288
db2 commit
db2 flush package cache dynamic
db2 get db cfg show detail
db2 connect reset

db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 4
db2 get dbm cfg show detail
db2 detach
```

操作の順序がスクリプトごとに異なることに注意してください。メモリー要件を減らす操作を完了してから、メモリー要件を増やす操作が実行されるようにします。さもないと、メモリーの増加に失敗する可能性があります。

すべてのバッファー・プールのサイズ変更の後にはコミット操作が実行されていますが、これはバッファー・プールの変更が SQL 操作であり、トランザクションの一部であるためです。

パッケージ・キャッシュは再構成のたびにフラッシュされて、構成に重大な変更が加えられたこと、およびすべての既存の動的 SQL アクセス・プランが無効になっていることをオプティマイザーに通知します。

動的データベース・マネージャー再構成操作はアプリケーションがインスタンスに接続している間に実行され、動的データベース操作はアプリケーションがデータベースに接続している間に実行されます。

GET DATABASE MANAGER CONFIGURATION コマンド または GET DATABASE CONFIGURATION コマンドに SHOW DETAIL 文節を指定して実行し、動的再構成操作が有効になっているかどうかを検査できます。

このシナリオで動的に変更された可能性のある、その他のデータベース構成パラメーターは以下のとおりです。

- *locklist* パラメーター。データベースでロック・エスカレーションが頻繁に行われる場合に、このパラメーターが動的に増加している可能性があります。ただし、このパラメーターの値を動的に減らすことはできません。
- *dft_queryopt* パラメーター。新規接続のみに適用されるパラメーターで、意思決定支援ワークロードを開始する前に最適化レベルを増やすために使用されます。
- *dft_degree* パラメーター。これも新規接続のみに適用されるパラメーターで、意思決定支援照会に照会内並列処理を追加で提供するために使用されます。

構成パラメーターのサマリー

データベース・マネージャー構成パラメーター・サマリー

以下の表には、データベース・サーバー用のデータベース・マネージャー構成ファイルのパラメーターがリストされています。データベース・マネージャー構成パラメーターを変更する際は、各パラメーターの詳細情報も考慮に入れてください。デフォルトを含む個々のオペレーティング環境情報については、それぞれのパラメーターの説明に述べてあります。

一部のデータベース・マネージャー構成パラメーターの場合、新しいパラメーター値を反映させるために、データベース・マネージャーを一度停止 (`db2stop`) してから再始動 (`db2start`) する必要があります。その他のパラメーターはオンラインで変更できます。これらのパラメーターは、*構成可能なオンライン構成パラメーター* と呼ばれます。インスタンスにアタッチしている間に、構成可能なオンラインのデータベース・マネージャー構成パラメーターの設定を変更する場合、 UPDATE DBM CFG コマンドのデフォルトの動作は、即時に変更を適用します。変更を即時に適用させたくない場合には、 UPDATE DBM CFG コマンドで DEFERRED オプションを使用します。

以下の表の中の「自動」という列は、パラメーターが UPDATE DATABASE MANAGER CONFIGURATION コマンド上の AUTOMATIC キーワードをサポート

するかどうかを示しています。パラメーターを自動に設定する場合、 DB2 は、現在のリソース要件を反映するようにパラメーターを自動的に調整します。

「パフォーマンスへの影響」の欄は、各パラメーターのシステム性能に与える影響の重大度を示しています。この欄について完全に正確な情報を記述することはできないので、大まかな情報として扱ってください。

- **高** - パフォーマンスへの影響が大きいことを示します。これらのパラメーターの値を決定するときには注意が必要です。これは、場合によっては、提供されているデフォルト値を受け入れることを意味します。
- **中** - パラメーターがパフォーマンスに何らかの影響を与える可能性があることを示します。これらのパラメーターをどの程度調整する必要があるかは、ユーザーの環境および必要によって異なります。
- **低** - このパラメーターはパフォーマンスへの影響が低いことを示します。
- **なし** - このパラメーターはパフォーマンスへの直接の影響がないことを示します。これらのパラメーターをパフォーマンス向上のために調整する必要はありませんが、通信サポートなど、システム構成のその他の局面でこれらのパラメーターが非常に重要になる可能性があります。

「トークン」、「トークン値」、および「データ・タイプ」列は、 **db2CfgGet** または **db2CfgSet** API を呼び出す際に必要となる情報を提供します。この情報には、構成パラメーター ID、*db2CfgParam* データ構造の *token* エレメントの項目、およびその構造に渡される値のデータ・タイプが含まれます。

表 41. 構成可能データベース・マネージャー構成パラメーター

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
agent_stack_sz	いいえ	いいえ	低	SQLF_KTN_AGENT_STACK_SZ	61	Uint16	402 ページの『agent_stack_sz - エージェント・スタック・サイズ』
agentpri	いいえ	いいえ	高	SQLF_KTN_AGENTPRI	26	Sint16	436 ページの『agentpri - エージェントの優先順位』
aslheapsz	いいえ	いいえ	高	SQLF_KTN_ASLHEAPSZ	15	Uint32	413 ページの『aslheapsz - アプリケーション・サポート層ヒープ・サイズ』
audit_buf_sz	いいえ	いいえ	高	SQLF_KTN_AUDIT_BUF_SZ	312	Sint32	417 ページの『audit_buf_sz - 監査バッファー・サイズ』
authentication 注 1	いいえ	いいえ	低	SQLF_KTN_AUTHENTICATION	78	Uint16	542 ページの『authentication - 認証タイプ』
catalog_noauth	はい	いいえ	なし	SQLF_KTN_CATALOG_NOAUTH	314	Uint16	543 ページの『catalog_noauth - 権限なしで許可されるカタログ』
clnt_krb_plugin	いいえ	いいえ	なし	SQLF_KTN_CLNT_KRB_PLUGIN	812	char(33)	544 ページの『clnt_krb_plugin - クライアント Kerberos プラグイン』
clnt_pw_plugin	いいえ	いいえ	なし	SQLF_KTN_CLNT_PW_PLUGIN	811	char(33)	545 ページの『clnt_pw_plugin - クライアント・ユーザー ID パスワード・プラグイン』
comm_bandwidth	はい	いいえ	中	SQLF_KTN_COMM_BANDWIDTH	307	float	533 ページの『comm_bandwidth - 通信スピード』
conn_elapse	はい	いいえ	中	SQLF_KTN_CONN_ELAPSE	508	Uint16	517 ページの『conn_elapse - 接続経過時間』
cpuspeed	はい	いいえ	低 注 2	SQLF_KTN_CPUSPEED	42	float	534 ページの『cpuspeed - CPU 速度』
datalinks	いいえ	いいえ	低	SQLF_KTN_DATALINKS	603	Sint16	496 ページの『datalinks - データ・リンク・サポート使用可能』

表 41. 構成可能データベース・マネージャー構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
dft_account_str	はい	いいえ	なし	SQLF_KTN_DFT_ACCOUNT_STR	28	char(25)	535 ページの『dft_account_str - デフォルト・チャージバック・アカウント』
dft_monswitches • *dft_mon_bufpool* • *dft_mon_lock* • *dft_mon_sort* • *dft_mon_stmt* • *dft_mon_table* • *dft_mon_timestamp* • *dft_mon_uow*	はい	いいえ	中	SQLF_KTN_DFT_MONSWITCHES 注 3 • SQLF_KTN_DFT_MON_BUFPOOL • SQLF_KTN_DFT_MON_LOCK • SQLF_KTN_DFT_MON_SORT • SQLF_KTN_DFT_MON_STMT • SQLF_KTN_DFT_MON_TABLE • SQLF_KTN_DFT_MON_TIMESTAMP • SQLF_KTN_DFT_MON_UOW	29 • 33 • 34 • 35 • 31 • 32 • 36 • 30	Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16	531 ページの『dft_monswitches - デフォルトのデータベース・システム・モニター・スイッチ』
dftdbpath	はい	いいえ	なし	SQLF_KTN_DFTDBPATH	27	char(215)	545 ページの『dftdbpath - デフォルト・データベース・パス』
diaglevel	はい	いいえ	低	SQLF_KTN_DIAGLEVEL	64	Uint16	527 ページの『diaglevel - 診断エラー・キャプチャー・レベル』
diagpath	はい	いいえ	なし	SQLF_KTN_DIAGPATH	65	char(215)	528 ページの『diagpath - 診断データ・ディレクトリー・パス』
dir_cache	いいえ	いいえ	中	SQLF_KTN_DIR_CACHE	40	Uint16	418 ページの『dir_cache - ディレクトリー・キャッシュ・サポート』
discover 注 4	いいえ	いいえ	中	SQLF_KTN_DISCOVER	304	Uint16	515 ページの『discover - ディスカバリー・モード』
discover_inst	はい	いいえ	低	SQLF_KTN_DISCOVER_INST	308	Uint16	516 ページの『discover_inst - サーバー・インスタンスのディスカバリー』
fcm_num_anchors	はい	はい	中	SQLF_KTN_FCM_NUM_ANCHORS	506	Sint32	518 ページの『fcm_num_anchors - FCM メッセージ・アンカー数』
fcm_num_buffers	はい	はい	中	SQLF_KTN_FCM_NUM_BUFFERS	503	Uint32	519 ページの『fcm_num_buffers - FCM バッファー数』
fcm_num_connect	はい	はい	中	SQLF_KTN_FCM_NUM_CONNECT	505	Sint32	520 ページの『fcm_num_connect - FCM 接続項目数』
fcm_num_rqb	はい	はい	中	SQLF_KTN_FCM_NUM_RQB	504	Uint32	521 ページの『fcm_num_rqb - FCM 要求ブロック数』
fed_noauth	はい	いいえ	なし	SQLF_KTN_FED_NOAUTH	806	Uint16	547 ページの『fed_noauth - フェデレーテッド・データベース認証をバイパス』
federated	いいえ	いいえ	中	SQLF_KTN_FEDERATED	604	Sint16	536 ページの『federated - フェデレーテッド・データベース・システム・サポート』
fenced_pool	いいえ	いいえ	中	SQLF_KTN_FENCED_POOL	80	Sint32	448 ページの『fenced_pool - fenced プロセスの最大数』
group_plugin	いいえ	いいえ	なし	SQLF_KTN_GROUP_PLUGIN	810	char(33)	547 ページの『group_plugin - グループ・プラグイン』
health_mon	はい	いいえ	低	SQLF_KTN_HEALTH_MON	804	Uint16	529 ページの『health_mon - ヘルス・モニター』
indexrec 注 5	はい	いいえ	中	SQLF_KTN_INDEXREC	20	Uint16	481 ページの『indexrec - 索引再作成時点』
instance_memory	いいえ	はい	中	SQLF_KTN_INSTANCE_MEMORY	803	Uint64	420 ページの『instance_memory - インスタンス・メモリー』
intra_parallel	いいえ	いいえ	高	SQLF_KTN_INTRA_PARALLEL	306	Sint16	525 ページの『intra_parallel - パーティション内並列処理機能の使用可能化』

表 41. 構成可能データベース・マネージャー構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
java_heap_sz	いいえ	いいえ	高	SQLF_KTN_JAVA_HEAP_SZ	310	Sint32	421 ページの『java_heap_sz - Java インタープリター最大ヒープ・サイズ』
jdk_path	いいえ	いいえ	なし	SQLF_KTN_JDK_PATH	311	char(255)	536 ページの『jdk_path - Software Developer's Kit for Java インストール・パス』
keepfenced	いいえ	いいえ	中	SQLF_KTN_KEEPFENCED	81	Uint16	449 ページの『keepfenced - fenced プロセスの保持』
local_gssplugin	いいえ	いいえ	なし	SQLF_KTN_LOCAL_GSSPLUGIN	816	char(33)	548 ページの『local_gssplugin - ローカル・インスタンス・レベル許可に使用する GSS API プラグイン』
max_connections	いいえ	いいえ	中	SQLF_DBTN_MAX_CONNECTIONS	802	Sint32	438 ページの『max_connections - クライアント接続の最大数』
max_connretries	はい	いいえ	中	SQLF_KTN_MAX_CONNRETRIES	509	Uint16	522 ページの『max_connretries - ノード接続再試行回数』
max_coordagents	いいえ	いいえ	中	SQLF_KTN_MAX_COORDAGENTS	501	Sint32	439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
max_querydegree	はい	いいえ	高	SQLF_KTN_MAX_QUERYDEGREE	303	Sint32	525 ページの『max_querydegree - 照会の最大並列処理多重度』
max_time_diff	いいえ	いいえ	中	SQLF_KTN_MAX_TIME_DIFF	510	Uint16	523 ページの『max_time_diff - ノード間最大時差』
maxagents	いいえ	いいえ	中	SQLF_KTN_MAXAGENTS	12	Uint32	441 ページの『maxagents - エージェントの最大数』
maxcagents	いいえ	いいえ	中	SQLF_KTN_MAXCAGENTS	13	Sint32	443 ページの『maxcagents - 同時エージェントの最大数』
maxtotfilop	いいえ	いいえ	中	SQLF_KTN_MAXTOTFILOP	45	Uint16	445 ページの『maxtotfilop - 最大合計オープン・ファイル数』
min_priv_mem	いいえ	いいえ	中	SQLF_KTN_MIN_PRIV_MEM	43	Uint32	405 ページの『min_priv_mem - 最小コミット済み専用メモリー』
mon_heap_sz	いいえ	いいえ	低	SQLF_KTN_MON_HEAP_SZ	79	Uint16	422 ページの『mon_heap_sz - データベース・システム・モニター・ヒープ・サイズ』
nname	いいえ	いいえ	なし	SQLF_KTN_NNAME	7	char(8)	512 ページの『nname - NetBIOS ワークステーション名』
notifylevel	はい	いいえ	低	SQLF_KTN_NOTIFYLEVEL	605	Sint16	530 ページの『notifylevel - 通知レベル』
num_initagents	いいえ	いいえ	中	SQLF_KTN_NUM_INITAGENTS	500	Uint32	446 ページの『num_initagents - プール内エージェントの初期数』
num_initfenced	いいえ	いいえ	中	SQLF_KTN_NUM_INITFENCED	601	Sint32	450 ページの『num_initfenced - fenced プロセスの初期数』
num_poolagents	いいえ	いいえ	高	SQLF_KTN_NUM_POOLAGENTS	502	Sint32	447 ページの『num_poolagents - エージェント・プール・サイズ』
numdb	いいえ	いいえ	低	SQLF_KTN_NUMDB	6	Uint16	537 ページの『numdb - ホストおよび iSeries データベースを含めた並行アクティブ・データベースの最大数』
priv_mem_thresh	いいえ	いいえ	中	SQLF_KTN_PRIV_MEM_THRESH	44	Sint32	406 ページの『priv_mem_thresh - 専用メモリーしきい値』
query_heap_sz	いいえ	いいえ	中	SQLF_KTN_QUERY_HEAP_SZ	49	Sint32	407 ページの『query_heap_sz - 照会ヒープ・サイズ』

表 41. 構成可能データベース・マネージャー構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
resync_interval	いいえ	いいえ	なし	SQLF_KTN_RESYNC_INTERVAL	68	Uint16	488 ページの『resync_interval - トランザクション再同期インターバル』
rqrioblk	いいえ	いいえ	高	SQLF_KTN_RQRIOBLK	1	Uint16	416 ページの『rqrioblk - クライアント入出力ブロック・サイズ』
sheapthres	いいえ	いいえ	高	SQLF_KTN_SHEAPTHRES	21	Uint32	408 ページの『sheapthres - ソート・ヒープしきい値』
spm_log_file_sz	いいえ	いいえ	低	SQLF_KTN_SPM_LOG_FILE_SZ	90	Sint32	488 ページの『spm_log_file_sz - 同期点マネージャー・ログ・ファイル・サイズ』
spm_log_path	いいえ	いいえ	中	SQLF_KTN_SPM_LOG_PATH	313	char(226)	489 ページの『spm_log_path - 同期点マネージャー・ログ・ファイル・パス』
spm_max_resync	いいえ	いいえ	低	SQLF_KTN_SPM_MAX_RESYNC	91	Sint32	490 ページの『spm_max_resync - 同期点マネージャー再同期エージェント限界』
spm_name	いいえ	いいえ	なし	SQLF_KTN_SPM_NAME	92	char(8)	490 ページの『spm_name - 同期点マネージャー名』
srvcon_auth	いいえ	いいえ	なし	SQLF_KTN_SRVCON_AUTH	815	Uint16	548 ページの『srvcon_auth - サーバーでの着信接続の認証タイプ』
srvcon_gssplugin_list	いいえ	いいえ	なし	SQLF_KTN_SRVCON_GSSPLUGIN_LIST	814	char(256)	549 ページの『srvcon_gssplugin_list - サーバーでの着信接続用の GSS API プラグインのリスト』
srv_plugin_mode	いいえ	いいえ	なし	SQLF_KTN_SRV_PLUGIN_MODE	809	Uint16	551 ページの『srv_plugin_mode - サーバー・プラグイン・モード』
srvcon_pw_plugin	いいえ	いいえ	なし	SQLF_KTN_SRVCON_PW_PLUGIN	813	char(33)	550 ページの『srvcon_pw_plugin - サーバーでの着信接続用のユーザー ID-パスワード・プラグイン』
start_stop_time	はい	いいえ	低	SQLF_KTN_START_STOP_TIME	511	Uint16	524 ページの『start_stop_time - タイムアウトの開始および停止』
svcename	いいえ	いいえ	なし	SQLF_KTN_SVCENAME	24	char(14)	513 ページの『svcename - TCP/IP サービス名』
sysadm_group	いいえ	いいえ	なし	SQLF_KTN_SYSADM_GROUP	39	char(16)	551 ページの『sysadm_group - システム管理権限グループ名』
sysctrl_group	いいえ	いいえ	なし	SQLF_KTN_SYSCTRL_GROUP	63	char(16)	552 ページの『sysctrl_group - システム制御権限グループ名』
sysmaint_group	いいえ	いいえ	なし	SQLF_KTN_SYSMAINT_GROUP	62	char(16)	553 ページの『sysmaint_group - システム保守権限グループ名』
sysmon_group	いいえ	いいえ	なし	SQLF_KTN_SYSMON	808	char(9)	554 ページの『sysmon_group - システム・モニター権限グループ名』
tm_database	いいえ	いいえ	なし	SQLF_KTN_TM_DATABASE	67	char(8)	491 ページの『tm_database - トランザクション・マネージャー・データベース名』
tp_mon_name	いいえ	いいえ	なし	SQLF_KTN_TP_MON_NAME	66	char(19)	539 ページの『tp_mon_name - トランザクション・プロセッサー・モニター名』
tpname	いいえ	いいえ	なし	SQLF_KTN_TPNAME	25	char(64)	514 ページの『tpname - APPC トランザクション・プログラム名』
trust_allclnts 注 6	いいえ	いいえ	なし	SQLF_KTN_TRUST_ALLCLNTS	301	Uint16	555 ページの『trust_allclnts - 全クライアントのトラステッド化』
trust_clntauth	いいえ	いいえ	なし	SQLF_KTN_TRUST_CLNTAUTH	302	Uint16	556 ページの『trust_clntauth - トラステッド・クライアント認証』

表 41. 構成可能データベース・マネージャー構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
use_sna_auth	はい	いいえ	なし	SQLF_KTN_USE_SNA_AUTH	805	Uint16	557 ページの『use_sna_auth - SNA 認証の使用』
util_impact_lim	はい	いいえ	高	SQLF_KTN_UTIL_IMPACT_LIM	807	Uint32	541 ページの『util_impact_lim - インスタンス影響ポリシー』

注:

1. 有効な値 (sqlenv.h で定義):

```
SQL_AUTHENTICATION_SERVER (0)
SQL_AUTHENTICATION_CLIENT (1)
SQL_AUTHENTICATION_DCS (2)
SQL_AUTHENTICATION_DCE (3)
SQL_AUTHENTICATION_SVR_ENCRYPT (4)
SQL_AUTHENTICATION_DCS_ENCRYPT (5)
SQL_AUTHENTICATION_DCE_SVR_ENC (6)
SQL_AUTHENTICATION_KERBEROS (7)
SQL_AUTHENTICATION_KRB_SVR_ENC (8)
SQL_AUTHENTICATION_GSSPLUGIN (9)
SQL_AUTHENTICATION_GSS_SVR_ENC (10)
SQL_AUTHENTICATION_DATAENC (11)
SQL_AUTHENTICATION_DATAENC_CMP (12)
SQL_AUTHENTICATION_NOT_SPEC (255)
```

2. *cpuspeed* パラメーターはパフォーマンスに大きな影響を与える可能性がありますが、パラメーター記述に記述されているとおり、特定の場合以外はデフォルト値を使用してください。
3.
```
Bit 1 (xxxx xxx1): dft_mon_uow
Bit 2 (xxxx xx1x): dft_mon_stmt
Bit 3 (xxxx x1xx): dft_mon_table
Bit 4 (xxxx 1xxx): dft_mon_buffpool
Bit 5 (xxx1 xxxx): dft_mon_lock
Bit 6 (xx1x xxxx): dft_mon_sort
Bit 7 (x1xx xxxx): dft_mon_timestamp
```
4. 有効な値 (sqlutil.h で定義):

```
SQLF_DSCVR_KNOWN (1)
SQLF_DSCVR_SEARCH (2)
```

5. 有効な値 (sqlutil.h で定義):

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
```

6. 有効な値 (sqlutil.h で定義):

```
SQLF_TRUST_ALLCLNTS_NO (0)
SQLF_TRUST_ALLCLNTS_YES (1)
SQLF_TRUST_ALLCLNTS_DRDAONLY (2)
```

表 42. 情報提供用のデータベース・マネージャー構成パラメーター

パラメーター	トークン	トークン値	データ・タイプ	追加情報
nodetype 注 1	SQLF_KTN_NODETYPE	100	Uint16	537 ページの『nodetype - マシン・ノード・タイプ』
release	SQLF_KTN_RELEASE	101	Uint16	495 ページの『release - 構成ファイル・リリース・レベル』

表 42. 情報提供用のデータベース・マネージャー構成パラメーター (続き)

パラメーター	トークン	トークン値	データ・タイプ	追加情報
注: 1. 有効な値 (sqlutil.h で定義): SQLF_NT_STANDALONE (0) SQLF_NT_SERVER (1) SQLF_NT_REQUESTOR (2) SQLF_NT_STAND_REQ (3) SQLF_NT_MPP (4) SQLF_NT_SATELLITE (5)				

データベース構成パラメーターのサマリー

次の表は、データベース構成ファイルに入っているパラメーターをリストしています。データベース構成パラメーターを変更する際は、パラメーターの詳細情報も考慮に入れてください。

一部のデータベース構成パラメーターの場合、データベースが再活動化されるときにのみ変更が有効になります。これらの場合、まずすべてのアプリケーションをデータベースから切断する必要があります。 (データベースが活動化している場合は、いったん非活動化し、それから活動化し直す必要があります。) 次にデータベースに接続する際に、変更が有効になります。その他のパラメーターはオンラインで変更できます。これらのパラメーターは、*構成可能なオンライン構成パラメーター* と呼ばれます。

以下の表の中の「自動」という列は、パラメーターが UPDATE DATABASE MANAGER CONFIGURATION コマンド上の AUTOMATIC キーワードをサポートするかどうかを示しています。パラメーターを自動に設定する場合、 DB2 は、現在のリソース要件を反映するようにパラメーターを自動的に調整します。

「パフォーマンスへの影響」の欄は、各パラメーターのシステム性能に与える影響の重大度を示しています。この欄について完全に正確な情報を記述することはできないので、大まかな情報として扱ってください。

- **高** - パラメーターがパフォーマンスに大きな影響を与える可能性があることを示します。これらのパラメーターの値を決定するときには注意が必要です。これは、場合によっては、提供されているデフォルト値を受け入れることを意味します。
- **中** - このパラメーターがパフォーマンスへの影響が多少あることを示します。これらのパラメーターをどの程度調整する必要があるかは、ユーザーの環境および必要によって異なります。
- **低** - このパラメーターはパフォーマンスへの影響が低いことを示します。
- **なし** - このパラメーターはパフォーマンスへの直接の影響がないことを示します。これらのパラメーターをパフォーマンス向上のために調整する必要はありませんが、通信サポートなど、システム構成のその他の局面でこれらのパラメーターが非常に重要になる可能性があります。

「トークン」、「トークン値」、および「データ・タイプ」列は、 **db2CfgGet** または **db2CfgSet** API を呼び出す際に必要となる情報を提供します。この情報には、構成パラメーター ID、*db2CfgParam* データ構造の *token* エレメントの項目、およびその構造に渡される値のデータ・タイプが含まれます。

表 43. 構成可能なデータベース構成パラメーター

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
alt_collate	いいえ	いいえ	なし	SQLF_DBTN_ALT_COLLATE	809	Uint32	493 ページの『alt_collate - 代替照合シーケンス』
app_ctl_heap_sz	いいえ	いいえ	中	SQLF_DBTN_APP_CTL_HEAP_SZ	500	Uint16	398 ページの『app_ctl_heap_sz - アプリケーション・コントロール・ヒープ・サイズ』
appgroup_mem_sz	いいえ	いいえ	中	SQLF_DBTN_APPGROUP_MEM_SZ	800	Uint32	399 ページの『appgroup_mem_sz - アプリケーション・グループ・メモリー・セットの最大サイズ』
applheapsz	いいえ	いいえ	中	SQLF_DBTN_APPLHEAPSZ	51	Uint16	404 ページの『applheapsz - アプリケーション・ヒープ・サイズ』
archretrydelay	はい	いいえ	なし	SQLF_DBTN_ARCHRETRYDELAY	828	Uint16	463 ページの『archretrydelay - エラー時のアーカイブ再試行遅延』
autonomic_switches • *auto_maint* • *auto_db_backup* • *auto_tbl_maint* • *auto_runstats* • *auto_stats_prof* • *auto_prof_upd* • *auto_reorg*	はい	いいえ	中	SQLF_DBTN_AUTONOMIC_SWITCHES [注 1] • SQLF_ENABLE_AUTO_MAINT • SQLF_ENABLE_AUTO_DB_BACKUP • SQLF_ENABLE_AUTO_TBL_MAINT • SQLF_ENABLE_AUTO_RUNSTATS • SQLF_ENABLE_AUTO_STATS_PROF • SQLF_ENABLE_AUTO_PROF_UPD • SQLF_ENABLE_AUTO_REORG	830 • 831 • 833 • 835 • 837 • 839 • 844 • 841	Uint32	510 ページの『autonomic_switches - 自動保守スイッチ』
autorestart	はい	いいえ	低	SQLF_DBTN_AUTO_RESTART	25	Uint16	475 ページの『autorestart - 自動再始動使用可能』
avg_appls	はい	いいえ	高	SQLF_DBTN_AVG_APPLS	47	Uint16	438 ページの『avg_appls - アクティブ・アプリケーションの平均数』
blk_log_dsk_ful	はい	いいえ	なし	SQLF_DBTN_BLK_LOG_DSK_FUL	804	Uint16	464 ページの『blk_log_dsk_ful - ログ・ディスク・フル時のブロック』
catalogcache_sz	はい	いいえ	高	SQLF_DBTN_CATALOGCACHE_SZ	56	Sint32	386 ページの『catalogcache_sz - カタログ・キャッシュ・サイズ』
chngpgs_thresh	いいえ	いいえ	高	SQLF_DBTN_CHNGPGS_THRESH	38	Uint16	428 ページの『chngpgs_thresh - 変更済みページしきい値』
database_memory	いいえ	はい	中	SQLF_DBTN_DATABASE_MEMORY	803	Uint64	388 ページの『database_memory - データベース共用メモリー・サイズ』
dbheap	はい	いいえ	中	SQLF_DBTN_DB_HEAP	58	Uint64	389 ページの『dbheap - データベース・ヒープ』
dft_degree	はい	いいえ	高	SQLF_DBTN_DFT_DEGREE	301	Sint32	503 ページの『dft_degree - デフォルト多重度』

表 43. 構成可能なデータベース構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
dft_extent_sz	はい	いいえ	中	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32	429 ページの『dft_extent_sz - 表スペースのデフォルトのエクステント・サイズ』
dft_loadrec_ses	はい	いいえ	中	SQLF_DBTN_DFT_LOADREC_SES	42	Sint16	476 ページの『dft_loadrec_ses - ロード・リカバリー・セッションのデフォルト数』
dft_mttb_types	いいえ	いいえ	なし	SQLF_DBTN_DFT_MTTB_TYPES	843	Uint32	504 ページの『dft_mttb_types - 最適化用デフォルト保守表タイプ』
dft_prefetch_sz	はい	はい	中	SQLF_DBTN_DFT_PREFETCH_SZ	40	Sint16	430 ページの『dft_prefetch_sz - デフォルト・プリフェッチ・サイズ』
dft_queryopt	はい	いいえ	中	SQLF_DBTN_DFT_QUERYOPT	57	Sint32	504 ページの『dft_queryopt - デフォルト照会最適化クラス』
dft_refresh_age	いいえ	いいえ	中	SQLF_DBTN_DFT_REFRESH_AGE	702	char(22)	505 ページの『dft_refresh_age - デフォルト・リフレッシュ経過時間』
dft_sqlmathwarn	いいえ	いいえ	なし	SQLF_DBTN_DFT_SQLMATHWARN	309	Sint16	505 ページの『dft_sqlmathwarn - 演算例外時継続』
discover_db	はい	いいえ	中	SQLF_DBTN_DISCOVER	308	Uint16	516 ページの『discover_db - データベースのディスカバリー』
dl_expint	はい	いいえ	なし	SQLF_DBTN_DL_EXPINT	350	Sint32	497 ページの『dl_expint - データ・リンク・アクセス・トークン有効期限インターバル』
dl_num_copies	はい	いいえ	なし	SQLF_DBTN_DL_NUM_COPIES	351	Uint16	497 ページの『dl_num_copies - データ・リンクのコピー数』
dl_time_drop	はい	いいえ	なし	SQLF_DBTN_DL_TIME_DROP	353	Uint16	498 ページの『dl_time_drop - ドロップ後のデータ・リンク時間』
dl_token	はい	いいえ	低	SQLF_DBTN_DL_TOKEN	602	char(10)	498 ページの『dl_token - データ・リンク・トークン・アルゴリズム』
dl_upper	はい	いいえ	なし	SQLF_DBTN_DL_UPPER	603	Sint16	499 ページの『dl_upper - 大文字のデータ・リンク・トークン』
dl_wt_iexpint	はい	いいえ	なし	SQLF_DBTN_DL_WT_IEXPINT	354	Sint32	499 ページの『dl_wt_iexpint - データ・リンク書き込みトークン初期有効期限インターバル』
dlchktime	はい	いいえ	中	SQLF_DBTN_DLCHKTIME	9	Uint32	424 ページの『dlchktime - デッドロック・チェック・インターバル』
dyn_query_mgmt	いいえ	いいえ	低	SQLF_DBTN_DYN_QUERY_MGMT	604	Uint16	492 ページの『dyn_query_mgmt - 動的 SQL 照会管理』
estore_seg_sz	いいえ	いいえ	中	SQLF_DBTN_ESTORE_SEG_SZ	303	Sint32	431 ページの『estore_seg_sz - 拡張ストレージ・メモリー・セグメント・サイズ』

表 43. 構成可能なデータベース構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
failarchpath	はい	いいえ	なし	SQLF_DBTN_FAILARCHPATH	826	char(243)	464 ページの『failarchpath - フェイルオーバー・ログ・アーカイブ・パス』
groupheap_ratio	いいえ	いいえ	中	SQLF_DBTN_GROUPHEAP_RATIO	801	Uint16	401 ページの『groupheap_ratio - アプリケーション・グループ・ヒープ用メモリーのパーセント』
hadr_local_host	いいえ	いいえ	なし	SQLF_DBTN_HADR_LOCAL_HOST	811	char(256)	477 ページの『hadr_local_host - HADR ローカル・ホスト名』
hadr_local_svc	いいえ	いいえ	なし	SQLF_DBTN_HADR_LOCAL_SVC	812	char(41)	478 ページの『hadr_local_svc - HADR ローカル・サービス名』
hadr_remote_host	いいえ	いいえ	なし	SQLF_DBTN_HADR_REMOTE_HOST	813	char(256)	478 ページの『hadr_remote_host - HADR リモート・ホスト名』
hadr_remote_inst	いいえ	いいえ	なし	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)	479 ページの『hadr_remote_inst - リモート・サーバーの HADR インスタンス名』
hadr_remote_svc	いいえ	いいえ	なし	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)	479 ページの『hadr_remote_svc - HADR リモート・サービス名』
hadr_syncmode	いいえ	いいえ	なし	SQLF_DBTN_HADR_SYNCMODE	817	Uint32	480 ページの『hadr_syncmode - 対等状態にあるログ書き込みのための HADR 同期モード』
hadr_timeout	いいえ	いいえ	なし	SQLF_DBTN_HADR_TIMEOUT	816	Sint32	481 ページの『hadr_timeout - HADR タイムアウト値』
indexrec 注 2	はい	いいえ	中	SQLF_DBTN_INDEXREC	30	Uint16	481 ページの『indexrec - 索引再作成時点』
locklist	はい	いいえ	高 (エスカレーションに影響するとき)	SQLF_DBTN_LOCK_LIST	704	Uint64	390 ページの『locklist - ロック・リスト用最大ストレージ』
locktimeout	いいえ	いいえ	中	SQLF_DBTN_LOCKTIMEOUT	34	Sint16	425 ページの『locktimeout - ロック・タイムアウト』
logarchmeth1	はい	いいえ	なし	SQLF_DBTN_LOGARCHMETH1	822	Uint16	465 ページの『logarchmeth1 - 1 次ログ・アーカイブ方式』
logarchmeth2	はい	いいえ	なし	SQLF_DBTN_LOGARCHMETH2	823	Uint16	466 ページの『logarchmeth2 - 2 次ログ・アーカイブ方式』
logarchopt1	はい	いいえ	なし	SQLF_DBTN_LOGARCHOPT1	824	char(243)	466 ページの『logarchopt1 - 1 次ログ・アーカイブ・オプション』
logarchopt2	はい	いいえ	なし	SQLF_DBTN_LOGARCHOPT2	825	char(243)	467 ページの『logarchopt2 - 2 次ログ・アーカイブ・オプション』
logbufsz	いいえ	いいえ	高	SQLF_DBTN_LOGBUFSZ	33	Uint16	393 ページの『logbufsz - ログ・バッファー・サイズ』
logfilsiz	いいえ	いいえ	中	SQLF_DBTN_LOGFIL_SIZ	92	Uint32	452 ページの『logfilsiz - ログ・ファイルのサイズ』
logindexbuild	はい	はい	なし	SQLF_DBTN_LOGINDEXBUILD	818	Uint32	467 ページの『logindexbuild - 作成済み索引ページのログ』

表 43. 構成可能なデータベース構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
logprimary	いいえ	いいえ	中	SQLF_DBTN_LOGPRIMARY	16	Uint16	454 ページの『logprimary - 1 次ログ・ファイル数』
logretain 注 3	いいえ	いいえ	低	SQLF_DBTN_LOG_RETAIN	23	Uint16	468 ページの『logretain - ログ保存使用可能』
logsecond	はい	いいえ	中	SQLF_DBTN_LOGSECOND	17	Uint16	456 ページの『logsecond - 2 次ログ・ファイル数』
max_log	はい	はい		SQLF_DBTN_MAX_LOG	807	Uint16	457 ページの『max_log - トランザクション当たりの最大ログ』
maxappls	はい	はい	中	SQLF_DBTN_MAXAPPLS	6	Uint16	442 ページの『maxappls - アクティブ・アプリケーションの最大数』
maxfilop	はい	いいえ	中	SQLF_DBTN_MAXFILOP	3	Uint16	444 ページの『maxfilop - アプリケーション単位の最大データベース・ファイル・オープン数』
maxlocks	はい	いいえ	高 (エスカレーションに影響するとき)	SQLF_DBTN_MAXLOCKS	15	Uint16	426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
min_dec_div_3	いいえ	いいえ	高	SQLF_DBTN_MIN_DEC_DIV_3	605	Sint32	414 ページの『min_dec_div_3 - 10 進数除算の位取り 3』
mincommit	はい	いいえ	高	SQLF_DBTN_MINCOMMIT	32	Uint16	469 ページの『mincommit - グループ化するコミット数』
mirrorlogpath	いいえ	いいえ	低	SQLF_DBTN_MIRRORLOGPATH	806	char(242)	458 ページの『mirrorlogpath - ミラー・ログ・パス』
newlogpath	いいえ	いいえ	低	SQLF_DBTN_NEWLOGPATH	20	char(242)	459 ページの『newlogpath - データベース・ログ・パスの変更』
num_db_backups	はい	いいえ	なし	SQLF_DBTN_NUM_DB_BACKUPS	601	Uint16	483 ページの『num_db_backups - データベース・バックアップの数』
num_estore_segs	いいえ	いいえ	中	SQLF_DBTN_NUM_ESTORE_SEGS	304	Sint32	432 ページの『num_estore_segs - 拡張ストレージ・メモリー・セグメント数』
num_freqvalues	はい	いいえ	低	SQLF_DBTN_NUM_FREQVALUES	36	Uint16	507 ページの『num_freqvalues - 保存される頻度値の数』
num_iocleaners	いいえ	いいえ	高	SQLF_DBTN_NUM_IOCLEANERS	37	Uint16	432 ページの『num_iocleaners - 非同期ページ・クリーナーの数』
num_ioservers	いいえ	いいえ	高	SQLF_DBTN_NUM_IOSERVERS	39	Uint16	434 ページの『num_ioservers - I/O サーバーの数』
num_log_span	はい	はい		SQLF_DBTN_NUM_LOG_SPAN	808	Uint16	461 ページの『num_log_span - 番号ログ幅』
num_quantiles	はい	いいえ	低	SQLF_DBTN_NUM_QUANTILES	48	Uint16	508 ページの『num_quantiles - 列の変位値の数』
numarchretry	はい	いいえ	なし	SQLF_DBTN_NUMARCHRETRY	827	Uint16	470 ページの『numarchretry - エラー時の再試行数』
overflowlogpath	いいえ	いいえ	中	SQLF_DBTN_OVERFLOWLOGPATH	805	char(242)	461 ページの『overflowlogpath - オーバーフロー・ログ・パス』

表 43. 構成可能なデータベース構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報
pckcachesz	はい	いいえ	高	SQLF_DBTN_PCKCACHE_SZ	505	Uint32	394 ページの『pckcachesz - パッケージ・キャッシュ・サイズ』
rec_his_retentn	いいえ	いいえ	なし	SQLF_DBTN_REC_HIS_RETENTN	43	Sint16	484 ページの『rec_his_retentn - リカバリー履歴保存期間』
seqdetect	はい	いいえ	高	SQLF_DBTN_SEQDETECT	41	Uint16	435 ページの『seqdetect - 順次検出フラグ』
sheapthres_shr	いいえ	いいえ	高	SQLF_DBTN_SHEAPTHRES_SHR	802	Uint32	396 ページの『sheapthres_shr - 共用ソートのソート・ヒープのしきい値』
softmax	いいえ	いいえ	中	SQLF_DBTN_SOFTMAX	5	Uint16	471 ページの『softmax - リカバリー範囲およびソフト・チェックポイント・インターバル』
sortheap	はい	いいえ	高	SQLF_DBTN_SORT_HEAP	52	Uint32	410 ページの『sortheap - ソート・ヒープ・サイズ』
stat_heap_sz	いいえ	いいえ	低	SQLF_DBTN_STAT_HEAP_SZ	45	Uint32	411 ページの『stat_heap_sz - 統計ヒープ・サイズ』
stmtheap	はい	いいえ	中	SQLF_DBTN_STMT_HEAP	821	Uint32	412 ページの『stmtheap - ステートメント・ヒープ・サイズ』
trackmod	いいえ	いいえ	低	SQLF_DBTN_TRACKMOD	703	Uint16	485 ページの『trackmod - 変更されたページの追跡使用可能化』
tsm_mgmtclass	はい	いいえ	なし	SQLF_DBTN_TSM_MGMTCLASS	307	char(30)	485 ページの『tsm_mgmtclass - Tivoli Storage Manager 管理クラス』
tsm_nodename	はい	いいえ	なし	SQLF_DBTN_TSM_NODENAME	306	char(64)	486 ページの『tsm_nodename - Tivoli Storage Manager ノード名』
tsm_owner	はい	いいえ	なし	SQLF_DBTN_TSM_OWNER	305	char(64)	486 ページの『tsm_owner - Tivoli Storage Manager 所有者名』
tsm_password	はい	いいえ	なし	SQLF_DBTN_TSM_PASSWORD	501	char(64)	487 ページの『tsm_password - Tivoli Storage Manager パスワード』
userexit	いいえ	いいえ	低	SQLF_DBTN_USER_EXIT	24	Uint16	473 ページの『userexit - ユーザー出口使用可能』
util_heap_sz	はい	いいえ	低	SQLF_DBTN_UTIL_HEAP_SZ	55	Uint32	397 ページの『util_heap_sz - ユーティリティー・ヒープ・サイズ』
vendoropt	はい	いいえ	なし	SQLF_DBTN_VENDOROPT	829	char(242)	474 ページの『vendoropt - ベンダー・オプション』

表 43. 構成可能なデータベース構成パラメーター (続き)

パラメーター	オンラインで構成可能	自動	パフォーマンスへの影響	トークン	トークン値	データ・タイプ	追加情報

注:

1. Default => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint
   Bit 2 off (xxxx xxxx xxxx xx0x): auto_db_backup
   Bit 3 on (xxxx xxxx xxxx x0xx): auto_tbl_maint
   Bit 4 on (xxxx xxxx xxxx 1xxx): auto_runstats
   Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof
   Bit 6 off (xxxx xxxx xx0x xxxx): auto_prof_upd
   Bit 7 off (xxxx xxxx x0xx xxxx): auto_reorg
   0 0 1 9

   Maximum => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint
   Bit 2 off (xxxx xxxx xxxx xx1x): auto_db_backup
   Bit 3 on (xxxx xxxx xxxx x1xx): auto_tbl_maint
   Bit 4 on (xxxx xxxx xxxx 1xxx): auto_runstats
   Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof
   Bit 6 off (xxxx xxxx xx1x xxxx): auto_prof_upd
   Bit 7 off (xxxx xxxx x1xx xxxx): auto_reorg
   0 0 7 F
2. 有効な値 (sqlutil.h で定義):

   SQLF_INX_REC_SYSTEM (0)
   SQLF_INX_REC_REFERENCE (1)
   SQLF_INX_REC_RESTART (2)
3. 有効な値 (sqlutil.h で定義):

   SQLF_LOGRETAIN_NO (0)
   SQLF_LOGRETAIN_RECOVERY (1)
   SQLF_LOGRETAIN_CAPTURE (2)

表 44. 情報提供用のデータベース構成パラメーター

パラメーター	トークン	トークン値	データ・タイプ	追加情報
backup_pending	SQLF_DBTN_BACKUP_PENDING	112	Uint16	500 ページの『backup_pending - バックアップ・ペンディング標識』
codepage	SQLF_DBTN_CODEPAGE	101	Uint16	494 ページの『codepage - データベースのコード・ページ』
codeset	SQLF_DBTN_CODESET	120	char(9) 注 1	494 ページの『codeset - データベース用コード・セット』
collate_info	SQLF_DBTN_COLLATE_INFO	44	char(260)	494 ページの『collate_info - 照合情報』
country	SQLF_DBTN_COUNTRY	100	Uint16	495 ページの『country - データベース・テリトリー・コード』
database_consistent	SQLF_DBTN_CONSISTENT	111	Uint16	501 ページの『database_consistent - データベースの整合性』

表 44. 情報提供用のデータベース構成パラメーター (続き)

パラメーター	トークン	トークン値	データ・タイプ	追加情報
database_level	SQLF_DBTN_DATABASE_LEVEL	124	Uint16	495 ページの『database_level - データベース・リリース・レベル』
hadr_db_role	SQLF_DBTN_HADR_DB_ROLE	810	Uint32	476 ページの『hadr_db_role - HADR データベース役割』
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	Uint16	501 ページの『log_retain_status - ログ保存状況標識』
loghead	SQLF_DBTN_LOGHEAD	105	char(12)	453 ページの『loghead - 最初のアクティブ・ログ・ファイル』
logpath	SQLF_DBTN_LOGPATH	103	char(242)	453 ページの『logpath - ログ・ファイルのロケーション』
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	Uint16	501 ページの『multipage_alloc - マルチページ・ファイル割り振り使用可能』
numsegs	SQLF_DBTN_NUMSEGS	122	Uint16	435 ページの『numsegs - SMS コンテナーのデフォルト数』
release	SQLF_DBTN_RELEASE	102	Uint16	495 ページの『release - 構成ファイル・リリース・レベル』
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	Uint16	502 ページの『restore_pending - リストア・ペンディング』
rollfwd_pending	SQLF_DBTN_ROLLFWD_PENDING	113	Uint16	502 ページの『rollfwd_pending - ロールフォワード・ペンディング標識』
territory	SQLF_DBTN_TERRITORY	121	char(5) 注 2	496 ページの『territory - データベース・テリトリー』
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	Uint16	502 ページの『user_exit_status - ユーザー出口状況標識』

注:

1. HP-UX および Solaris オペレーティング環境 では char(17)。
2. HP-UX および Solaris オペレーティング環境 では char(33)。

DB2 Administration Server (DAS) 構成パラメーターのサマリー

表 45. DAS 構成パラメーター

パラメーター	パラメーター・タイプ	追加情報
authentication	構成可能	558 ページの『authentication - 認証タイプ DAS』
contact_host	オンラインで構成可能	559 ページの『contact_host - 連絡先リストのロケーション』
das_codepage	オンラインで構成可能	560 ページの『das_codepage - DAS コード・ページ』
das_territory	オンラインで構成可能	560 ページの『das_territory - DAS テリトリー』
dasadm_group	構成可能	561 ページの『dasadm_group - DAS 管理者権限グループ名』
db2system	オンラインで構成可能	561 ページの『db2system - DB2 サーバー・システムの名前』
discover	オンラインで構成可能	562 ページの『discover - DAS ディスカバリー・モード』
exec_exp_task	構成可能	563 ページの『exec_exp_task - 有効期限切れタスクの実行』
jdk_64_path	オンラインで構成可能	563 ページの『jdk_64_path - 64 ビット Software Developer's Kit for Java インストール・パス DAS』
jdk_path	オンラインで構成可能	564 ページの『jdk_path - Software Developer's Kit for Java インストール・パス DAS』
sched_enable	構成可能	565 ページの『sched_enable - スケジューラー・モード』
sched_userid	通知	565 ページの『sched_userid - スケジューラー・ユーザー ID』
smtp_server	オンラインで構成可能	566 ページの『smtp_server - SMTP サーバー』
toolscat_db	構成可能	567 ページの『toolscat_db - ツール・カタログ・データベース』
toolscat_inst	構成可能	567 ページの『toolscat_inst - ツール・カタログ・データベース・インスタンス』
toolscat_schema	構成可能	568 ページの『toolscat_schema - ツール・カタログ・データベース・スキーマ』

機能別のパラメーター詳細

次のセクションでは、さまざまな構成パラメーターについて理解し、チューニングするための、追加情報を詳細に記述します。次のように、個々のパラメーターについて機能または目的別に説明します。

- 385 ページの『キャパシティー管理』
- 451 ページの『ロギングおよびリカバリー』
- 492 ページの『データベース管理』
- 512 ページの『通信』
- 517 ページの『パーティション・データベース環境』
- 527 ページの『インスタンス管理』
- 558 ページの『DB2 Administration Server』

パラメーターごとに、次の情報について説明します。

構成タイプ
パラメーターの設定値がどの構成ファイルに入っているか。

- データベース・マネージャー は、データベース・マネージャーのインスタンス、およびそのイ

ンスタンス中で定義されているすべてのデータベースに影響を及ぼします。

- データベース は、特定のデータベースに影響を与えます。

パラメーター・タイプ

パラメーター値が変更可能かどうか、また、変更がオンラインで有効になるかどうか。

- *構成可能*

  パラメーターは、ある範囲の値をとることができ、データベース管理者がアプリケーションについてもっている知識およびベンチマークの経験のいずれかに基づいて、パラメーターを調整することができます。

- *オンラインで構成可能*

  パラメーターは、ある範囲の値をとることができ、データベース管理者がアプリケーションについてもっている知識およびベンチマークの経験のいずれか、またはその両方に基づいて、パラメーターを調整することができます。データベースがオンラインである間にも変更を適用することができます。この場合データベース・マネージャーをいったん停止してから再始動する、またはデータベースを再活動化する必要はありません。

- *通知*

  これらのパラメーターを変更できるのはデータベース・マネージャーだけであり、データベースが作成された際の DB2 のリリース、または必要なバックアップがペンディングになっている指示などの情報を含むことができます。

キャパシティー管理

データベースとデータベース・マネージャーの両方のレベルで、システムのスループットを制御する構成パラメーターがいくつかあります。この種のパラメーターは、次のカテゴリーに分けることができます。

- 386 ページの『データベース共用メモリー』
- 398 ページの『アプリケーション共用メモリー』
- 401 ページの『エージェント専用メモリー』
- 412 ページの『エージェント/アプリケーション通信メモリー』
- 417 ページの『データベース・マネージャー・インスタンス・メモリー』
- 424 ページの『ロック』
- 428 ページの『入出力およびストレージ』
- 436 ページの『エージェント』
- 448 ページの『ストアード・プロシージャーおよびユーザー定義関数』

データベース共用メモリー

次のパラメーターは、システムで割り振られるデータベース・グローバル・メモリーを制御します。

- 『catalogcache_sz - カタログ・キャッシュ・サイズ』
- 388 ページの『database_memory - データベース共用メモリー・サイズ』
- 389 ページの『dbheap - データベース・ヒープ』
- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 393 ページの『logbufsz - ログ・バッファー・サイズ』
- 394 ページの『pckcachesz - パッケージ・キャッシュ・サイズ』
- 396 ページの『sheapthres_shr - 共用ソートのソート・ヒープのしきい値』
- 397 ページの『util_heap_sz - ユーティリティー・ヒープ・サイズ』

catalogcache_sz - カタログ・キャッシュ・サイズ

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
伝搬クラス	即時
デフォルト[範囲]	-1 [8 - 524 288]
単位	ページ (4 KB)
割り振られるタイミング	データベースの初期化時
解放されるタイミング	データベースが停止されるとき

このパラメーターは、データベース共用メモリーから割り振られ、システム・カタログ情報をキャッシュに入れる場合に使用されます。パーティション・データベース・システムでは、それぞれのデータベース・パーティションごとにカタログ・キャッシュが 1 つずつあります。

個々のパーティションでカタログ情報をキャッシュに入れると、データベース・マネージャーは、以前検索された情報を入手するためにシステム・カタログ (またはパーティション・データベース環境におけるカタログ・ノード、あるいはその両方) にアクセスする必要がなくなるので、その内部オーバーヘッドを低減できます。カタログ・キャッシュは、次の情報を保管するのに使用されます。

- SYSTABLES 情報 (パック記述子を含む)
- SYSDBAUTH 情報およびルーチンの実行特権を含む、許可情報
- SYSROUTINES 情報

カタログ・キャッシュを使用することにより、次の操作の総合的なパフォーマンスを向上することができます。

- パッケージのバインドおよび SQL ステートメントのコンパイル
- データベース・レベル特権のチェックを伴う操作
- ルーチンの実行特権のチェックを伴う操作
- パーティション・データベース環境で非カタログ・ノードに接続されるアプリケーション

サーバーまたはパーティション・データベース環境でデフォルト (-1) を取ることによって、ページ割り振りの計算に使用される値は、*maxappls* 構成パラメーターに指定されている値の 4 倍になります。これに対する例外が生じるのは、*maxappls* の 4 倍が 8 より小さい場合です。この状態では、デフォルト値 -1 で、*catalogcache_sz* は 8 に設定されます。

推奨: デフォルト値で開始し、データベース・システム・モニターを使用して調整してください。このパラメーターを調整するときは、カタログ・キャッシュ用として予約されている余分のメモリーについて、たとえば、バッファー・プールやパッケージ・キャッシュなどといった別の目的に割り振った方が、その有効性が増すかどうか考慮する必要があります。

短時間に SQL コンパイルが集中し、その後はほとんど発生しないようなケースでは、このパラメーターの調整が特に重要です。キャッシュが大き過ぎる場合は、使用されなくなった情報のコピーの保留にメモリーが浪費される可能性があります。

パーティション・データベース環境では、非カタログ・ノードで必要とされるカタログ情報は、必ず最初にカタログ・ノードでキャッシュに入れられるので、カタログ・ノードの *catalogcache_sz* は、設定値を大きくする必要があるかどうか考慮してください。

cat_cache_lookups (カタログ・キャッシュ参照数)、*cat_cache_inserts* (カタログ・キャッシュ挿入)、*cat_cache_overflows* (カタログ・キャッシュ・オーバーフロー)、および *cat_cache_size_top* (カタログ・キャッシュ最高水準点) モニター・エレメントは、この構成パラメーターを調整する必要があるかどうか判別する場合に役立ちます。

注: カタログ・キャッシュは、パーティション・データベース環境のすべてのノードに存在します。それぞれのノードごとにローカル・データベース構成ファイルがあるので、それぞれのノードの *catalogcache_sz* 値によって、ローカル・カタログ・キャッシュのサイズが定義されます。キャッシングが効率的に行われ、オーバーフローが発生しないようにするために、それぞれのノードで *catalogcache_sz* 値を明示的に設定し、非カタログ・ノードの *catalogcache_sz* をカタログ・ノードの値よりも小さい値に設定できる可能性を考慮する必要があります。非カタログ・ノードでキャッシュに入れる必要のある情報は、カタログ・ノードのキャッシュから検索されるということを念頭に置いておいてください。したがって、非カタログ・ノードのカタログ・キャッシュは、カタログ・ノードのカタログ・キャッシュにある情報のサブセットのようなものです。

一般的に、キャッシュ・スペースが多く必要になるのは、作業単位に幾つもの動的 SQL ステートメントが含まれる場合、または多数の静的 SQL ステートメントが含まれるパッケージをバインドする場合です。

関連資料:

- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 「システム・モニター *ガイドおよびリファレンス*」の『cat_cache_lookups カタログ・キャッシュ参照数 : モニター・エレメント』

- 「システム・モニター *ガイドおよびリファレンス*」の『cat_cache_inserts カタログ・キャッシュ挿入数 : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『cat_cache_overflows カタログ・キャッシュ・オーバーフロー数 : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「システム・モニター *ガイドおよびリファレンス*」の『cat_cache_size_top カタログ・キャッシュ最高水準点 : モニター・エレメント』

database_memory - データベース共用メモリー・サイズ

構成タイプ データベース

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] AUTOMATIC [0 — 4 294 967 295]

単位 ページ (4 KB)

割り振られるタイミング データベースが活動化された時点

解放されるタイミング データベースが非活動化された時点

このパラメーターは、メモリー領域を共有するデータベース用として予約されている共用メモリーの量を指定します。この量が個々のパラメーター (たとえば、locklist、ユーティリティー・ヒープ、バッファー・プールなど) から計算した量よりも少ない場合には、その大きい方の量が使用されます。

このパラメーターの管理を単純化するために、AUTOMATIC 設定によって、必要なメモリーの量の計算、およびデータベース活動化時点におけるその割り振りを、DB2 に指示します。 DB2 はオーバーフロー・バッファー用にいくらか追加メモリーを割り当てることもします。ヒープが構成サイズを超える場合は常に、データベース共用メモリー領域内のヒープのピーク・メモリー要件を満たすためにオーバーフロー・バッファーが使用されます。動的構成更新などその他の操作も、このオーバーフロー・バッファーにアクセスできます。 **db2pd** コマンドに -memsets オプションを指定して使用し、オーバーフロー・バッファーに残されている未使用のメモリーの量をモニターできます。 64 ビット DB2 (AIX 版) では、データベースの実共用メモリーの使用量は動的に 64GB まで増加し、データベースの必要に対応します。ですから、DATABASE_MEMORY パラメーターを明示的に制御する必要はありません。ただし、DB2_PINNED_BP または DB2_LGPAGE_BP レジストリー変

数を設定すると、データベース共用メモリーを増やす機能が制限されます。『パフォーマンス変数』で、上記のレジストリー変数の説明を参照してください。

推奨: この値は、通常、AUTOMATIC のままになっています。ただし、将来の拡張に備えて追加のメモリーを予約するのに使用できます。たとえば、追加のメモリーは、新規バッファー・プールを作成する場合や、既存のバッファー・プールのサイズを大きくする場合に使用できます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 591 ページの『パフォーマンス変数』
- 「コマンド・リファレンス」の『db2pd - Monitor and Troubleshoot DB2 コマンド』

dbheap - データベース・ヒープ

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

伝搬クラス 即時

デフォルト[範囲]

UNIX 1200 [32 - 524 288]

ローカル・クライアントおよびリモート・クライアントを持つ Windows データベース・サーバー
600 [32 - 524 288]

ローカル・クライアントを持つ Windows 64 ビット・データベース・サーバー
600 [32 - 524 288]

ローカル・クライアントを持つ Windows 32 ビット・データベース・サーバー
300 [32 - 524 288]

単位 ページ (4 KB)

割り振られるタイミング データベースが活動化された時点

解放されるタイミング データベースが非活動化された時点

データベースごとに 1 つのデータベース・ヒープが存在し、データベース・マネージャーは、データベースに接続されているすべてのアプリケーションの代わりに、データベース・ヒープを使用します。これには表、索引、表スペース、およびバッファー・プールのコントロール・ブロック情報が含まれます。また、ログ・バッファー (*logbufsz*)、およびユーティリティーによって使用される一時メモリー用のスペースも含まれます。したがって、ヒープのサイズは多数の変数によって決まること

になります。コントロール・ブロック情報は、すべてのアプリケーションがデータベースから切断されるまで、ヒープ内に保持されます。

データベース・マネージャーが始動のために取得する必要がある最少量は、最初の接続時に割り振られます。このデータ域は、最終的にデータベース共用メモリーの全オーバーフロー・メモリー領域が使用されるに至るまで、必要に応じて拡張されます。

データベース・システム・モニターを使用すると、db_heap_top (割り振られる最大データベース・ヒープ) エレメントを使用して、データベース・ヒープ用として使用されたメモリーの最大量を見ることができます。

関連資料:

- 393 ページの『logbufsz - ログ・バッファー・サイズ』
- 「システム・モニター *ガイドおよびリファレンス*」の『db_heap_top 割り振られた最大データベース・ヒープ : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

locklist - ロック・リスト用最大ストレージ

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

伝搬クラス 即時

デフォルト[範囲]

UNIX 100 [4 - 524 288]

ローカル・クライアントおよびリモート・クライアントを持つ Windows データベース・サーバー
50 [4 - 524 288]

ローカル・クライアントを持つ Windows 64 ビット・データベース・サーバー
50 [4 - 60 000]

ローカル・クライアントを持つ Windows 32 ビット・データベース・サーバー
25 [4 - 60 000]

単位 ページ (4 KB)

割り振られるタイミング 最初のアプリケーションがデータベースに接続する時

解放されるタイミング 最後のアプリケーションがデータベースから切り離される時

このパラメーターは、ロック・リストに割り振られているストレージ量を示します。データベースごとに 1 つのロック・リストが存在し、ロック・リストには、データベースに同時に接続しているすべてのアプリケーションが保持しているロックが含まれています。ロッキングは、複数のアプリケーションがデータベース内にあるデータに並行アクセスするのをコントロールするために、データベース・マネージャーが使用するメカニズムです。行と表の両方がロックの対象です。データベース・マネージャーは Internal Lock を獲得する場合もあります。

このパラメーターはオンラインで変更できますが、オンラインでは値を大きくすることができるだけで、値を小さくすることはできません。 *locklist* の値を小さくしたい場合は、データベースを再活動化する必要があります。

32 ビットのプラットフォームの場合は、それぞれのロックには、そのオブジェクトで他のロックが保持されているかどうかによって、ロック・リストの 40 バイトまたは 80 バイトが必要です。

- 他にロックが保留されていないオブジェクトのロックを保留するには、 80 バイトが必要です。
- 既存のロックが保持されているオブジェクトのロックを記録するには、 40 バイトが必要です。

64 ビットのプラットフォームの場合は、それぞれのロックには、そのオブジェクトで他のロックが保持されているかどうかによって、ロック・リストの 64 バイトまたは 128 バイトが必要です。

- 他にロックが保留されていないオブジェクトのロックを保留するには、128 バイトが必要です。
- 既存のロックが保持されているオブジェクトのロックを記録するには、64 バイトが必要です。

1 つのアプリケーションが使用するロック・リストのパーセント (%) が *maxlocks* に達すると、データベース・マネージャーは、そのアプリケーションが保持するロックに対して (以下に記述)、行から表にロック・エスカレーションを行います。エスカレーション処理そのものにはそれほど時間がかかりませんが、(個別の行に対するロッキングに対して) 表全体のロッキングは並列処理を減少させ、影響を受けた表に対する後続のアクセスのために、データベース・パフォーマンス全体が低下する可能性があります。推奨されるロック・リスト・サイズのコントロール方法は、以下のとおりです。

- ロックを解放するために頻繁に COMMIT を実行します。
- 多くの更新を実行するときには、(SQL LOCK TABLE ステートメントを使用して) 更新前に表全体をロックします。これは 1 つのロックのみを使用し、他のロックが更新に干渉しないようにしますが、データの並列処理は減少します。

  また、ALTER TABLE ステートメントの LOCKSIZE オプションを使用して、特定の表のロッキング方法をコントロールすることもできます。

  反復可能読み取り分離レベルを使用すると、結果として表ロックになってしまう場合があります。

- 保持された共有ロック数の減少が可能な場合は、カーソル固定分離レベルを使用します。 アプリケーション保全性要件と折り合わない場合は、ロッキングの量をさらに減らすために、カーソル固定ではなく非コミット読み取りを使用してください。

ロック・リストがいっぱいになると、ロック・エスカレーションが行のロックよりも表のロックを多く行うので、パフォーマンスが低下する場合があり、これにより、データベースの共有オブジェクトにおける並列処理が低下します。さらに、アプリケーション間のデッドロックが増える可能性があり (すべてのアプリケーションが、限られた数の表ロックを待つため)、これによりトランザクションがロールバックされることになります。データベースに対するロック要求の最大数に達すると、アプリケーションが SQLCODE -912 を受け取ります。

推奨 : ロック・エスカレーションによってパフォーマンスの問題が生じている場合、このパラメーターか *maxlocks* パラメーターの値を増やす必要があります。データベース・システム・モニターを使用すると、ロック・エスカレーションが起きているかどうかを判別できます。 *lock_escals (ロック・エスカレーション)* モニター・エレメントを参照してください。

以下のステップは、ロック・リストに必要なページ数を決定するのに役立ちます。

1. ロック・リストのサイズの下限を計算します。ユーザー環境により、以下の計算式の中からひとつを使用して計算します。
   a. `(512 * x * maxappls) / 4096`
   b. コンセントレーターが使用可能になっている場合。

      `(512 * x * max_coordagents) / 4096`
   c. コンセントレーターが使用可能になっているパーティション・データベースの場合。

      `(512 * x * max_coordagents * データベース・パーティションの数) / 4096`

   ただし、512 はアプリケーション当たりの平均ロック数の見積もりであり、x は、既存のロックがあるオブジェクトに対するそれぞれのロックに必要なバイト数 (32 ビット・プラットフォームでは 40 バイト、64 ビット・プラットフォームでは 64 バイト) です。
2. ロック・リスト・サイズの上限を計算します。

   `(512 * y * maxappls) / 4096`

   y はオブジェクトに対する最初のロックに必要なバイト数です。 (32 ビットのプラットフォームの場合は 80 バイトで、64 ビットのプラットフォームの場合は 128 バイトです。)
3. データに対する並列量を見積もり、また計算した上限と下限の間になるように、予測に基づいて *locklist* の初期値を選択します。
4. 以下に記述されているように、データベース・システム・モニターを使用してこのパラメーターの値を調整します。

データベース・システム・モニターを使用すると、指定したトランザクションによって保持される最大ロック数を判別することができます。*locks_held_top (保留されているロックの最大数)* モニター・エレメントを参照してください。

この情報は、アプリケーションあたりのロックの見積数の検査または調整に役立ちます。この妥当性検査を実行するには、モニター情報がアプリケーション・レベルではなく、トランザクション・レベルで提供されていることに注意して、複数のアプリケーションをサンプルとする必要があります。

また、*maxappls* を増やしたか、または実行中のアプリケーションが頻繁にコミットを実行していない場合は、*locklist* の増加が必要になる場合もあります。

このパラメーターを変更した場合は、アプリケーションの再バインド (REBIND コマンドを使用) を考慮してください。

関連資料:

- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 「システム・モニター *ガイドおよびリファレンス*」の『lock_escals ロック・エスカレーション数 : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『locks_held_top ロック保持最大数 : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『REBIND コマンド』

logbufsz - ログ・バッファー・サイズ

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲]

32 ビット・プラットフォーム
8 [4 — 4 096]

64 ビット・プラットフォーム
8 [4 — 65 535]

単位 ページ (4 KB)

このパラメーターによって、*dbheap* パラメーターによって定義されるデータベース・ヒープの中から、ログ・バッファーのサイズを指定します。ログ・バッファーはログ・レコードをディスクに書き込む前に、ログ・レコードのバッファーとして使用されます。以下のいずれかのときに、ログ・レコードがディスクに書き込まれます。

- *mincommit* 構成パラメーターによって定義されたように、トランザクションのコミット、またはトランザクションのグループのコミットが行われたとき。
- ログ・バッファーがいっぱいになったとき。

- その他のいくつかの内部データベース・マネージャー・イベントの結果として。

また、このパラメーターは *dbheap* パラメーター以下でなければなりません。ログ・レコードのバッファリングは、ログ・レコードのディスクへの書き込み頻度が少なくなり、一度により多くのログ・レコードが書き込まれるため、より効率的なロギング・ファイル入出力になります。

推奨: 専用ログ・ディスクに相当数の読み取り活動がある場合、またはディスクの使用率が高い場合、バッファー域のサイズを大きくしてください。 このパラメーターの値を増やす場合は、ログ・バッファー域が、 *dbheap* パラメーターでコントロールされるスペースを使用するので、*dbheap* パラメーターも考慮してください。

データベース・システム・モニターを使用すると、特定トランザクション (または作業単位) で使用されたログ・バッファー・スペースの量を判別できます。*log_space_used* (使用される作業単位ログ・スペース) モニター・エレメントを参照してください。

関連資料:

- 469 ページの『mincommit - グループ化するコミット数』
- 389 ページの『dbheap - データベース・ヒープ』
- 「システム・モニター *ガイドおよびリファレンス*」の『uow_log_space_used 作業単位ログ・スペース : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

pckcachesz - パッケージ・キャッシュ・サイズ

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
伝搬クラス	即時
デフォルト[範囲]	
	32 ビット・プラットフォーム -1 [-1, 32 — 128 000]
	64 ビット・プラットフォーム -1 [-1, 32 — 524 288]
単位	ページ (4 KB)
割り振られるタイミング	データベースの初期化時
解放されるタイミング	データベースが停止されるとき

このパラメーターは、データベース共用メモリーから割り振られ、データベース上の静的および動的 SQL ステートメントのセクションをキャッシュするために使用

されます。パーティション・データベース・システムでは、データベース・パーティションごとに 1 つのパッケージ・キャッシュがあります。

パッケージをキャッシュすると、データベース・マネージャーの場合はパッケージを再ロードするときにシステム・カタログにアクセスする必要がなくなるために内部オーバーヘッドが減少し、また動的 SQL の場合はコンパイルする必要がなくなることによって、内部オーバーヘッドが減少します。セクションは、以下のいずれかの状況になるまで、パッケージ・キャッシュ内に保持されます。

- データベースが停止されるとき
- パッケージまたは動的 SQL ステートメントが無効にされる時
- キャッシュがスペースを使い果たした時

静的または動的 SQL ステートメントのセクションのこのキャッシュは、データベースに接続されたアプリケーションによって同じステートメントが複数回使用される場合に、特にパフォーマンスを改善することができます。 これは、トランザクション処理アプリケーションでは特に重要です。

デフォルト (-1) を使うことにより、ページ割り振りの計算に使用される値は、*maxappls* 構成パラメーターに指定された値の 8 倍になります。 *maxappls* の 8 倍が 32 よりも小さい場合は、例外となります。この場合、デフォルト値 -1 を指定すると *pckcachesz* が 32 に設定されます。

推奨: このパラメーターを調整するときは、パッケージ・キャッシュ用に予約されている余分なメモリーが、バッファー・プールまたはカタログ・キャッシュなどの別の目的のために割り振られた場合にさらに有効に使用できるかどうかを考慮する必要があります。 上記の理由により、このパラメーターの調整時には、ベンチマーク技法を使用してください。

このパラメーターの調整は、いくつかのセクションが最初に使用され、その後は 2、3 のセクションだけが繰り返し実行される場合に特に重要です。 キャッシュが大きすぎると、最初のセクションのコピーを保留している分だけメモリーが無駄になっています。

これらのモニター・エレメントは、この構成パラメーターを調整すべきかどうかの判別に役立ちます。

- *pkg_cache_lookups* (パッケージ・キャッシュ参照)
- *pkg_cache_inserts* (パッケージ・キャッシュ挿入)
- *pkg_cache_size_top* (パッケージ・キャッシュ最高水準点)
- *pkg_cache_num_overflows* (パッケージ・キャッシュ・オーバーフロー)

注: パッケージ・キャッシュは作業キャッシュであるため、このパラメーターをゼロに設定することはできません。 現在実行されている SQL ステートメントのすべてのセクションを保留するには、このキャッシュに十分なメモリーが割り振られていなければなりません。現行の必要スペース以上のスペースが割り振られている場合、セクションがキャッシュされます。これらのセクションは、次に必要とされるときにはロードやコンパイルをせずに実行できます。

pckcachesz パラメーターによって指定された制限は柔軟性のある制限です。メモリーがデータベース共有セットでまだ使用可能な場合、必要であればこの制

限を超えることが可能です。パッケージ・キャッシュが一番大きくなったサイズを判別するには、*pkg_cache_size_top* モニター・エレメントを使用し、*pckcachesz* パラメーターによって指定された制限を超えた回数を判別するには、*pkg_cache_num_overflows* モニター・エレメントを使用することができます。

関連資料:

- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 「システム・モニター *ガイドおよびリファレンス*」の『pkg_cache_lookups パッケージ・キャッシュ参照 : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『pkg_cache_inserts パッケージ・キャッシュ挿入 : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「システム・モニター *ガイドおよびリファレンス*」の『pkg_cache_num_overflows パッケージ・キャッシュ・オーバーフロー : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『pkg_cache_size_top パッケージ・キャッシュの最高水準点 : モニター・エレメント』

sheapthres_shr - 共用ソートのソート・ヒープのしきい値

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲]

32 ビット・プラットフォーム
sheapthres [250 — 2 097 152]

64 ビット・プラットフォーム
sheapthres [250 — 2 147 483 647]

単位 ページ (4 KB)

このパラメーターは、任意の一時点にソート用として使用できるデータベース共用メモリーの合計量に対するハード制限を表します。アクティブ共有ソート用の共用メモリーの合計量がこの限度に達すると、後続のソートは失敗します (SQL0955C)。*sheapthres_shr* の値が 0 の場合は、共有ソート・メモリーのしきい値は、*sheapthres* データベース・マネージャー構成パラメーターの値に等しく、専用ソートのソート・メモリーしきい値を表す場合にも使用されます。*sheapthres_shr* の値が非ゼロの場合は、この非ゼロ値が共有ソート・メモリーしきい値として使用されます。

sheapthres_shr に意味があるのは、次の 2 つの場合だけです。

- *intra_parallel* データベース・マネージャー構成パラメーターが *yes* に設定されている場合。*intra_parallel* が *no* に設定されているときは、共有ソートはないからです。
- コンセントレーターがオンの場合 (つまり、*max_connections* が *max_coordagents* より大のとき)。WITH HOLD オプションを指定して宣言されたカーソルを使用するソートは、共用メモリーから割り振られるからです。

関連資料:

- 408 ページの『sheapthres - ソート・ヒープしきい値』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

util_heap_sz - ユーティリティー・ヒープ・サイズ

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
伝搬クラス	即時
デフォルト[範囲]	5000 [16 - 524 288]
単位	ページ (4 KB)
割り振られるタイミング	データベース・マネージャー・ユーティリティーにより必要とされたとき
解放されるタイミング	ユーティリティーがメモリーを必要としなくなったとき

このパラメーターは、BACKUP、RESTORE、および LOAD (ロード・リカバリーを含む) ユーティリティーによって同時に使用できるメモリーの最大量を示します。

推奨: ユーティリティーがスペースを使い尽くした場合は、この値を大きくする必要がありますが、そうならない限り、デフォルト値を使用してください。システム上のメモリーが制約されている場合は、このパラメーターの値を小さくして、データベース・ユーティリティーによって使用されるメモリーを制限できます。このパラメーターを低過ぎる値に設定したためにオーバーフロー域で使用可能なメモリーがなくなっている場合は、複数のユーティリティーを同時に実行できない可能性があります。必要に応じて、このパラメーターを動的に更新してください。ユーティリティーの数が少ない場合は、このパラメーターを小さい値に設定します。ユーティリティーの数が多い場合、またはメモリーの使用量が大きいユーティリティーの場合は、このパラメーターを大きい値に設定します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

アプリケーション共用メモリー

以下のパラメーターは、アプリケーションのために働くすべてのエージェント (調整およびサブエージェント) が使用する作業域を指定します。

- 『app_ctl_heap_sz - アプリケーション・コントロール・ヒープ・サイズ』
- 399 ページの『appgroup_mem_sz - アプリケーション・グループ・メモリー・セットの最大サイズ』
- 401 ページの『groupheap_ratio - アプリケーション・グループ・ヒープ用メモリーのパーセント』

app_ctl_heap_sz - アプリケーション・コントロール・ヒープ・サイズ

構成タイプ
データベース

パラメーター・タイプ
構成可能

デフォルト[範囲]

ローカルとリモート・クライアントを持つデータベース・サーバー
128 [1-64 000]

ローカル・クライアントを持つデータベース・サーバー
64 [1-64 000] (非 UNIX プラットフォームの場合)

128 [1-64 000] (UNIX ベースのプラットフォームの場合)

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
512 [1-64 000]

単位
ページ (4 KB)

割り振られるタイミング
アプリケーション開始時期

解放されるタイミング
アプリケーション完了時期

パーティション・データベースの場合、およびパーティション内並列処理が使用可能 (intra_parallel=ON) の非パーティション・データベースの場合は、このパラメーターは、アプリケーション用として割り振られる共用メモリー領域の平均サイズを指定します。パーティション内並列処理が使用不可 (intra_parallel=OFF) の非パーティション・データベースの場合は、これはヒープとして割り振られる最大専用メモリー・サイズです。それぞれのパーティションごとに 1 つの接続に 1 つずつアプリケーション・コントロール・ヒープがあります。

アプリケーション・コントロール・ヒープが必要なのは、主として、同一要求のために作動するエージェント間で情報を共用するためです。このヒープの使用量が最小になるのは、非パーティション・データベースで、1 に等しい並列処理の多重度で照会を実行しているときです。

このヒープは、宣言済み一時表の記述子情報を保管することにも使用されます。明示的にドロップされていないすべての宣言済み一時表の記述子情報はこのヒープのメモリーに保持され、宣言済み一時表がドロップされるまでドロップすることはできません。

推奨: 最初はデフォルト値で開始してください。複雑なアプリケーションを実行する場合、多数のデータベース・パーティションを含むシステムである場合、または宣言済み一時表を使用する場合は、値をより高く設定しなければならないことがあります。必要とされるメモリーの量は、並行してアクティブな宣言済み一時表の数とともに増加します。多くの列を持つ宣言済み一時表の表記述子サイズは、少ない列を持つ表のものよりも大きくなります。そのため、アプリケーションの宣言済み一時表が多数の列を持つと、アプリケーション・コントロール・ヒープへの要求も増えます。

関連資料:

- 525 ページの『intra_parallel - パーティション内並列処理機能の使用可能化』
- 404 ページの『applheapsz - アプリケーション・ヒープ・サイズ』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 399 ページの『appgroup_mem_sz - アプリケーション・グループ・メモリー・セットの最大サイズ』
- 401 ページの『groupheap_ratio - アプリケーション・グループ・ヒープ用メモリーのパーセント』

appgroup_mem_sz - アプリケーション・グループ・メモリー・セットの最大サイズ

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲]

ローカル・クライアントを持つ UNIX データベース・サーバー (32 ビット HP-UX を除く)
20 000 [1 - 1 000 000]

32 ビット HP-UX

- 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

10 000 [1 - 1 000 000]

ローカル・クライアントを持つ Windows データベース・サーバー

10 000 [1 - 1 000 000]

ローカル・クライアントおよびリモート・クライアントを持つデータベース・サーバー (32 ビット HP-UX を除く)

30 000 [1 - 1 000 000]

ローカル・クライアントおよびリモート・クライアントを持つパーティション化されたデータベース・サーバー (32 ビット HP-UX を除く)

40 000 [1 - 1 000 000]

単位

ページ (4 KB)

このパラメーターでは、アプリケーション・グループ共用メモリー・セグメントのサイズを決定します。同じアプリケーションを使用するエージェント間で共有する必要がある情報は、アプリケーション・グループ共用メモリー・セグメントに保管されます。

パーティション・データベース、またはパーティション内並列処理が使用可能か、コンセントレーターが使用可能の非パーティション・データベースでは、複数のアプリケーションが 1 つのアプリケーション・グループを共有します。アプリケーション・グループ共用メモリー・セグメントが 1 つ、アプリケーション・グループに割り振られます。アプリケーション・グループ共用メモリー・セグメント内では、各アプリケーションにそれぞれユニークのアプリケーション・コントロール・ヒープがあり、すべてのアプリケーションで 1 つのアプリケーション・グループ共有ヒープを共有します。

1 つのアプリケーション・グループ内のアプリケーションの数は、次のようにして計算されます。

```
appgroup_mem_sz / app_ctl_heap_sz
```

アプリケーション・グループ共有ヒープ・サイズは、次のようにして計算されます。

```
appgroup_mem_sz * groupheap_ratio / 100
```

各アプリケーション・コントロール・ヒープのサイズは、次のようにして計算されます。

```
app_ctl_heap_sz * (100 - groupheap_ratio) / 100
```

推奨: パフォーマンス上の問題が検出されない限り、このパラメーターのデフォルト値を変更しないでください。

関連資料:

- 398 ページの『app_ctl_heap_sz - アプリケーション・コントロール・ヒープ・サイズ』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 401 ページの『groupheap_ratio - アプリケーション・グループ・ヒープ用メモリーのパーセント』

groupheap_ratio - アプリケーション・グループ・ヒープ用メモリーのパーセント

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	70 [1 - 99]
単位	パーセント

このパラメーターは、アプリケーション・グループ共有ヒープ専用メモリーが、アプリケーション・コントロール共用メモリー・セットに占めるパーセンテージを指定します。

このパラメーターは、コンセントレーターがオフに設定され、パーティション内並列処理が使用不可になっている非パーティション・データベースには、まったく影響がありません。

推奨: パフォーマンス上の問題が検出されない限り、このパラメーターのデフォルト値を変更しないでください。

関連資料:

- 398 ページの『app_ctl_heap_sz - アプリケーション・コントロール・ヒープ・サイズ』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 399 ページの『appgroup_mem_sz - アプリケーション・グループ・メモリー・セットの最大サイズ』

エージェント専用メモリー

次のパラメーターは、各データベース・エージェントで使用されるメモリーの量を制御します。

- 『agent_stack_sz - エージェント・スタック・サイズ』
- 404 ページの『applheapsz - アプリケーション・ヒープ・サイズ』
- 405 ページの『min_priv_mem - 最小コミット済み専用メモリー』
- 406 ページの『priv_mem_thresh - 専用メモリーしきい値』
- 407 ページの『query_heap_sz - 照会ヒープ・サイズ』
- 408 ページの『sheapthres - ソート・ヒープしきい値』
- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 411 ページの『stat_heap_sz - 統計ヒープ・サイズ』
- 412 ページの『stmtheap - ステートメント・ヒープ・サイズ』

agent_stack_sz - エージェント・スタック・サイズ

構成タイプ
データベース・マネージャー

適用
- ローカルとリモート・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
構成可能

デフォルト[範囲]
16 [8 - 1000]

単位
ページ (4 KB)

割り振られるタイミング
エージェントが、アプリケーションの作業を行うように初期化されるとき

解放されるタイミング
エージェントがアプリケーションの作業を完了するとき

エージェント・スタックは、エージェントごとに DB2 によって割り振られる仮想メモリーです。このメモリーは、SQL ステートメントを処理するのに必要な場合にコミットされます。照会が複雑になるほど、単純な照会が使用するスペースと比較して、より多くのスタック・スペースを使用します。単純照会に使用されるスペースと比較して、照会がより複雑になればより多くのスタック・スペースが使用されます。

このパラメーターは、Windows 環境でそれぞれのエージェントごとに初期コミット済みスタック・サイズを設定する場合に使用します。デフォルトでは、それぞれのエージェント・スタックは、最大でデフォルトの予約スタック・サイズである 256 KB (64 4 KB ページ) まで増やせます。ほとんどのデータベース操作には、この上限で十分です。ただし、準備中の SQL ステートメントが大きい場合は、エージェントがスタック・スペースを使い尽くす可能性があり、システムがオーバーフロー例外 (0xC000000D) を生成することになります。こうなったときは、エラーがリカバリー不能のため、サーバーはシャットダウンします。

エージェント・スタック・サイズは、*agent_stack_sz* をデフォルトの予約スタック・サイズ 64 ページよりも大きい値に設定することで、大きくすることができます。*agent_stack_sz* の値は、デフォルトの予約スタック・サイズよりも大きくなると、Windows オペレーティング・システムによって、その値に最も近い 1 MB の倍数に丸められることに注意してください。したがって、エージェント・スタック・サイズを 128 4 KB ページに設定すると、実際にはそれぞれのエージェントごとに 1 MB が予約されることになります。 *agent_stack_sz* の値をデフォルトの予約スタック・サイズよりも小さい値に設定した場合は、必要なら、スタックが最大でデフォルトの予約スタック・サイズまで大きくなるため、デフォルトの予約スタック・サイズが最大限度に影響することはありません。この場合は、*agent_stack_sz* の値は、エージェントが作成されるときのスタック用初期コミット済みメモリーになります。

デフォルトの予約スタック・サイズは、db2syscs.exe ファイルに関するヘッダー情報を変更する db2hdr ユーティリティーを使用して変更できます。デフォルトの予約スタック・サイズを変更すると、*agent_stack_sz* の変更では、エージェントのスタック・サイズが影響を受けるだけであるのに対して、すべてのスレッドに影響が生じます。db2hdr ユーティリティーを使用してデフォルトのスタック・サイズを変更すると、より優れた細分性が得られ、そのためにスタック・サイズを最少所要スタック・サイズに設定できるという利点があります。ただし、db2syscs.exe に対する変更を有効にするためには、DB2 をいったん停止してから、再始動する必要があります。

推奨: ほとんどの場合、デフォルトのスタック・サイズを使用することができます。現在の環境に非常に複雑な照会が多く存在する場合にのみ、このパラメーターの値を増やす必要があります。

環境が以下のいずれかにあてはまる場合は、他のクライアントがより多くのアドレス・スペースを使用できるようにするために、スタック・サイズを減らすことができます。

- 複雑な照会を持たない、単純なアプリケーション (たとえば単純な OLTP) だけがある場合
- 相対的に多くの (たとえば 100 より多い) 並行クライアントを必要とする場合

エージェント・スタック・サイズおよび並行クライアントの数は、逆比例の関係にあります。大きなスタック・サイズは、実行できる並行クライアントの潜在数を減らします。こうなるのは、Windows プラットフォームではアドレス・スペースが限定されているためです。

このパラメーターは、UNIX ベースのプラットフォームには適用されません。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

applheapsz - アプリケーション・ヒープ・サイズ

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲]

ローカルとリモート・クライアントを持つ 32 ビット・データベース・サーバー
256 [16 - 60 000]

ローカルとリモート・クライアントを持つ 64 ビット・データベース・サーバー
256 [16 - 60 000]

ローカルとリモート・クライアントを持つ 32 ビット・パーティション・データベース・サーバー
64 [16 - 60 000]

ローカルとリモート・クライアントを持つ 64 ビット・パーティション・データベース・サーバー
128 [16 - 60 000]

単位 ページ (4 KB)

割り振られるタイミング エージェントが、アプリケーションの作業を行うように初期化されるとき

解放されるタイミング エージェントがアプリケーションの作業を完了するとき

このパラメーターは、データベース・マネージャーが特定のエージェントあるいはサブエージェントのために使用することのできる専用メモリー・ページの数を定義します。

エージェントまたはサブエージェントがアプリケーション用に初期化されると、ヒープが割り振られます。割り振られる量は、エージェントまたはサブエージェントに指定された要求を処理するのに最低限必要な量です。 エージェントまたはサブエージェントが、大きな SQL ステートメントを処理するためにさらにヒープ・スペースを要求した場合は、データベース・マネージャーが必要に応じて、このパラメーターで指定された最大値までメモリーを割り振ります。

注: パーティションのあるデータベース環境では、アプリケーション・コントロール・ヒープ (*app_ctl_heap_sz*) が、エージェントとサブエージェントの SQL ステートメントの実行セクションのコピーを保管するのに使用されます。 ただし、SMP サブエージェントは、他の全ての環境でエージェントを実行したときと同様に、*applheapsz* を使用します。

推奨: アプリケーションが、アプリケーション・ヒープに十分なストレージがないことを示すエラーを受け取った場合は、このパラメーターの値を増やしてください。

アプリケーション・ヒープ (*applheapsz*) は、エージェント専用メモリーから割り振られます。

関連資料:

- 398 ページの『app_ctl_heap_sz - アプリケーション・コントロール・ヒープ・サイズ』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

min_priv_mem - 最小コミット済み専用メモリー

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 32 [32 - 112 000]

単位 ページ (4 KB)

割り振られるタイミング データベース・マネージャーが始動されるとき

解放されるタイミング データベース・マネージャーが停止されるとき

このパラメーターは、データベース・マネージャー・インスタンスが始動 (db2start) したときに、データベース・サーバー・プロセスが専用仮想メモリーとして予約されるページ数を指定します。サーバーがより大きな専用メモリーを必要とする場合は、必要時に、サーバーがオペレーティング・システムからメモリーを取得しようとします。

このパラメーターは、UNIX ベースのシステムには適用されません。

推奨 : デフォルト値を使用してください。

データベース・サーバーにより多くのメモリーをコミットしたい場合、このパラメーターの値を変更すればよいだけです。このアクションにより、割り振り時間が節約されます。しかし、非 DB2 アプリケーションのパフォーマンスに影響を与える可能性があるため、その値を高く設定しすぎないように注意してください。

関連資料:

- 406 ページの『priv_mem_thresh - 専用メモリーしきい値』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

priv_mem_thresh - 専用メモリーしきい値

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 20 000 [-1; 32 - 112 000]

単位 ページ (4 KB)

このパラメーターは、開始される新しいエージェントが使用できる状態にある、未使用のエージェント専用メモリー量の判別に使用されます。このパラメーターは、UNIX ベースのプラットフォームには適用されません。

-1 の値を指定すると、このパラメーターは、*min_priv_mem* パラメーターの値を使用します。

推奨: このパラメーターを設定する場合、クライアント接続/切断パターンと、同じマシン上の他の複数のプロセスのメモリー要件を考慮してください。

多くのクライアントがデータベースに同時に接続する期間が短い場合、しきい値を高くすると未使用メモリーがコミット解除されなくなり、他のプロセスで使用できなくなります。この場合、十分なメモリー管理が行われず、メモリーを必要とする他のプロセスに影響を与えます。

並行クライアントの数があまり変わらず、その値の範囲内で頻繁に変動する場合、このしきい値を高くすることによってクライアント・プロセスの為にメモリーを確保し、メモリーの割り振りと割り振り解除のオーバーヘッドを低減させることができます。

関連資料:

- 405 ページの『min_priv_mem - 最小コミット済み専用メモリー』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

query_heap_sz - 照会ヒープ・サイズ

構成タイプ	データベース・マネージャー
適用	• ローカルとリモート・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	構成可能
デフォルト[範囲]	1000 [2 - 524 288]
単位	ページ (4 KB)
割り振られるタイミング	アプリケーション (ローカルまたはリモートの) がデータベースに接続するとき
解放されるタイミング	アプリケーションがデータベースから切断されたか、またはインスタンスから切断されたとき

このパラメーターは、照会ヒープに割り振ることができるメモリーの**最大**数を指定します。照会ヒープは、各照会をエージェントの専用メモリーに格納するために使用されます。各照会の情報は、入力と出力 SQLDA、ステートメント・テキスト、SQLCA、パッケージ名、作成者、セクション番号、および整合性トークンで構成されます。このパラメーターは、アプリケーションに、エージェント内で不要に大きな仮想メモリーを消費させないために提供されています。

照会ヒープは、ブロック・カーソルに割り振られるメモリー用としても使用されます。このメモリーは、カーソル・コントロール・ブロックとその内容を表現する出力 SQLDA から構成されます。

割り振られる初期照会ヒープは、*aslheapsz* パラメーターによって指定されたアプリケーション・サポート層ヒープと同じサイズです。照会ヒープのサイズは 2 以上でなければならず、*aslheapsz* パラメーター以上でなければなりません。この照会ヒープの大きさが不十分なために指定された要求を処理できない場合は、要求に必要なサイズまで
(*query_heap_sz* を超えない範囲で) 再割り振りが行われます。この新しい照会ヒープが *aslheapsz* の 1.5 倍を超える大きさの場合、照会が終了したときに照会ヒープは *aslheapsz* のサイズまで再度割り振られます。

推奨: ほとんどの場合、デフォルト値が効率的です。最小値として、*query_heap_sz* を少なくとも *aslheapsz* の 5 倍より大きい値に設定してください。これにより、*aslheapsz* より大きな照会が使用可能になり、指定時にオープンされる 3 つまたは 4 つのブロック・カーソルに追加メモリーが提供されます。

とても大きな LOB を持っている場合、このパラメーターの値を増やして、照会ヒープをそれらの LOB を収容する大きさにすることが必要になる場合があります。

関連資料:

- 413 ページの『aslheapsz - アプリケーション・サポート層ヒープ・サイズ』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

sheapthres - ソート・ヒープしきい値

構成タイプ
データベース・マネージャー

適用
- ローカルとリモート・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
構成可能

デフォルト[範囲]

UNIX 32 ビット・プラットフォーム
20 000 [250 — 2 097 152]

Windows プラットフォーム
10 000 [250 — 2 097 152]

64 ビット・プラットフォーム
20 000 [250 — 2 147 483 647]

単位
ページ (4 KB)

専用および共有ソートは、2 つの異なるメモリー・ソースからメモリーを使用します。共有ソート・メモリー領域のサイズは、データベースへの最初の接続時に *sheapthres* の値に基づいて静的に事前決定されます。 専用ソート・メモリー領域のサイズは制限されていません。

sheapthres パラメーターは、専用および共有ソートで以下のように異なった使い方をされます。

- 専用ソートの場合、このパラメーターは指定された時間に専用ソートによって使用されるメモリーの合計量における、インスタンス担任の単位の緩やかな 制限です。インスタンスの専用ソートのメモリー使用量の合計がこの制限に達すると、追加の入力専用ソート要求が相当量削減されます。
- 共有ソートの場合は、このパラメーターは、特定の時点で共有ソートによって使用されるメモリーの合計量に対するデータベース単位の厳格な制限です。この制限に達すると、共有ソート・メモリーの合計使用量が *sheapthres* によって指定された制限より下に落ちるまで、共有ソート・メモリー要求は許可されなくなりま

す。 (特定の環境で共有ソート最大値を構成する別の方法は、 *sheapthres_shr* データベース構成パラメーターを使用することです。)

ソート・ヒープを使用する操作の例としては、ハッシュ結合、動的ビットマップ (索引 ANDing および Star Join で使用される)、および表がメモリー内にある操作などがあります。

しきい値を明示的に定義すると、データベース・マネージャーが大規模なソートに対して余分なメモリー量を使用することを防げます。

非パーティション・データベース環境からパーティション・データベース環境に移行するにあたって、このパラメーターの値を大きくする理由はありません。単一データベース・パーティション環境でデータベースおよびデータベース・マネージャーの構成パラメーターを調整してあれば、ほとんどの場合、同じ値がパーティション・データベース環境にも適合します。

「ソート・ヒープしきい値」パラメーターは、データベース・マネージャー構成パラメーターとして、DB2 インスタンス全体にわたって適用されます。このパラメーターを異なるノードや異なるパーティションで異なる値に設定するには、複数の DB2 インスタンスを作成する以外に方法はありません。このためには、異なるデータベース・パーティション・グループにまたがる異なる DB2 データベースの管理が必要になります。こうした方法では、パーティション・データベース環境の利点の多くの目的が達成されないことになります。

推奨: このパラメーターをデータベース・マネージャー・インスタンスの最大の *sortheap* パラメーターの適切な倍数に設定することが理想です。このパラメーターは、**少なくとも**インスタンス内のデータベースに定義された最大の *sortheap* の 2 倍の大きさにする必要があります。

専用ソートを実行しており、システムにメモリーの制約がない場合は、このパラメーターの理想値を以下の手順で計算することができます。

1. 以下のように各データベースの通常のソート・ヒープ使用率を計算します。

```
    (typical number of concurrent agents running against the database)
  * (sortheap, as defined for that database)
```

2. 上記の結果の合計を計算すると、インスタンス内のすべてのデータベースに対する典型的な状況において、使用可能なソート合計ヒープが提供されます。

ソート・パフォーマンスとメモリー使用率を適切にバランスするためには、このパラメーターを調整するためにベンチマーク技法を使用する必要があります。

データベース・システム・モニターを使用すると、ポストしきい値ソート (*post_threshold_sorts*) モニター・エレメントを使用して、ソート・アクティビティーを追跡できます。

関連資料:

- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 「システム・モニター *ガイドおよびリファレンス*」の『post_threshold_sorts ポストしきい値ソート : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 396 ページの『sheapthres_shr - 共用ソートのソート・ヒープのしきい値』

sortheap - ソート・ヒープ・サイズ

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

伝搬クラス ステートメント境界

デフォルト[範囲]

32 ビット・プラットフォーム
256 [16 - 524 288]

64 ビット・プラットフォーム
256 [16 - 4 194 303]

単位 ページ (4 KB)

割り振られるタイミング ソートを実行するための必要に応じて

解放されるタイミング ソートの完了時

このパラメーターでは、専用ソートで使用される専用メモリー・ページの最大数、または共有ソートで使用される共用メモリー・ページの最大数を定義します。ソートが専用ソートの場合は、このパラメーターによってエージェント専用メモリーに影響が生じます。ソートが共有ソートの場合は、このパラメーターによってデータベース共用メモリーに影響が生じます。それぞれのソートには、必要に応じてデータベース・マネージャーによって割り振られる別々のソート・ヒープがあります。このソート・ヒープは、データがソートされる領域です。オプティマイザーによる指示があった場合は、オプティマイザーが提供する情報を使用して、このパラメーターによって指定されたソート・ヒープよりも小さいソート・ヒープが割り振られます。

推奨: ソート・ヒープを使用して作業する場合は、次の事項を考慮する必要があります。

- 適切な索引によってソート・ヒープの使用を最小化できます。
- ハッシュ結合バッファーおよび動的ビットマップ (索引 ANDing および Star Join で使用される) では、ソート・ヒープ・メモリーを使用します。したがって、これらの技法が使用されるときは、このパラメーターのサイズを大きくします。
- ラージ・ソートが頻繁に必要なときは、このパラメーターのサイズを大きくします。
- このパラメーターの値を大きくするときは、データベース・マネージャー構成ファイルにある *sheapthres* パラメーターも調整する必要があるかどうか調べる必要があります。
- ソート・ヒープ・サイズは、オプティマイザーがアクセス・パスを決定する際に使用します。このパラメーターを変更した場合は、アプリケーションの再バインド (REBIND コマンドを使用) を考慮してください。

関連資料:

- 408 ページの『sheapthres - ソート・ヒープしきい値』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『REBIND コマンド』
- 396 ページの『sheapthres_shr - 共用ソートのソート・ヒープのしきい値』

stat_heap_sz - 統計ヒープ・サイズ

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	4384 [1096 - 524 288]
単位	ページ (4 KB)
割り振られるタイミング	RUNSTATS ユーティリティーが開始されたとき
解放されるタイミング	RUNSTATS ユーティリティーが完了したとき

このパラメーターは、RUNSTATS コマンドによる統計の収集の際に使用される、ヒープの**最大**サイズを示します。

推奨: 分散統計がまったく収集されないときや、分散統計が相対的に幅の狭い表についてしか収集されていないときは、デフォルト値が適切です。分散統計が収集されているときは、最小値は推奨**できません**。ヒープに収まるのが、1 列または 2 列の表だけになるからです。

このパラメーターは、統計が収集される列の数を基にして調整する必要があります。相対的に列数の少ない、幅の狭い表の場合は、収集される分散統計に必要とされるメモリーは少なくなります。列数の多い、幅の広い表の場合は、かなり多くのメモリーが必要です。非常に幅の広い、大きなヒープを必要とする表に関する分散統計を収集する場合、システム・アクティビティーが低いときに統計を収集して、他のユーザーのメモリーの使用を妨げないようにすることを検討してください。

関連資料:

- 507 ページの『num_freqvalues - 保存される頻度値の数』
- 508 ページの『num_quantiles - 列の変位値の数』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RUNSTATS コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

stmtheap - ステートメント・ヒープ・サイズ

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
伝搬クラス	ステートメント境界
デフォルト[範囲]	2048 [128 - 65 535]
単位	ページ (4 KB)
割り振られるタイミング	それぞれのステートメントごとに、プリコンパイル時またはバインド時
解放されるタイミング	それぞれのステートメントのプリコンパイルまたはバインドが完了した時点

ステートメント・ヒープは、SQL ステートメントのコンパイル時に SQL コンパイラー用のワークスペースとして使用されます。このパラメーターでは、このワークスペースのサイズを指定します。

この作業域は、永続的に割り振られた状態のままになっているのではなく、SQL ステートメントが処理されるごとに、その割り振りと解放が行われます。この作業域は、動的 SQL ステートメントの場合は、プログラムの実行時に使用されるのに対して、静的 SQL ステートメントの場合は、バインド処理時に使用され、プログラム実行時には使用されないことに注意してください。

推奨: ほとんどの場合、このパラメーターのデフォルト値が使用できます。 SQL ステートメントが非常に大きく、データベース・マネージャーがステートメントの最適化を試みているときにエラー (ステートメントが複雑過ぎるという) を出した場合は、エラー状態が解消されるまで、規則的な増分 (たとえば、256 や 1024 など) によってこのパラメーターの値を大きくする必要があります。

関連資料:

- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 404 ページの『applheapsz - アプリケーション・ヒープ・サイズ』
- 411 ページの『stat_heap_sz - 統計ヒープ・サイズ』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

エージェント/アプリケーション通信メモリー

次のパラメーターは、ユーザー・アプリケーションとエージェント・プロセスとの間でデータを渡すために割り振られるメモリーの量を制御します。

- 413 ページの『aslheapsz - アプリケーション・サポート層ヒープ・サイズ』
- 414 ページの『min_dec_div_3 - 10 進数除算の位取り 3』
- 416 ページの『rqrioblk - クライアント入出力ブロック・サイズ』

aslheapsz - アプリケーション・サポート層ヒープ・サイズ

構成タイプ	データベース・マネージャー
適用	• ローカルとリモート・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	構成可能
デフォルト[範囲]	15 [1 - 524 288]
単位	ページ (4 KB)
割り振られるタイミング	データベース・マネージャー・エージェント・プロセスがローカル・アプリケーションで開始するとき
解放されるタイミング	データベース・マネージャー・エージェント・プロセスが終了したとき

アプリケーション・サポート層ヒープは、ローカル・アプリケーションとその関連エージェントの間の通信バッファーを示します。このバッファーはデータベース・マネージャー・エージェントが開始するたびに共用メモリーとして割り振られます。

データベース・マネージャーへの要求、またはそれに関連する応答がバッファーに適合しない場合は、バッファーが 2 つ以上の送受信の組み合わせに分割されます。このバッファーのサイズは、単一の送受信の組み合わせを使用する大多数の要求を処理するように、設定する必要があります。要求のサイズは、次のものを保存するのに必要なストレージに基づいて決まります。

- 入力 SQLDA
- SQLVAR 内のすべての関連データ
- 出力 SQLDA
- 通常は 250 バイトを超えないその他のフィールド

このパラメーターは、この通信バッファーだけでなく、それ以外に次の 2 つの目的でも使用されます。

- ブロック・カーソルがオープンされているとき、入出力ブロック・サイズを決定するのに使用されます。ブロック・カーソル用のこのメモリーは、アプリケーションの専用アドレス・スペースから割り振られるので、それぞれのアプリケーション・プログラムごとに割り振る専用メモリーの最適量を決定する必要があります。データベース・クライアントがアプリケーションの専用メモリーからブロック・カーソル用のスペースを割り振れない場合は、非ブロッキング・カーソルがオープンされます。
- これは、エージェント・プロセスと db2fmp プロセスの間の通信のサイズを決定するのに使用されます (db2fmp プロセスは、ユーザー定義関数でも fenced スト

アード・プロシージャーでも構いません)。システム上でアクティブな各 db2fmp プロセスまたはスレッドのために、共用メモリーから何バイトかのメモリーが割り振られます。

ローカル・アプリケーションから送信されたデータは、データベース・マネージャーによって、照会ヒープから割り振られた連続したメモリーのセットに受信されます。 *aslheapsz* パラメーターは、照会ヒープの初期サイズ (ローカルおよびリモート・クライアントの両方について) の決定に使用されます。照会ヒープの最大サイズは、*query_heap_sz* パラメーターによって定義されます。

推奨: アプリケーションの要求が通常は少なく、そのアプリケーションが、メモリーに制約があるシステムで実行されている場合は、このパラメーターの値を減らすことが必要になる可能性があります。通常は非常に大きい照会をしていて複数の送受信要求を必要とし、またシステムがメモリーによる制約を受けていない場合は、このパラメーターの値を増やすことが必要になる可能性があります。

次の公式を使用して、*aslheapsz* の最小ページ数を計算します。

```
aslheapsz >= ( sizeof(入力 SQLDA)
             + sizeof(各入力 SQLVAR)
             + sizeof(出力 SQLDA)
             + 250 ) / 4096
```

ただし、sizeof(x) は、特定の入力値または出力値のページ数を計算する x のサイズ (バイト数) です。

ブロック・カーソルの数および潜在的なサイズに対するこのパラメーターの影響についても考慮する必要があります。転送される行の数が多い、またはそのサイズが大きい場合 (たとえば、データの量が 4 096 バイトより大の場合) は、行ブロックが大きければ、パフォーマンスの向上がもたらされる可能性もあります。ただし、レコード・ブロックを大きくすると、それぞれの接続ごとの作業セット・メモリーのサイズも増大するので、トレードオフが必要になります。

また、レコード・ブロックが大きくなれば、フェッチ要求も実際にアプリケーションで必要とされる数よりも多くなる可能性もあります。フェッチ要求の数は、アプリケーションの SELECT ステートメントの OPTIMIZE FOR 文節を使ってコントロールできます。

関連資料:

- 407 ページの『query_heap_sz - 照会ヒープ・サイズ』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

min_dec_div_3 - 10 進数除算の位取り 3

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲] No [Yes, No]

min_dec_div_3 データベース構成パラメーターは、SQL での 10 進数除算の位取りの計算への変更を可能にする簡易な手段として用意され、*min_dec_div_3* は「Yes」または「No」に設定できます。 *min_dec_div_3* のデフォルト値は「No」です。

min_dec_div_3 は、10 進数除算の位取りを変更するデータベース構成パラメーターです。値が「No」の場合は、位取りは 31-p+s-s' として計算されます。値が「Yes」に設定されている場合、位取りは MAX(3, 31-p+s-s') として計算されます。したがって、10 進数の除算の結果は、常に位取りが少なくとも 3 になります。精度は常に 31 です。

このデータベース構成パラメーターを変更すると、既存のデータベースで使用するアプリケーションに変更がもたらされる場合があります。これが起こる可能性があるのは、このデータベース構成パラメーターを変更することによって、10 進数除算の結果の位取りが影響を受けるような場合です。アプリケーションに影響する可能性があるシナリオをいくつか下にリストしてあります。これらのシナリオについては、既存のデータベースがあるデータベース・サーバー上の *min_dec_div_3* を変更する前に考慮する必要があります。

- ビュー列の結果の位取りの 1 つが変更された場合、設定が 1 つしかない環境内で定義されているビューは、データベース構成パラメーターの変更後に参照されると、SQLCODE -344 を出して失敗する可能性があります。メッセージ SQL0344N は再帰的共通表式を識別しますが、オブジェクト名 (最初のトークン) がビューの場合は、そのビューをドロップした上で、再度作成してこのエラーを回避する必要があります。
- 静的パッケージの振る舞いは、暗黙的あるいは明示的に、パッケージが再バインドされるまで変わりません。たとえば、値を NO から YES に変更した後は、再バインドが行われるまでは、追加の位取りの数字は結果に組み込まれない可能性があります。変更後の静的パッケージの場合は、明示的 REBIND コマンドを使用して、バインドを強制できます。
- 10 進数除算に関係するチェック制約によって、以前は受け入れられていた一部の値が制約される場合があります。そのような行は現在では制約に違反しますが、チェック制約行に関係する列の 1 つが更新されるか、IMMEDIATE CHECKED オプションを指定した SET INTEGRITY ステートメントが処理されるまでは検出されません。そのような制約のチェックを強制的に行うためには、チェック制約をドロップするために ALTER TABLE ステートメントを実行してから、ALTER TABLE ステートメントを実行して再度制約を追加します。

注: *min_dec_div_3* には、次の制限もあります。

1. コマンド `GET DB CFG FOR DBNAME` では *min_dec_div_3* の設定は表示されません。10 進数除算の結果の副次作用を監視するのが、現行設定を判別する最良の方法です。たとえば、次のステートメントを考察してみましょう。

   ```
   VALUES (DEC(1,31,0)/DEC(1,31,5))
   ```

   このステートメントが sqlcode SQL0419N を戻した場合は、データベースには *min_dec_div_3* のサポートがないか、「No」に設定されています。ステートメントが `1.000` を戻した場合は、*min_dec_div_3* は「Yes」に設定されています。

2. *min_dec_div_3* は、次のコマンドを実行したときは構成キーワードのリストに示されません: `? UPDATE DB CFG`

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

rqrioblk - クライアント入出力ブロック・サイズ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] 32 767 [4 096 - 65 535]

単位 バイト

割り振られるタイミング

- リモート・クライアント・アプリケーションが、サーバー・データベースに接続要求を発行するとき
- ブロック・カーソルがオープンされ、追加のブロックがクライアントでオープンされたとき

解放されるタイミング

- リモート・アプリケーションがサーバー・データベースから切断されたとき
- ブロック・カーソルがクローズされるとき

このパラメーターは、リモート・アプリケーションと、データベース・サーバー上のデータベース・エージェントの間の通信バッファーのサイズを指定します。データベース・クライアントがリモート・データベースへの接続を要求すると、この通信バッファーがクライアントに割り振られます。データベース・サーバーでは、接続が確立されて、サーバーでクライアントの *rqrioblk* の値を判別できるようになるまでは、最初に 32 767 バイトの通信バッファーが割り振られます。サーバーにこの値が分かった後で、クライアントのバッファーが 32 767 バイトでない場合、サーバーは、その通信バッファーを再割り振りします。

この通信バッファーに加えて、このパラメーターは、ブロック・カーソルがオープンされるときのデータベース・クライアントの入出力ブロック・サイズの決定にも使用されます。ブロック・カーソル用のこのメモリーは、アプリケーションの専用アドレス・スペースから割り振られるので、それぞれのアプリケーション・プログラムごとに割り振る専用メモリーの最適量を決定する必要があります。データベース・クライアントがアプリケーションの専用メモリーからブロック・カーソル用のスペースを割り振れない場合は、非ブロッキング・カーソルがオープンされます。

推奨: 非ブロック・カーソルの場合、1 つの SQL ステートメントによって送信されるデータ (たとえばラージ・オブジェクト・データ) が大きすぎてデフォルト値では不十分な場合には、このパラメーターの値を増やしてください。

ブロック・カーソルの数および潜在的なサイズに対するこのパラメーターの影響についても考慮する必要があります。転送される行の数が多い、またはそのサイズが大きい場合 (たとえば、データの量が 4 096 バイトより大の場合) は、行ブロックが大きければ、パフォーマンスの向上がもたらされる可能性もあります。ただし、レコード・ブロックを大きくすると、それぞれの接続ごとの作業セット・メモリーのサイズも増大するので、トレードオフが必要になります。

また、レコード・ブロックが大きくなれば、フェッチ要求も実際にアプリケーションで必要とされる数よりも多くなる可能性もあります。フェッチ要求の数は、アプリケーションの SELECT ステートメントの OPTIMIZE FOR 文節を使ってコントロールできます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

データベース・マネージャー・インスタンス・メモリー

次のパラメーターは、インスタンス・レベルで割り振られて使用されるメモリーを制御します。

- 『audit_buf_sz - 監査バッファー・サイズ』
- 418 ページの『dir_cache - ディレクトリー・キャッシュ・サポート』
- 420 ページの『instance_memory - インスタンス・メモリー』
- 421 ページの『java_heap_sz - Java インタープリター最大ヒープ・サイズ』
- 422 ページの『mon_heap_sz - データベース・システム・モニター・ヒープ・サイズ』

audit_buf_sz - 監査バッファー・サイズ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 0 [0 - 65 000]

単位 ページ (4 KB)

割り振られるタイミング DB2 の開始時

解放されるタイミング DB2 の終了時

このパラメーターは、データベースを監査するときに使用されるバッファーのサイズを指定します。

このパラメーターのデフォルト値はゼロ (0) です。 値がゼロ (0) である場合、監査バッファーは使用されません。値がゼロ (0) よりも大きい場合、監査機能によって監査レコードが生成されるときに、そのレコードが置かれる位置にスペースが割り振られます。 4 KB ページの倍数の値が監査バッファーに割り振られたスペースの量です。監査バッファーは動的には割り振られません。このパラメーターに新しい値を指定してそれを有効にするには、DB2 をいったん終了してから再始動する必要があります。

このパラメーターをデフォルト値からゼロ (0) よりも大きい値に変更すると、監査機能は、監査レコードを生成するステートメントの実行とは非同期にレコードをディスクに書き込みます。これは、パラメーター値をゼロ (0) のままにしておくよりも DB2 のパフォーマンスを向上させます。値をゼロ (0) にすると、監査機能は、監査レコードを生成するステートメントの実行と同期で (同時に) レコードをディスクに書き込みます。監査時の同期操作は、DB2 で実行しているアプリケーションのパフォーマンスを低下させます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

dir_cache - ディレクトリー・キャッシュ・サポート

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント

- 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 構成可能

デフォルト[範囲] Yes [Yes; No]

割り振られるタイミング

- アプリケーションが最初の接続を発行したとき、アプリケーション・キャッシュが割り振られます。
- データベース・マネージャー・インスタンスが開始されたとき (db2start)、サーバー・ディレクトリー・キャッシュが割り振られます。

解放されるタイミング

- アプリケーションが処理を終了したとき、アプリケーション・キャッシュが解放されます。
- データベース・マネージャー・インスタンスが停止したとき (db2stop)、サーバー・ディレクトリー・キャッシュが解放されます。

dir_cache を Yes に設定すると、データベース、ノード、および DCS ディレクトリー・ファイルがメモリーにキャッシュされます。 ディレクトリー・キャッシュを使用すると、ディレクトリー・ファイル入出力がなくなり、ディレクトリー情報を得るためのディレクトリー検索が最小化されるため、接続のコストが低減します。ディレクトリー・キャッシュには次の 2 つのタイプがあります。

- 各アプリケーションについて、アプリケーションを実行中のマシン上に割り振られ使用される、アプリケーション・ディレクトリー・キャッシュ
- 内部データベース・マネージャー・プロセスのいくつかについて割り振られ使用される、サーバー・ディレクトリー・キャッシュ

アプリケーション・ディレクトリー・キャッシュでは、アプリケーションがその最初の接続を発行したとき、各ディレクトリー・ファイルが読み取られ、情報がこのアプリケーションのために専用メモリーにキャッシュされます。 キャッシュは、後続の接続要求でアプリケーション処理によって使用され、アプリケーション処理が終了するまで保持されます。データベースがアプリケーション・ディレクトリー・キャッシュに見つからない場合、ディレクトリー・ファイルが検索されますが、キャッシュは更新されません。 アプリケーションがディレクトリー項目を変更した場合、そのアプリケーションで次の接続が発行されると、このアプリケーションのキャッシュはリフレッシュされます。 他のアプリケーションのアプリケーション・ディレクトリー・キャッシュはリフレッシュされません。 アプリケーション処理が終了すると、キャッシュは解放されます。 (コマンド行プロセッサー・セッションが使用していたディレクトリー・キャッシュをリフレッシュするには、 db2 terminate コマンドを発行してください。)

サーバー・ディレクトリー・キャッシュでは、データベース・マネージャー・インスタンスが開始されると (db2start)、それぞれのディレクトリー・ファイルが読み

取られ、情報が共用メモリーにキャッシュされます。このキャッシュは、インスタンスが停止するまで、(db2stop) 保持されます。ディレクトリー項目がこのキャッシュに見つからない場合、ディレクトリー・ファイルが検索されて情報が検索されます。このサーバー・ディレクトリー・キャッシュは、インスタンスの実行中はリフレッシュされません。

推奨: ディレクトリー・ファイルの変更が頻繁ではなく、パフォーマンスが重要な場合は、ディレクトリー・キャッシュを使用してください。

また、リモート・クライアントでは、ディレクトリー・キャッシュは、アプリケーションがいくつかの異なる接続要求を発行する場合に役立ちます。 この場合、キャッシュを行うと、1 つのアプリケーションがディレクトリー・ファイルを読む必要回数が減少します。

ディレクトリー・キャッシュは、データベース・システム・モニター・スナップショットの取得のパフォーマンスも向上させます。さらに、データベースの別名を使用する代わりに、スナップショット呼び出しでデータベース名を明示的に参照することもできます。

注: ディレクトリー・キャッシュがオンで、データベース・マネージャーの開始後に、データベースがカタログ、アンカタログ、作成、またはドロップされる場合、スナップショット呼び出しを実行したときに、エラーが起きる場合があります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

instance_memory - インスタンス・メモリー

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] Automatic [8 — 524 288]

単位 ページ (4 KB)

割り振られるタイミング インスタンスが開始されたとき

解放されるタイミング インスタンスが停止されたとき

このパラメーターは、インスタンス管理用として予約する必要のあるメモリーの量を指定します。これには、インスタンス上でデータベースを記述するメモリー領域も含まれます。

このパラメーターを Automatic に設定した場合は、 DB2 が現行構成で必要なインスタンス・メモリーの量を計算します。 DB2 はオーバーフロー・バッファー用にいくらか追加メモリーを割り当てることもします。ヒープが構成サイズを超える場合は常に、インスタンス共用メモリー領域内のヒープのピーク・メモリー要件を満たすためにオーバーフロー・バッファーが使用されます。動的構成更新などその他の操作も、このオーバーフロー・バッファーにアクセスできます。 **db2pd** コマンドに -memsets オプションを指定して使用し、オーバーフロー・バッファーに残されている未使用のメモリーの量をモニターできます。

関連資料:

- 441 ページの『maxagents - エージェントの最大数』
- 537 ページの『numdb - ホストおよび iSeries データベースを含めた並行アクティブ・データベースの最大数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『db2pd - Monitor and Troubleshoot DB2 コマンド』

java_heap_sz - Java インタープリター最大ヒープ・サイズ

構成タイプ
データベース・マネージャー

適用
- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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
構成可能

デフォルト[範囲]
512 [0 - 524 288]

単位
ページ (4 KB)

割り振られるタイミング
Java ストアード・プロシージャーまたは UDF が始動するとき

解放されるタイミング
db2fmp プロセス (fenced) または db2agent プロセス (トラステッド) が終了するとき

このパラメーターでは、Java DB2 ストアード・プロシージャーおよび UDF にサービスするために開始された Java インタープリターによって使用される、ヒープの最大サイズを決定します。

ヒープは、それぞれの DB2 プロセスごとに 1 つずつ (UNIX ベースのプラットフォームの場合は、それぞれのエージェントまたはサブエージェントごとに 1 つずつ、およびそれ以外のプラットフォームの場合は、それぞれのインスタンスごとに 1 つずつ) あります。ヒープは、それぞれの fenced UDF および fenced ストアード・プロシージャー・プロセスごとに 1 つずつあります。ヒープは、トラステッド・ルーチンのエージェント (サブエージェントは含まない) ごとに 1 つずつあります。ヒープは、Java ストアード・プロシージャーを実行する db2fmp プロセスごとに 1 つずつあります。マルチスレッド db2fmp プロセスの場合は、スレッド・セーフ fenced ルーチンを使用する複数のアプリケーションが単一のヒープからサービスを受けます。いずれの状態においても、このメモリーを割り振ることがあるのは、Java UDF またはストアード・プロシージャーを実行するエージェントまたはプロセスだけです。パーティション・データベース・システムでは、それぞれのパーティションで同じ値が使用されます。

関連資料:

- 536 ページの『jdk_path - Software Developer's Kit for Java インストール・パス』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

mon_heap_sz - データベース・システム・モニター・ヒープ・サイズ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲]

UNIX 90 [0 - 60 000]

ローカル・クライアントおよびリモート・クライアントを持つ Windows データベース・サーバー
66 [0 - 60 000]

ローカル・クライアントを持つ Windows データベース・サーバー
46 [0 - 60 000]

単位 ページ (4 KB)

割り振られるタイミング データベース・マネージャーが *db2start* コマンドで開始されるとき

解放されるタイミング データベース・マネージャーが *db2stop* コマンドで停止されたとき

このパラメーターは、データベース・システム・モニター・データに割り振られるメモリーの大きさ (ページ数) を決定します。メモリーは、スナップショットの取得、モニター・スイッチのオン、モニターのリセット、あるいはイベント・モニターの活動化といった、データベースのモニター活動を実行するときにモニター・ヒープから割り振られます。

ゼロの値を指定すると、データベース・マネージャーはデータベース・システム・モニター・データを収集しません。

推奨: 活動のモニターに必要なメモリーの量は、スイッチが設定されるモニター・アプリケーション (スナップショットまたはイベント・モニターを取るアプリケーション) の数と、データベース活動のレベルに依存します。

このヒープ内の使用可能なメモリーが使い尽くされ、オーバーフロー・バッファーに未使用のメモリーがない場合、以下のいずれかが行われます。

- 最初のアプリケーションが、このイベント・モニターが定義されているデータベースに接続すると、エラー・メッセージが管理通知ログに書き込まれます。
- SET EVENT MONITOR ステートメントを使用して動的に開始されているイベント・モニターが失敗した場合は、エラー・コードがアプリケーションに戻されます。
- モニター・コマンドまたは API サブルーチンが失敗した場合は、エラー・コードがアプリケーションに戻されます。

関連概念:

- 「システム・モニター *ガイドおよびリファレンス*」の『データベース・システム・モニターのメモリー所要量』

関連資料:

- 531 ページの『dft_monswitches - デフォルトのデータベース・システム・モニター・スイッチ』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

ロック

次のパラメーターは、ユーザーの環境でのロック管理を制御します。

- 『dlchktime - デッドロック・チェック・インターバル』
- 425 ページの『locktimeout - ロック・タイムアウト』
- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』

390 ページの『locklist - ロック・リスト用最大ストレージ』も参照してください。

dlchktime - デッドロック・チェック・インターバル

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
伝搬クラス	即時
デフォルト[範囲]	10 000 (10 秒) [1 000 - 600 000]
単位	ミリ秒

デッドロックが発生するのは、同じデータベースに接続されている複数のアプリケーションが 1 つのリソースを際限なく待っているときです。この待機が解決されることはありません。それぞれのアプリケーションが、他のアプリケーションで継続する必要のあるリソースを保留しているからです。

デッドロック・チェック・インターバルでは、1 つのデータベースに接続されているすべてのアプリケーションの間にデッドロックがないか、データベース・マネージャーがチェックする頻度を定義します。

注:

1. パーティション・データベース環境では、このパラメーターが適用されるのは、カタログ・ノードの場合だけです。
2. パーティション・データベース環境では、2 回目の反復の後でないと、デッドロックにフラグが立てられることはありません。

推奨: このパラメーターの値を大きくすると、デッドロックをチェックする頻度が低くなり、したがって、アプリケーション・プログラムでデッドロックの解決を待つ必要のある時間が長くなります。

このパラメーター値を減らすとデッドロック・チェックの頻度が増え、その結果デッドロックが解決されるのをアプリケーション・プログラムが待たなければならない時間が減りますが、データベース・マネージャーがデッドロックをチェックするための時間が増えます。デッドロック・チェック・インターバルが小さすぎると、データベース・マネージャーは頻繁にデッドロック検出を実行するため、ランタイム・パフォーマンスが落ちる可能性があります。並行性を高めるために、このパラメーターが低目に設定されている場合は、*maxlocks* と *locklist* を適切に設定することにより、ロック競合の増加を招き、その結果として、デッドロック状態が増えることになりかねない、不必要なロック・エスカレーションを避けることができます。

関連資料:

- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

locktimeout - ロック・タイムアウト

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	-1 [-1; 0 - 32 767]
単位	秒

このパラメーターは、アプリケーションがロックを獲得するために待機する秒数を指定します。このパラメーターは、アプリケーションのグローバル・デッドロックを避けるために役立ちます。

このパラメーターを 0 に設定した場合、ロックは待機されません。この状態では、要求時にロックが使用可能でない場合は、アプリケーションは即時に -911 を受け取ります。

このパラメーターを -1 に設定する場合、ロック・タイムアウト検出はオフにされます。この状態では、次のいずれかになるまで、ロックを待ちます (ただし、要求時にロックが使用可能でない場合)。

- ロックが GRANT される
- デッドロックが発生する

推奨: トランザクション処理 (OLTP) 環境では、30 秒が初期値として適当です。照会のみの環境では、より高い値で開始できます。両方の場合とも、ベンチマーク技法を使用して、このパラメーターを調整してください。

Data Links Manager を使用して作業するときに、 Data Links Manager (dlfm) インスタンスの管理通知ログ内にロック・タイムアウトが存在する場合は、 *locktimeout* の値を増やす必要があります。 *locklist* の値を大きくすることも考慮する必要があります。

ユーザーがワークステーションを離れたためにトランザクションが停止した場合のように、異常な状況が原因で発生している待ち状態をすみやかに検出できるような値を設定してください。ワークロードがピークであるときに多数の待機状態のロックがあっても、有効なロック要求がタイムアウトにならないような、十分に高い値を設定すべきです。

データベース・システム・モニターを使用すると、アプリケーション (接続) がロック・タイムアウトを経験した回数、またはデータベースがすべての接続しているアプリケーションについて、タイムアウト状態を検出した回数を追跡することができます。

次のような場合は、*lock_timeout* (ロック・タイムアウトの数) モニター・エレメントの値が高くなる原因になり得ます。

- この構成パラメーターの値が低すぎます。
- ロックを保留している時間が伸びているアプリケーション (トランザクション)。データベース・システム・モニターを使用すると、これらのアプリケーションの詳細を調べることができます。
- ロック・エスカレーション (行レベル・ロックから表レベル・ロックへの) によって生じる可能性のある並行性の問題。

関連資料:

- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
- 「システム・モニター *ガイドおよびリファレンス*」の『lock_timeouts ロック・タイムアウト数 : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

maxlocks - エスカレーション前のロック・リストの最大パーセント

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
伝搬クラス	即時
デフォルト[範囲]	
UNIX	10 [1 - 100]
Windows	22 [1 - 100]
単位	パーセント

ロック・エスカレーションとは、行ロックを表ロックで置き換えて、リスト内のロック数を減らす処理のことです。このパラメーターは、アプリケーションが保持するロック・リストのパーセンテージであり、これに達するとデータベース・マネージャーがエスカレーションを実行します。ある 1 つのアプリケーションによって保持されているロックの数が、合計ロック・リスト・サイズに対してこのパーセントに達すると、そのアプリケーションによって保持されているロックに関してロック・エスカレーションが行われます。ロック・リストがスペースを使い尽くした場合も、ロック・エスカレーションが発生します。

データベース・マネージャーは、アプリケーションのロック・リストを調べ、行ロック数が最も多い表を検索して、ロック・エスカレーションの対象となるロックを判別します。行ロックを単一の表ロックで置き換えた後、*maxlocks* 値を超えることがなくなっていれば、ロック・エスカレーションは停止します。そうならない場合は、保持されているロック・リストのパーセンテージが *maxlocks* の値より低くなるまで、ロック・エスカレーションは続きます。*maxlocks* パラメーターに *maxappls* パラメーターを掛けた値が 100 より小であってはなりません。

推奨: 次の公式を使用すると、アプリケーションが平均ロック数の 2 倍のロックを保留できるように、*maxlocks* を設定できます。

```
maxlocks = 2 * 100 / maxappls
```

ただし、平均の 2 倍を達成するために 2 が使用され、100 は許容最大パーセント値を表します。同時に実行するアプリケーションの数が少ない場合は、上記の公式に代えて次の公式を使用できます。

```
maxlocks = 2 * 100 / (average number of applications running
concurrently)
```

maxlocks の設定時に考慮する必要のある事項の 1 つに、ロック・リスト (*locklist*) のサイズとの併用があります。ロック・エスカレーションが行われる前にアプリケーションが保持するロックの数の実際の限度は、次のとおりです。

```
maxlocks * locklist * 4 096 / (100 * 40) (32 ビット・システムの場合)

maxlocks * locklist * 4 096 / (100 * 64) (64 ビット・システムの場合)
```

ただし、4 096 は 1 ページのバイト数、100 は *maxlocks* に許容される最大のパーセント値、40 は 32 ビット・システムの場合の 1 つのロック当たりのバイト数、64 は 64 ビット・システムの場合の 1 つのロック当たりのバイト数です。アプリケーションの 1 つで 1 000 ロックが必要であることが分かっていて、しかもロック・エスカレーションが行われないようにしたい場合は、この公式の *maxlocks* および *locklist* の値を選択すれば、結果は 1 000 より大になります (*maxlocks* に 10 を、*locklist* に 100 を使用すると、この公式によって、必要な 1 000 ロックより大という結果が得られます)。

maxlocks の設定が低過ぎると、他の同時アプリケーションにまだ十分のロック・スペースがあるときに、ロック・エスカレーションが行われます。*maxlocks* の設定が高過ぎると、一部のわずかなアプリケーションだけでロック・スペースのほとんどを使用してしまう可能性があり、それ以外のアプリケーションではロック・エスカレーションを実行する必要が生じます。この場合にロック・エスカレーションが必要になるということは、並行性が低下する結果になります。

データベース・システム・モニターを使用すると、この構成パラメーターの追跡および調整ができます。

関連資料:

- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

入出力およびストレージ

次のパラメーターは、データベースの操作に伴う入出力およびストレージのコストを制御します。

- 『chngpgs_thresh - 変更済みページしきい値』
- 429 ページの『dft_extent_sz - 表スペースのデフォルトのエクステント・サイズ』
- 430 ページの『dft_prefetch_sz - デフォルト・プリフェッチ・サイズ』
- 431 ページの『estore_seg_sz - 拡張ストレージ・メモリー・セグメント・サイズ』
- 432 ページの『num_estore_segs - 拡張ストレージ・メモリー・セグメント数』
- 432 ページの『num_iocleaners - 非同期ページ・クリーナーの数』
- 434 ページの『num_ioservers - I/O サーバーの数』
- 435 ページの『numsegs - SMS コンテナーのデフォルト数』
- 435 ページの『seqdetect - 順次検出フラグ』

chngpgs_thresh - 変更済みページしきい値

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	60 [5 - 99]
単位	パーセント

非同期ページ・クリーナーは、バッファー・プール内のスペースがデータベース・エージェントによって必要とされる前に、変更済みページをバッファー・プール (複数の場合もある) からディスクに書き込みます。その結果として、データベース・エージェントは、変更済みページが書き出されて、バッファー・プール内のスペースを使用できるようになるのを待機する必要がなくなります。こうして、データベース・アプリケーション全体のパフォーマンスが向上します。

このパラメーターを使用すると、非同期ページ・クリーナーが現在アクティブでない場合に、非同期ページ・クリーナーが始動する変更済みページ数のレベル (パーセント) を指定できます。ページ・クリーナーを始動すると、ディスクに書き込むページのリストが作成されます。これらのページのディスクへの書き込みが完了すると、ページ・クリーナーは再度非アクティブになり、次のトリガーの開始を待ちます。

読み取り専用 (たとえば、照会) 環境では、このようなページ・クリーナーは使用されません。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が設定されると (つまり、ページ・クリーニングの代替方法を使用すると)、*chngpgs_thresh* パラメーターは無効になり、 DB2 はバッファー・プールに保持するダーティー・ページの数を自動的に判別します。

推奨: 更新トランザクション・ワークロードが大きいデータベースの場合は、パラメーター値をデフォルト値以下に設定することによって、一般的には、バッファー・プール内に十分なクリーン・ページを確保できます。データベースに非常に大きな表が少数しかない場合は、パーセンテージをデフォルトよりも大きくすると、パフォーマンスを上げることができます。

関連資料:

- 432 ページの『num_iocleaners - 非同期ページ・クリーナーの数』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dft_extent_sz - 表スペースのデフォルトのエクステント・サイズ

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
伝搬クラス	即時
デフォルト[範囲]	32 [2 - 256]
単位	ページ

表スペースの作成時に、オプションで `EXTENTSIZE n` を指定できます。 `n` はエクステント・サイズです。 CREATE TABLESPACE ステートメントでエクステント・サイズを指定しなかった場合、データベース・マネージャーはこのパラメーターで与えられた値を使用します。

推奨 : 多くの場合、表スペースの作成時に、エクステント・サイズを明示的に指定します。このパラメーターの値を選択する前に、CREATE TABLESPACE ステートメントのエクステント・サイズを明示的に指定する方法を理解しておくべきです。

関連概念:

- 「*管理ガイド: プランニング*」の『エクステント・サイズ』

関連資料:

- 430 ページの『dft_prefetch_sz - デフォルト・プリフェッチ・サイズ』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dft_prefetch_sz - デフォルト・プリフェッチ・サイズ

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

伝搬クラス 即時

デフォルト[範囲]

UNIX	AUTOMATIC [0 — 32 767]
Windows	AUTOMATIC [0 — 32 767]

単位 ページ

表スペースの作成時に、オプションで PREFETCHSIZE *n* を指定できます (*n* は、プリフェッチが行われる場合に、データベース・マネージャーが読み取るページ数です)。 CREATE TABLESPACE ステートメントの呼び出し時にエクステント・サイズを指定しなかった場合、データベース・マネージャーは *dft_prefetch_sz* パラメーターの現行値を使用します。

表スペースが AUTOMATIC DFT_PREFETCH_SZ を使用して作成される場合、表スペースのプリフェッチ・サイズは AUTOMATIC になります。 AUTOMATIC は、DB2 が次の式を使用して表スペースのプリフェッチ・サイズを自動的に計算および更新することを意味します。

```
prefetch size = (# containers) * (# physical spindles) * extent size
```

phisical spindle の数はデフォルト設定が 1 であり、 DB2 レジストリー変数 DB2_PARALLEL_IO で指定できます。この計算は以下のタイミングで実行されます。

- データベース始動時
- AUTOMATIC プリフェッチ・サイズの指定で、初めて表スペースが作成されるとき
- ALTER TABLESPACE ステートメントの実行によって、表スペースのコンテナーの数が変更されるとき
- ALTER TABLESPACE ステートメントの実行によって、表スペースのプリフェッチ・サイズが AUTOMATIC に更新されるとき

ALTER TABLESPACE ステートメントを呼び出してプリフェッチ・サイズを手動で更新すると、直ちにプリフェッチ・サイズの AUTOMATIC 状態はオンまたはオフに切り替わります。

推奨: システム・モニター・ツールを使用して、システムが入出力を待機しているときに CPU がアイドルになっているかどうかを判別できます。このパラメーターの値を増やすと、使用される表スペースのプリフェッチ・サイズが定義されていない場合に役立ちます。

このパラメーターは、データベース全体にデフォルト値を適用しますが、データベース内のすべての表スペースに適しているとは限りません。たとえば、値 32 はエクステント・サイズが 32 ページの表スペースには適しているかもしれませんが、

エクステント・サイズが 25 ページの表スペースには適していません。一番よい方法は、それぞれの表スペースにプリフェッチ・サイズを明示的に設定することです。

デフォルトの表スペース・エクステント・サイズ (*dft_extent_sz*) で定義された表スペースの入出力を最小化するには、このパラメーターを *dft_extent_sz* パラメーターの値の因数または倍数として指定してください。たとえば、*dft_extent_sz* パラメーターが 32 の場合は、*dft_prefetch_sz* を 16 (32 の因数) または 64 (32 の倍数) に設定します。プリフェッチ・サイズがエクステント・サイズの倍数である場合は、以下の条件が満たされていればデータベース・マネージャーは入出力を並列して行うことができます。

- プリフェッチ中のエクステントが異なる物理装置上にある
- 複数の入出力サーバーが構成されている (*num_ioservers*)

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 575 ページの『システム環境変数』

estore_seg_sz - 拡張ストレージ・メモリー・セグメント・サイズ

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	16 000 [0 - 1 048 575]
単位	ページ (4 KB)

このパラメーターは、データベース中の個々の拡張メモリー・セグメントのページ数を指定します。このパラメーターが使用されるのは、仮想アドレス可能メモリーの最大量よりも大きい実アドレス可能メモリーが、マシンにある場合だけです。

推奨: このパラメーターは、拡張ストレージが使用可能な場合だけ有効で、*num_estore_segs* パラメーターの指定に従って使用されます。個々の拡張メモリー・セグメントで使用されるページ数を指定する場合、 *num_estore_segs* パラメーターを見直して修正することにより、拡張メモリー・セグメントの数を考慮することも必要です。

関連資料:

- 432 ページの『num_estore_segs - 拡張ストレージ・メモリー・セグメント数』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

num_estore_segs - 拡張ストレージ・メモリー・セグメント数

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲] 0 [0 - 2 147 483 647]

このパラメーターは、データベースで使用する場合に使用可能な拡張ストレージ・メモリー・セグメントの数を指定します。

デフォルトでは、拡張ストレージ・メモリー・セグメントなしです。

推奨: このパラメーターを使用して、拡張ストレージ・メモリー・セグメントの使用を設定するのは、ご使用のプラットフォーム環境に最大アドレス・スペースよりも多くのメモリーがあり、しかもこのメモリーを使用する場合だけにしてください。セグメント数を指定する場合は、*estore_seg_sz* パラメーターの見直しと変更を行うことによって、セグメントのそれぞれのサイズを考慮することも必要です。

num_estore_segs と *estore_seg_sz* の両方の構成パラメーターが設定されているときは、CREATE/ALTER BUFFERPOOL ステートメントによって拡張メモリーを使用するバッファー・プールを指定する必要があります。

関連資料:

- 431 ページの『estore_seg_sz - 拡張ストレージ・メモリー・セグメント・サイズ』
- 「*SQL リファレンス 第 2 巻*」の『ALTER BUFFERPOOL ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE BUFFERPOOL ステートメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

num_iocleaners - 非同期ページ・クリーナーの数

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲] 1 [0 - 255]

単位 カウンター

このパラメーターを使用すると、データベースの非同期ページ・クリーナーの数を指定できます。これらのページ・クリーナーは、バッファー・プール内のスペースがデータベース・エージェントによって必要とされる前に、変更済みページをバッ

ファー・プールからディスクに書き込みます。その結果として、データベース・エージェントは、変更済みページが書き出されて、バッファー・プール内のスペースを使用できるようになるのを待機する必要がなくなります。こうして、データベース・アプリケーション全体のパフォーマンスが向上します。

このパラメーターをゼロ (0) に設定した場合は、開始されるページ・クリーナーがないので、その結果として、バッファー・プールからディスクへのページ書き込みのすべてを、データベース・エージェントが実行することになります。データベースが多くの物理ストレージ・デバイスにまたがって保管されている場合は、物理ストレージ・デバイスの 1 つがアイドル状態になる可能性が高いので、このパラメーターがデータベースに重大なパフォーマンス上の影響を及ぼす可能性があります。ページ・クリーナーが構成されていない場合は、周期的にログがいっぱいになる状態がアプリケーションで検出される可能性があります。

データベースを使用するアプリケーションが、主として、データを更新するトランザクションで構成されている場合は、クリーナーの数を増やすと、パフォーマンスが向上します。ページ・クリーナーの数を増やすと、どの時点においても、ディスク上のデータベースの内容の最新性が増すため、停電などのソフト障害からのリカバリー時間も短縮されます。

推奨: このパラメーターの値を設定する際は、次の要因を考慮してください。

- アプリケーションのタイプ
  - 更新が行われない照会専用データベースの場合は、このパラメーターはゼロ (0) になるように設定します。例外は、照会ワークロードによって TEMP 表が多数作成される結果になる (これについては、EXPLAIN ユーティリティーを使用して判別できます) 場合です。
  - データベースに対してトランザクションが実行される場合は、このパラメーターは、1 とデータベース用として使用されている物理ストレージ・デバイスの台数の間になるように設定します。
- ワークロード

  更新トランザクションの率が高い環境では、ページ・クリーナーの数を増やして構成する必要があります。
- バッファー・プール・サイズ

  バッファー・プールの容量が大きい環境でも、ページ・クリーナーの数を増やして構成する必要があります。

データベース・システム・モニターを使用すると、バッファー・プールからの書き込みアクティビティーについてのイベント・モニターから得られる情報を使用して、この構成パラメーターを調整する場合に役立ちます。

- 次の 2 つの条件が真の場合は、パラメーターの値を小さくすることができます。
  - *pool_data_writes* がおよそ *pool_async_data_writes* に等しい。
  - *pool_index_writes* がおよそ *pool_async_index_writes* に等しい。
- 次の 2 つの条件のいずれかが真の場合は、パラメーターの値を大きくする必要があります。
  - *pool_data_writes* が *pool_async_data_writes* よりもはるかに大きい。

– *pool_index_writes* が *pool_async_index_writes* よりもはるかに大きい。

関連資料:

- 428 ページの『chngpgs_thresh - 変更済みページしきい値』
- 「システム・モニター *ガイドおよびリファレンス*」の『pool_data_writes バッファー・プールへのデータの書き込み : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『pool_index_writes バッファー・プール索引の書き込み : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『pool_async_data_writes バッファー・プール非同期データ書き込み : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『pool_async_index_writes バッファー・プール非同期索引書き込み : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

num_ioservers - I/O サーバーの数

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	3 [1 - 255]
単位	カウンター
割り振られるタイミング	アプリケーションがデータベースに接続するとき
解放されるタイミング	アプリケーションがデータベースから切断されたとき

入出力サーバーは、データベース・エージェントの代わりに、バックアップおよびリストアなどのユーティリティーによるプリフェッチ入出力および非同期入出力を実行するために使用されます。このパラメーターは、データベースのための入出力サーバー数を指定します。あるデータベースに対して、この入出力数を超えるプリフェッチおよびユーティリティーを行うことはできません。入出力サーバーは、出された入出力が進行中の間、待ち状態になります。非プリフェッチ入出力は、データベース・エージェントから直接スケジュールされ、その結果 *num_ioservers* によって制限されます。

推奨: システム内のすべての入出力装置を完全に利用するために適正な値は、一般にデータベースが置かれている物理装置数に 1 または 2 を加えたものです。各入出力サーバーにはそれぞれほんの少しのオーバーヘッドが発生するのみで、未使用の入出力サーバーはアイドル状態のままになるため、追加の入出力サーバーを構成することをお薦めします。

関連資料:

- 430 ページの『dft_prefetch_sz - デフォルト・プリフェッチ・サイズ』

- 435 ページの『seqdetect - 順次検出フラグ』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

numsegs - SMS コンテナーのデフォルト数

構成タイプ データベース

パラメーター・タイプ 通知

単位 カウンター

SMS 表スペースにのみ適用されるこのパラメーターは、デフォルトの表スペース内で作成されるコンテナーの数を示します。このパラメーターは、CREATE DATABASE コマンドで明示的に指定されているか否かにかかわらず、データベースの作成時に使用された情報を示します。 CREATE TABLESPACE ステートメントは、このパラメーターを使用**しません。**

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

seqdetect - 順次検出フラグ

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

伝搬クラス 即時

デフォルト[範囲] Yes [Yes; No]

データベース・マネージャーは、入出力をモニターすることができ、順次ページ読み取りが行われている場合は、データベース・マネージャーは入出力プリフェッチを活動化できます。このタイプの順次プリフェッチは、*順次検出* という名前で呼ばれています。 *seqdetect* 構成パラメーターを使用して、データベース・マネージャーが順次検出を実行する必要があるかどうかコントロールできます。

このパラメーターが No に設定されている場合、プリフェッチは、たとえば、表ソート、表スキャン、またはリスト・プリフェッチなど、有用であることがデータベース・マネージャーに分かっている場合にのみ行われます。

推奨: ほとんどの場合、このパラメーターにはデフォルト値を使用する必要があります。順次検出をオフにするのは、それ以外のチューニングによってでは重大な照会パフォーマンスが解決できなかった場合だけにしてください。

関連資料:

- 430 ページの『dft_prefetch_sz - デフォルト・プリフェッチ・サイズ』

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

エージェント

次のパラメーターは、並行して実行できるアプリケーションの数を制御し、最適なパフォーマンスが得られるようにします。

- 『agentpri - エージェントの優先順位』
- 438 ページの『avg_appls - アクティブ・アプリケーションの平均数』
- 438 ページの『max_connections - クライアント接続の最大数』
- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 441 ページの『maxagents - エージェントの最大数』
- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 443 ページの『maxcagents - 同時エージェントの最大数』
- 444 ページの『maxfilop - アプリケーション単位の最大データベース・ファイル・オープン数』
- 445 ページの『maxtotfilop - 最大合計オープン・ファイル数』
- 446 ページの『num_initagents - プール内エージェントの初期数』
- 447 ページの『num_poolagents - エージェント・プール・サイズ』

agentpri - エージェントの優先順位

構成タイプ　データベース・マネージャー

適用

- ローカルとリモート・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　構成可能

デフォルト[範囲]

AIX　-1 [41 - 125]

他の UNIX
-1 [41 - 128]

Windows
-1 [0 - 6]

このパラメーターは、オペレーティング・システム・スケジューラーによって、すべてのエージェントと、データベース・マネージャー・インスタンス・プロセスお

よびスレッドの両方に与えられる優先順位をコントロールします。パーティション・データベース環境では、これにコーディネーター・エージェントとサブエージェントの両方と、並列システム・コントローラー、および FCM デーモンも含まれます。この優先順位は、DB2 プロセス、エージェント、他のプロセスに関連するスレッド、およびマシンで実行中のスレッドに対する CPU 時間の割り当て方法を決定します。パラメーターを -1 に設定すると、特殊なアクションは取られず、データベース・マネージャーは、オペレーティング・システムがすべてのプロセスとスレッドをスケジュールする通常の方法でスケジュールされます。パラメーターを -1 以外の値に設定すると、データベース・マネージャーは、そのパラメーターの値に設定される静的優先順位で、そのプロセスとスレッドを作成します。したがって、このパラメーターを使用すると、データベース・マネージャー・プロセスおよびスレッドをマシンで実行する優先順位をコントロールできます。

このパラメーターを使用して、データベース・マネージャーのスループットを増やすことができます。このパラメーターの設定値は、データベース・マネージャーが稼働しているオペレーティング・システムによって異なります。たとえば、UNIX ベース・システム環境では、数値が低いほど優先順位が高くなります。パラメーターを 41 ～ 125 の間の値に設定すると、データベース・マネージャーはそのパラメーターの値に設定される UNIX の静的優先順位で、そのエージェントを作成します。このことは UNIX ベースの環境では重要です。数値を小さくするとデータベース・マネージャーの優先順位が高くなりますが、他のプロセス (アプリケーションとユーザーの両方) で十分な CPU 時間を獲得できないため遅延が起きることがあるからです。このパラメーターの設定値と、マシン上で予期される他の活動との平衡を取る必要があります。

推奨: 最初は、デフォルト値を使用してください。デフォルト値では、他のユーザー/アプリケーションへの応答時間とデータベース・マネージャー・スループットの間の適切な折衷案を提供します。

データベースのパフォーマンスが問題になる場合は、ベンチマークの手法を使って、このパラメーターの最適な設定値を決定できます。データベース・マネージャーの優先順位を上げるときは、注意する必要があります。特に CPU 使用率が非常に高いときは、他のユーザー・プロセスのパフォーマンスが著しく低下する可能性があるからです。データベース・マネージャー・プロセスおよびスレッドの優先順位を上げると、有効なパフォーマンスの効果を得られる可能性があります。

注: このパラメーターを UNIX ベースのプラットフォーム上のデフォルト以外の値に設定すると、ガバナーを使用してエージェントの優先順位を変更することができなくなります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

avg_appls - アクティブ・アプリケーションの平均数

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
伝搬クラス	ステートメント境界
デフォルト[範囲]	1 [1 - maxappls]
単位	カウンター

このパラメーターを SQL オプティマイザーで使用すると、選択されたアクセス・プラン用としてランタイムに使用可能になる、バッファー・プールの量を見積もるのに役立ちます。

推奨: DB2 をマルチユーザー環境で、特に複雑な照会と大容量バッファー・プールを使用して実行する場合は、SQL オプティマイザーに複数のユーザー照会がシステムを使用していることを知らせて、 SQL オプティマイザーがバッファー・プールがどれだけ使えるかをより少なく見積もるようにする必要があります。

このパラメーターを設定するときは、このデータベースを通常使用する複雑な照会を実行するアプリケーションの数を見積もる必要があります。軽い OLTP アプリケーションはすべて、この見積もりから除外する必要があります。この数の見積もりが難しい場合は、以下の値を掛けて計算できます。

- データベースに対して実行中のアプリケーションの平均数。特定の時点におけるアプリケーションの数についての情報は、データベース・システム・モニターによって得ることができ、サンプリング技法を使用すれば、ある期間についての平均を計算できます。データベース・システム・モニターから得られる情報には、OLTP アプリケーションと非 OLTP アプリケーションの両方が含まれます。
- 複合な照会を実行するアプリケーションのパーセンテージの見積もり

オプティマイザーに影響する他の構成パラメーターの調整の場合と同じように、このパラメーターも増分を小さくして調整する必要があります。こうすることによって、アクセスパスの変化を小さくすることができます。

このパラメーターを変更した場合は、アプリケーションの再バインド (REBIND PACKAGE コマンドを使用) を考慮してください。

関連資料:

- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

max_connections - クライアント接続の最大数

構成タイプ	データベース・マネージャー
パラメーター・タイプ	構成可能

デフォルト[範囲] -1 (*max_coordagents*) [-1; *max_coordagents* — 64 000]

コンセントレーターがオフの場合、このパラメーターは、パーティション当たりの許容クライアント接続最大数を示します。コンセントレーターがオフなのは、*max_connections* が *max_coordagents* に等しいときです。*max_connections* が *max_coordagents* より大きいときは、コンセントレーターはオンです。

このパラメーターは、インスタンスに接続できるアプリケーションの最大数をコントロールします。通常、各アプリケーションにはコーディネーター・プログラム・エージェントが割り当てられます。エージェントは、アプリケーションとデータベースの間の操作を容易にします。このパラメーターにデフォルト値が使用される場合、コンセントレーター機能は活動化されません。結果として、各エージェントは専用メモリーを使用して機能し、データベース・マネージャーと、バッファー・プールなどのデータベース・グローバル・リソースを他のエージェントと共有します。パラメーターがデフォルトよりも大きな値に設定されると、コンセントレーター機能が活動化されます。コンセントレーターの意図は、1 つのクライアント・アプリケーションに対するサーバー・リソースを、DB2 Connect ゲートウェイによって 10 000 よりも多いクライアント接続を扱うことができるまで削減することです。

値が -1 では、限度が *max_coordagents* であることを示します。

以前のバージョンの DB2 では、このプログラムは *max_logicagents* という名前で呼ばれていました。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

max_coordagents - コーディネーター・エージェントの最大数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] -1 (*maxagents* - *num_initagents*)

[-1, 0 - *maxagents*]

intra_parallel が Yes に設定されているパーティション・データベース環境の場合は、デフォルトは *maxagents - num_initagents* です。それ以外の場合は、デフォルトは *maxagents* になります。 これは、非パーティション・データベース環境では、システムがパーティション内並列処理用に構成されていない限り、 *max_coordagents* は常に *maxagents* と同等であることになります。

現在の環境がパーティション・データベース環境ではなく、*intra_parallel* パラメーターが使用可能ではない場合、*max_coordagents* は *maxagents* と同等でなければなりません。

コンセントレーターがオフのとき、つまり、*max_connections* が *max_coordagents* に等しい場合、このパラメーターは、パーティションまたは非パーティション・データベース環境でサーバー上に同時に存在できるコーディネーター・エージェントの最大数を決定します。

データベースに接続する、またはインスタンスに接続するローカルまたはリモートのアプリケーションごとに 1 つのコーディネーター・エージェントが獲得されます。インスタンスへのアタッチを必要とする要求には、CREATE DATABASE、DROP DATABASE、およびデータベース・システム・モニター・コマンドがあります。

コンセントレーターがオンの場合、つまり、*max_connections* が *max_coordagents* より大きいときは、接続の数がそれにサービスするためのコーディネーター・エージェントの数よりも多くなります。アプリケーションがアクティブ状態であるのは、アプリケーションにサービスするコーディネーター・エージェントがある場合だけです。コーディネーター・エージェントがない場合、アプリケーションは非アクティブ状態です。アクティブ・アプリケーションからの要求には、データベース・コーディネーター・エージェント (および SMP または MPP 構成内のサブエージェント) がサービスします。非アクティブ・アプリケーションからの要求はキューに入れられ、そのアプリケーションにサービスするデータベース・コーディネーター・エージェントが割り当てられると、アプリケーションがアクティブになります。したがって、このパラメーターを使用すると、システムに対する負荷をコントロールできます。

関連資料:

- 446 ページの『num_initagents - プール内エージェントの初期数』
- 447 ページの『num_poolagents - エージェント・プール・サイズ』
- 525 ページの『intra_parallel - パーティション内並列処理機能の使用可能化』
- 441 ページの『maxagents - エージェントの最大数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

maxagents - エージェントの最大数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 200 [1 - 64 000]

400 [1 - 64 000] (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・サーバー上)

単位 カウンター

このパラメーターは、コーディネーター・エージェントかサブエージェントかにかかわらず、どの特定の時点においても、アプリケーション要求を受け入れるために使用可能なデータベース・マネージャー・エージェントの最大数を示します。コーディネーター・エージェントの数を制限したい場合は、 *max_coordagents* パラメーターを使用してください。

個々の追加エージェントには付加的なメモリーが必要になるので、このパラメーターはメモリーの制限がある環境でデータベース・マネージャーの合計メモリー使用率を制限するのに役立ちます。

推奨: *maxagents* の値は、少なくとも、同時にアクセスできるそれぞれのデータベース内の *maxappls* の値の合計であることが必要です。データベースの数が *numdb* パラメーターよりも多い場合は、*numdb* と *maxappls* の最大値の積を使用するのが最も安全な方法です。

追加エージェントごとに、データベース・マネージャーの開始時に割り振られる多少のリソース・オーバーヘッドが必要になります。

関連資料:

- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 447 ページの『num_poolagents - エージェント・プール・サイズ』
- 443 ページの『maxcagents - 同時エージェントの最大数』
- 448 ページの『fenced_pool - fenced プロセスの最大数』
- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 405 ページの『min_priv_mem - 最小コミット済み専用メモリー』

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

maxappls - アクティブ・アプリケーションの最大数

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
伝搬クラス	即時
デフォルト[範囲]	Automatic [Automatic; 1 - 60 000]
単位	カウンター

このパラメーターは、データベースに (ローカルとリモートの両方で) 接続できる並行アプリケーションの最大数を指定します。データベースにアタッチしている各アプリケーションが、いくつかの専用メモリーを割り振るので、同時アプリケーションの数を多くすると、多くのメモリーが使用される可能性があります。

maxappls を *Automatic* に設定すると、接続されたアプリケーションを幾つでも任意の数だけ使用できるという効果が生じます。 DB2 では、新しいアプリケーションをサポートするために必要なリソースを動的に割り振ります。

このパラメーターを *Automatic* に設定したくない場合は、このパラメーターの値は、接続されたアプリケーションの数に、これらと同じアプリケーションで 2 フェーズ・コミットおよびロールバックを完了する処理で同時に実行される数を加えた合計に等しいか、それよりも大でなければなりません。次に、どのようなときでも発生する可能性がある未確定トランザクション数をこの合計に追加します。

アプリケーションがデータベースへの接続を試みたときに、*maxappls* の値にすでに達していた場合は、すでに最大数のアプリケーションがデータベースに接続されていることを示すエラーが、アプリケーションに返されます。

Data Links Manager を使用するアプリケーションが多くなる場合は、 *maxappls* の値を増やす必要があります。次の公式を使用して、必要な値を計算します。

```
<maxappls> = 5 * (ノードの数) + (Data Links Manager を使用する
                  アクティブ・アプリケーションのピーク数)
```

Data Links Manager でサポートされている最大値は 2 000 です。

パーティション・データベース環境下では、これはデータベース・パーティションに対して並行してアクティブであるアプリケーションの最大数です。このパラメーターは、サーバーがアプリケーションのコーディネーター・プログラム・ノードであるかどうかにかかわらず、データベース・パーティション・サーバー上のデータベース・パーティションに対するアクティブなアプリケーションの数を制限します。パーティション・データベース環境下のカタログ・ノードでは、この環境下のすべてのアプリケーションがカタログ・ノードへの接続を要求するため、他のタイプの環境の場合よりも高い *maxappls* の値が必要です。

推奨: *maxlocks* パラメーターを低くせずにこのパラメーターの値を増やしたり、*locklist* パラメーターを増やしたりすると、アプリケーション限界ではなくロック (*locklist*) のデータベース限界に達し、その結果ロック・エスカレーションの問題が広がることになります。

ある程度は、アプリケーションの最大数も *maxagents* によって決まります。使用可能なエージェント (*maxagents*) だけでなく、使用可能な接続 (*maxappls*) が存在する場合、アプリケーションはデータベースにしか接続できません。さらに、アプリケーションの最大数は、*max_coordagents* 構成パラメーターによってもコントロールされます。*max_coordagents* に達している場合、新しいアプリケーション (すなわち、コーディネーター・エージェント) を始動できません。

関連タスク:

- 「*管理ガイド: プランニング*」の『未確定トランザクションの手動での解決』

関連資料:

- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 441 ページの『maxagents - エージェントの最大数』
- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 426 ページの『maxlocks - エスカレーション前のロック・リストの最大パーセント』
- 438 ページの『avg_appls - アクティブ・アプリケーションの平均数』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

maxcagents - 同時エージェントの最大数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] -1 (*max_coordagents*) [-1; 1 - *max_coordagents*]

単位 カウンター

データベース・マネージャー・トランザクションを同時に実行中であることが可能なデータベース・マネージャー・エージェントの最大数。このパラメーターは、多くのアプリケーション活動が同時に実行される場合に、システムの負荷をコントロ

ールするために使用されます。たとえば、多数の接続が必要なシステムがあるのに、その接続用のメモリーに制限がある場合などです。このパラメーターの調整は、多くの活動が同時に実行される場合に、極端なオペレーティング・システム・ページングの原因となるような環境で有効です。

このパラメーターは、データベースに接続できるアプリケーションの数を制限しません。これは、データベース・マネージャーが一度に並行して処理できるデータベース・マネージャー・エージェント数のみを制限しますが、これにより、処理のピーク時にシステム・リソースの使用率を制限できます。

値が -1 の場合は、限度が *max_coordagents* であることを示します。

推奨: ほとんどの場合に、このパラメーターのデフォルト値を使用できます。多くのアプリケーションを同時に実行することによって問題が生じている場合は、ベンチマーク・テストを使用してこのパラメーターを調整し、データベースのパフォーマンスを最適化できます。

関連資料:

- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 441 ページの『maxagents - エージェントの最大数』
- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

maxfilop - アプリケーション単位の最大データベース・ファイル・オープン数

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	
伝搬クラス	トランザクション境界	
デフォルト[範囲]		
	UNIX	64 [2 - 1950]
	Windows	64 [2 - 32 768]
単位	カウンター	

このパラメーターは、それぞれのデータベース・エージェントごとにオープンしておけるファイル・ハンドルの最大数を指定します。あるファイルをオープンしたために、この値を超えることになった場合は、使用中の一部のファイルがクローズされます。*maxfilop* が小さ過ぎる場合は、この限度を超えないようにするためにファイルをオープンしたり、クローズしたりするオーバーヘッドが過剰になり、パフォーマンスを低下させる可能性があります。

オペレーティング・システムとデータベース・マネージャーの間では、 SMS 表スペースと DMS 表スペース・ファイル・コンテナーが両方ともファイルとして処理されるので、ファイル・ハンドルが必須です。SMS 表スペースでは、DMS ファイル表スペースの場合に使用されるコンテナーの数に比べて、一般的には、多くのファイルが使用されます。したがって、SMS 表スペースを使用している場合は、DMS ファイル表スペースの場合に必要とする値に比べて大きな値が、このパラメーターに必要になります。

また、このパラメーターを使用すると、エージェント単位のファイル・ハンドル数を制限することによって、データベース・マネージャーで使用されるファイル・ハンドルの総計がオペレーティング・システム限度を超えることのないようにすることもできます。なお、実際の数は、同時に稼働するエージェントの数に応じて異なります。

関連資料:

- 442 ページの『maxappls - アクティブ・アプリケーションの最大数』
- 445 ページの『maxtotfilop - 最大合計オープン・ファイル数』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

maxtotfilop - 最大合計オープン・ファイル数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 16 000 [100 - 32 768]

単位 カウンター

このパラメーターは、単一のデータベース・マネージャー・インスタンスで実行中の、すべてのエージェントと他のスレッドがオープンできる、ファイルの最大数を定義します。ファイルをオープンすることによってこの値を超えた場合、エラーがアプリケーションに返されます。

注: このパラメーターは、UNIX ベースのプラットフォームには適用されません。

推奨: このパラメーターを設定する場合、データベース・マネージャー・インスタンス内の各データベースで使用できるファイル・ハンドルの数を考慮してください。このパラメーターの上限の見積もりは、次のようにして行います。

1. インスタンス内のそれぞれのデータベースごとにオープンできるファイル・ハンドルの最大数を、次の公式を使用して計算します。

   ```
   maxappls * maxfilop
   ```

2. 上の結果を合計し、その値がパラメーターを超えていないかを調べます。

新しいデータベースを作成する場合は、このパラメーターの値をもう一度見積もり直してください。

関連資料:

- 444 ページの『maxfilop - アプリケーション単位の最大データベース・ファイル・オープン数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

num_initagents - プール内エージェントの初期数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 0 [0 — *num_poolagents*]

このパラメーターは、DB2START 時にエージェント・プールで作成されるアイドル状態のエージェントの初期数を決定します。

関連資料:

- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 447 ページの『num_poolagents - エージェント・プール・サイズ』
- 441 ページの『maxagents - エージェントの最大数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

num_poolagents - エージェント・プール・サイズ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] -1 (*maxagents* / 2) [-1; 0 — *maxagents*]

コンセントレーターがオフの場合、つまり *max_connections* が *max_coordagents* に等しいときは、このパラメーターは、アイドル・エージェント・プールの最大サイズを決定します。アイドル・エージェントは、並列サブエージェントまたはコーディネーター・エージェントとして使用できます。このパラメーター値で示されている数より多くのエージェントが作成されると、それらのエージェントは現行の要求の実行が終了した時点でプールに戻されず終了します。

コンセントレーターがオンの場合、つまり *max_connections* が *max_coordagents* より大きいときは、このパラメーターの値にかかわりなく、エージェントは常にプールに戻されます。システム負荷およびプール内でエージェントがアイドル状態のままでいる時間に基づいて、エージェントは、アイドル・プールのサイズを構成されたパラメーター値まで小さくするため、必要に応じて自分自身を終了させます。

コンセントレーターがオンの場合を除き、このパラメーターの値が 0 の場合、エージェントは必要に応じて作成され、自分の現在の要求の実行が終わったら終了します。

推奨: 同時に接続しているアプリケーションの数が少ない意思決定支援環境を実行している場合、*num_poolagents* を小さい値に設定して、エージェント・プールがアイドル・エージェントでいっぱいにならないようにしてください。

多数のアプリケーションが同時に接続しているトランザクション処理環境を実行している場合、*num_poolagents* の値を大きくして、エージェントが頻繁に作成されて終了されることに CPU を使わないようにしてください。

関連資料:

- 446 ページの『num_initagents - プール内エージェントの初期数』
- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 525 ページの『max_querydegree - 照会の最大並列処理多重度』
- 441 ページの『maxagents - エージェントの最大数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

ストアード・プロシージャーおよびユーザー定義関数

次のパラメーターは、 fenced ストアード・プロシージャーおよびユーザー定義関数のパフォーマンスを制御します。

- 『fenced_pool - fenced プロセスの最大数』
- 449 ページの『keepfenced - fenced プロセスの保持』
- 450 ページの『num_initfenced - fenced プロセスの初期数』

fenced_pool - fenced プロセスの最大数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] -1 (*max_coordagents*)

単位 カウンター

スレッド化 db2fmp プロセス (スレッド・セーフ・ストアード・プロシージャーおよび UDF を実行するプロセス) の場合、このパラメーターは、それぞれの db2fmp プロセス内でキャッシュに入れられるスレッドの数を表します。非スレッド化 db2fmp プロセスの場合は、このパラメーターではキャッシュに入れられるプロセスの数を表します。

推奨: fenced ストアード・プロシージャーまたはユーザー定義関数を使用している環境の場合は、インスタンス上で実行する最大数の同時ストアード・プロシージャーおよび UDF をプロセスするために、適切な数の db2fmp プロセスが確実に使用可能であるようにして、ストアード・プロシージャーおよび UDF の実行の一環として新しい fenced モード・プロセスが作成される必要がないようにするには、このパラメーターが使用できます。

パラメーターが -1 に設定されている場合は、キャッシュに入れられる db2fmp プロセスの最大数は、*max_coordagents* パラメーターに設定されている値と同じになります。

デフォルト値では、不適切な量のシステム・リソースが db2fmp プロセスに付与され、データベース・マネージャーのパフォーマンスに影響を生じるため、ご使用の環境ではデフォルト値が適切でない場合は、下記を使用すれば、このパラメーターを調整するための手掛かりが得られます。

```
fenced_pool = # of applications allowed to make stored procedure and
UDF calls at one time
```

keepfenced が *yes* に設定されている場合は、fenced ルーチン呼び出しが処理され、エージェントに戻された後も、キャッシュ・プール内に作成されたそれぞれの db2fmp プロセスは存在し、システム・リソースを使用し続けます。

keepfenced が *no* に設定されている場合は、非スレッド化 db2fmp プロセスは、実行を完了した時点で終了するので、キャッシュ・プールはありません。マルチスレッド化 db2fmp プロセスは存在し続けますが、これらのプロセスにプールされるスレッドはありません。つまり、*keepfenced* が *no* に設定されている場合でも、スレッド化 C db2fmp プロセスを 1 つと、スレッド化 Java db2fmp プロセスを 1 つ、システム上にもつことができることを示します。

以前のバージョンの DB2 では、このパラメーターは *maxdari* という名前で呼ばれていました。

関連資料:

- 439 ページの『max_coordagents - コーディネーター・エージェントの最大数』
- 441 ページの『maxagents - エージェントの最大数』
- 449 ページの『keepfenced - fenced プロセスの保持』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 450 ページの『num_initfenced - fenced プロセスの初期数』

keepfenced - fenced プロセスの保持

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] Yes [Yes; No]

このパラメーターは fenced モード・ルーチン呼び出しの完了後に、fenced モード・プロセスが保持されるかどうかを示します。 fenced モード・プロセスは、ユーザー作成 fenced モード・コードをデータベース・マネージャー・エージェント・プロセスから分離するために、独立したシステム・エンティティーとして作成されます。このパラメーターは、データベース・サーバーにのみ適用できます。

keepfenced が *no* に設定されていて、実行されているルーチンがスレッド・セーフでない場合は、fenced モードの呼び出しごとに、新しい fenced モード・プロセスが作成および破棄されます。 *keepfenced* が *no* に設定されていて、実行されているルーチンがスレッド・セーフである場合は、fenced モード・プロセスは持続しますが、呼び出しのために作成されたスレッドは終了します。 *keepfenced* が *yes* に設定されている場合は、fenced モード・プロセスまたはスレッドが後続の fenced モード呼び出しのために再使用されます。データベース・マネージャーが停止されると、未解決の fenced モード・プロセスおよびスレッドすべてが終了します。

このパラメーターを *yes* に設定すると、追加のシステム・リソースが、活動化されている fenced モード・プロセスごとに、最大で *fenced_pool* パラメーターに含まれている値まで、データベース・マネージャーによって使用される結果になります。新しい処理は、既存の fenced モード・プロセスが使用可能でなく、後続の fenced ルーチン呼び出しを処理できない場合にのみ作成されます。 *fenced_pool* が 0 に設定されている場合は、このパラメーターは無視されます。

推奨: fenced モード要求の数が fenced でないモード要求に比べて相対的に大きく、しかもシステム・リソースが制約されていない環境では、このパラメーターは *yes* に設定できます。こうすれば、呼び出しの処理には既存の fenced モード・プロセスが使用されるので、初期 fenced モード・プロセス作成オーバーヘッドを回避することで、fenced モード・プロセスのパフォーマンスが向上します。特に、Java ルーチンの場合は、こうすることによって、Java 仮想マシン (JVM) を始動するコストが節減され、非常に大幅なパフォーマンスの向上がもたらされます。

たとえば、銀行用 OLTP の貸借トランザクション・アプリケーションでは、各トランザクションを実行するコードは、分散モード・プロセスで処理されるストアード・プロシージャーで実行することができます。このアプリケーションでは、メイン・ワークロードは分散モード・プロセスの外側で実行されます。このパラメーターが *no* に設定されている場合は、それぞれのトランザクションごとに新しい fenced モード・プロセスを作成するオーバーヘッドが発生して、大幅なパフォーマンスの低下を招きます。ただし、このパラメーターを *yes* に設定すると、各トランザクションが既存の分散モード・プロセスを使用しようとして、このオーバーヘッドを避けることになります。

以前のバージョンの DB2 では、このパラメーターは *keepdari* という名前で呼ばれていました。

関連資料:

- 448 ページの『fenced_pool - fenced プロセスの最大数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

num_initfenced - fenced プロセスの初期数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 0 [0 — *max_connections* + (*maxagents* - *max_coordagents*)]

このパラメーターは、DB2START 時に db2fmp プールに作成される、非スレッド化アイドル db2fmp プロセスの初期数を示します。このパラメーターを設定すると、非スレッド・セーフ C ルーチンおよび COBOL ルーチンを実行する場合に初期起動時間が短縮されます。 *keepfenced* が指定されていない場合、このパラメーターは無視されます。

DB2START 時に多くの db2fmp プロセスを開始するよりも、*fenced_pool* をシステムにとって適切なサイズに設定することの方がはるかに重要です。

以前のバージョンの DB2 では、このパラメーターは *num_initdaris* という名前で呼ばれていました。

関連資料:

- 449 ページの『keepfenced - fenced プロセスの保持』
- 448 ページの『fenced_pool - fenced プロセスの最大数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

ロギングおよびリカバリー

重要なデータが失われないようにするには、環境のリカバリーが非常に大切になります。環境を管理し、十分なレベルまでデータまたはトランザクションをリカバリーするのに使用できるパラメーターがいくつかあります。この種のパラメーターは、次のカテゴリーに分けることができます。

- 452 ページの『データベース・ログ・ファイル』
- 463 ページの『データベース・ログ・アクティビティー』
- 474 ページの『リカバリー』
- 487 ページの『分散作業単位リカバリー』

データベース・ログ・ファイル

次のパラメーターは、データベースのロギングで使用するファイルの数、サイズおよび状況について記述しています。

- 『logfilsiz - ログ・ファイルのサイズ』
- 453 ページの『loghead - 最初のアクティブ・ログ・ファイル』
- 453 ページの『logpath - ログ・ファイルのロケーション』
- 454 ページの『logprimary - 1 次ログ・ファイル数』
- 456 ページの『logsecond - 2 次ログ・ファイル数』
- 457 ページの『max_log - トランザクション当たりの最大ログ』
- 458 ページの『mirrorlogpath - ミラー・ログ・パス』
- 459 ページの『newlogpath - データベース・ログ・パスの変更』
- 461 ページの『num_log_span - 番号ログ幅』
- 461 ページの『overflowlogpath - オーバーフロー・ログ・パス』

logfilsiz - ログ・ファイルのサイズ

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲]

UNIX	1000 [4 — 262 144]
Windows	1000 [4 — 262 144]

単位 ページ (4 KB)

このパラメーターは、1 次および 2 次ログ・ファイルの各サイズを定義します。これらのログ・ファイルのサイズは、ファイルがいっぱいになり、新しいログ・ファイルが必要になる前に書き込めるログ・レコード数を制限します。

1 次および 2 次ログ・ファイルの使用は、ログ・ファイルがいっぱいになったときに実行されるアクションと同様、実行されているロギングのタイプによって次のように異なります。

- 循環ロギング

  1 次ログ・ファイルは、そのファイルに記録された変更がコミットされたときに再利用可能になります。ログ・ファイルのサイズが小さく、しかもアプリケーションが変更をコミットせずにデータベースへの変更を頻繁に処理すると、1 次ログ・ファイルがすぐにいっぱいになる可能性があります。 すべての 1 次ログ・ファイルがいっぱいになると、データベース・マネージャーは新しいログ・レコードを保留するために 2 次ログ・ファイルを割り振ります。

- ログ保存ロギング

  1 次ログ・ファイルがいっぱいになると、ログがアーカイブされ、新しい 1 次ログ・ファイルが割り振られます。

推奨: ログ・ファイルのサイズについては、次のように 1 次ログ・ファイルの数とのバランスを取る必要があります。

- データベースに対して行われる、ログ・ファイルを早急にいっぱいにする更新、削除または挿入トランザクションの数が多い場合は、 *logfilsiz* の値を増やす必要があります。

  注: ログ・ファイル・サイズの最大値とログ・ファイルの数 (*logprimary* + *logsecond*) の最大値まで使用すると、全体のアクティブ・ログ・スペースも 256 GB という最大値になります。

  ログ・ファイルが小さすぎると、古いログ・ファイルのアーカイブ、新しいログ・ファイルの割り振り、および使用可能ログ・ファイルの待機などのオーバーヘッドのために、システム・パフォーマンスに影響を与える場合があります。
- ディスク・スペースが不足している場合、1 次ログはこのサイズで事前割り振りされるので、*logfilsiz* の値を減らす必要があります。

  ログ・ファイルが大きすぎると、いくつかのメディアではログ・ファイル全体を保存できない可能性があるので、アーカイブ済みログ・ファイルおよびログ・ファイルのコピーの管理上、柔軟性が失われる可能性があります。

ログ保存を使用している場合、最後のアプリケーションがデータベースから切断されると、現在のアクティブ・ログ・ファイルはクローズされ、切り捨てられます。データベースへの接続が行われると、次のログ・ファイルが使用されます。したがって、並行アプリケーションのロギング要件がはっきりしている場合は、余分なスペースを割り振らないログ・ファイル・サイズを判別できる可能性があります。

関連資料:

- 454 ページの『logprimary - 1 次ログ・ファイル数』
- 456 ページの『logsecond - 2 次ログ・ファイル数』
- 471 ページの『softmax - リカバリー範囲およびソフト・チェックポイント・インターバル』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

loghead - 最初のアクティブ・ログ・ファイル

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターには、現在アクティブなログ・ファイルの名前が含まれます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

logpath - ログ・ファイルのロケーション

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターには、ロギング目的で使用されている現行パスが含まれます。このパラメーターは、*newlogpath* パラメーターに対する変更が有効になった後で、データベース・マネージャーによって設定されているので、直接変更することはできません。

データベースが作成されると、データベースのリカバリー・ログ・ファイルが、データベースが入っているディレクトリーのサブディレクトリーに作成されます。デフォルトでは、データベースを入れるために作成されたディレクトリーの下にある SQLOGDIR という名前のサブディレクトリーです。

関連資料:

- 459 ページの『newlogpath - データベース・ログ・パスの変更』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

logprimary - 1 次ログ・ファイル数

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲] 3 [2 - 256]

単位 カウンター

割り振られるタイミング

- データベースが作成されるとき
- ログが別のロケーションに移動されるとき (これが発生するのは、*logpath* パラメーターの更新時)
- すべてのユーザーが切断した後の次回データベース接続中に、このパラメーター (*logprimary*) の値を大きくした後
- ログ・ファイルがアーカイブされて、新規ログ・ファイルが割り振られるとき (*logretain* または *userexit* パラメーターが使用可能になっている必要があります)
- *logfilsiz* パラメーターが変更されている場合は、アクティブ・ログ・ファイルは、すべてのユーザーが切断された後の次回データベース接続中にサイズ変更されます。

解放されるタイミング このパラメーターの値が小さくならない限り、解放されることはありません。パラメーターの値を小さくした場合は、データベースへの次回接続時に、不要のログ・ファイルが削除されます。

1 次ログ・ファイル数によって、リカバリー・ログ・ファイルに割り振られる固定量のストレージが設定されます。このパラメーターを使用すると、事前割り振りされる 1 次ログ・ファイルの数を指定できます。

循環ロギング下では、1 次ロギングが順次繰り返して使用されます。つまり、あるログがいっぱいになると、そのシーケンス内の次のログが使用可能であれば、それが使用されることになります。ログが使用可能と見なされるのは、ログの中にログ・レコードがある作業単位がすべてコミットおよびロールバックされている場合です。シーケンス内の次の 1 次ログが使用不可の場合は、2 次ログが割り振られて使用されます。シーケンス内の次の 1 次ログが使用可能になるか、*logsecond* パラメーターによって設けられた限度に達するまで、追加の 2 次ログが割り振られて使用されます。これらの 2 次ログは、データベース・マネージャーで必要とされなくなると、動的に割り振り解除されます。

1 次ログ・ファイルと 2 次ログ・ファイルの数は、次の式に適合する必要があります。

- *logsecond* の値が -1 の場合、*logprimary* <= 256。
- *logsecond* の値が -1 以外の場合、 (*logprimary* + *logsecond*) <= 256。

推奨: このパラメーターの値として選択される値は、使用されるロギングのタイプ、ログ・ファイルのサイズ、および処理環境のタイプ (たとえば、トランザクションの長さやコミットの頻度など) を含む、数多くの要因に応じて決まります。

この値を大きくすると、ログ用のディスク所要量が増えます。 1 次ログ・ファイルは、データベースへの最初の接続時に事前割り振りされるからです。

2 次ログ・ファイルが割り振られる頻度が高いと思われる場合は、ログ・ファイル・サイズ (*logfilsiz*) を大きくするか、1 次ログ・ファイルの数を増やすことで、システム・パフォーマンスを上げることもできます。

データベースにアクセスする頻度が高くない場合は、ディスク・ストレージを節約するために、このパラメーターを 2 に設定します。ロールフォワード・リカバリーが使用可能になっているデータベースの場合は、ほとんど即時に新しいログを割り振るオーバーヘッドを回避するために、このパラメーターはもっと大きい値に設定します。

1 次ログ・ファイルをサイズ変更するには、データベース・システム・モニターを使用できます。ある期間にわたって次のモニター値を監視すると、進行中の要件は平均値に表れる場合が多いので、優れたチューニングの決定に役立ちます。

- *sec_log_used_top (使用される最大 2 次ログ・スペース)*
- *tot_log_used_top (使用される最大合計ログ・スペース)*
- *sec_logs_allocated (現在割り振られている 2 次ログ数)*

関連資料:

- 452 ページの『logfilsiz - ログ・ファイルのサイズ』
- 456 ページの『logsecond - 2 次ログ・ファイル数』
- 468 ページの『logretain - ログ保存使用可能』
- 473 ページの『userexit - ユーザー出口使用可能』
- 「システム・モニター *ガイドおよびリファレンス*」の『sec_log_used_top 使用された最大 2 次ログ・スペース : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『tot_log_used_top 使用された最大合計ログ・スペース : モニター・エレメント』

- 「システム・モニター ガイドおよびリファレンス」の『sec_logs_allocated 現在割り振られている 2 次ログ : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

logsecond - 2 次ログ・ファイル数

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
伝搬クラス	即時
デフォルト[範囲]	2 [-1; 0 - 254]
単位	カウンター
割り振られるタイミング	*logprimary* が不十分なとき、必要に応じて (下記の詳細を参照してください)
解放されるタイミング	必要なくなるというデータベース・マネージャーによる決定に応じて徐々に

このパラメーターは、リカバリー・ログ・ファイルとして作成および使用される (必要な場合のみ) 2 次ログ・ファイルの数を指定します。 1 次ログ・ファイルがいっぱいになると、2 次ログ・ファイル (サイズ *logfilsiz*) が、必要に応じて一度に 1 つずつ、このパラメーターがコントロールする最大数まで割り振られます。 2 次ログ・ファイルがこのパラメーターによって許可されている数を超えて必要とされた場合は、アプリケーションにエラー・コードが戻され、データベースはシャットダウンされます。

logsecond を -1 に設定した場合は、データベースは、アクティブ・ログ・スペースが無限として構成されます。したがって、データベース上で実行途中のトランザクションのサイズや数に制限はありません。 *logsecond* を -1 に設定した場合でも、DB2 がアクティブ・ログ・パスに保持する必要があるログ・ファイルの数を指定する場合は、やはり構成パラメーター *logprimary* および *logfilsiz* を使用します。 DB2 はログ・ファイルからログ・データを読み取る必要があるが、そのログ・ファイルがアクティブ・ログ・パスにない場合は、DB2 は、ユーザー出口プログラムを呼び出して、そのログ・ファイルをアーカイブからアクティブ・ログ・パスに取り出します (DB2 がファイルをオーバーフロー・ログ・パスに取り出すのは、オーバーフロー・ログ・パスが構成してある場合です)。ログ・ファイルが取り出されると、DB2 はこのファイルをアクティブ・ログ・パス内でキャッシュに入れるので、他に同じファイルからログ・データを読み取る場合は、読み取りが高速化されます。 DB2 は、必要に応じて、これらのログ・ファイルの検索、キャッシング、および除去を管理します。

ログ・パスがロー・デバイスである場合は、*logsecond* を -1 に設定するためには、*overflowlogpath* 構成パラメーターを構成する必要があります。

logsecond を -1 に設定することによって、作業単位のサイズにも並行作業単位の数にも制限がなくなります。ただし、アーカイブからログ・ファイルを取り出す必要があるため、ロールバック (保管点レベルと作業単位レベルの両方での) が非常に遅くなる可能性があります。クラッシュ・リカバリーも同じ理由で非常に遅くなる恐れがあります。 DB2 は、db2diag.log および管理通知ログにメッセージを書き込んで、現行セットのアクティブ作業単位が 1 次ログ・ファイル数を超えていることを警告します。これは、ロールバックやクラッシュ・リカバリーが極端に遅くなる恐れがあることを示します。

logsecond を -1 に設定するには、*userexit* 構成パラメーターを *yes* に設定する必要があります。

推奨: データベースが周期的に大量のログ・スペースを必要とする場合は、2 次ログ・ファイルを使用してください。たとえば、1 か月に一度実行されるアプリケーションの場合は、1 次ログ・ファイルで用意されているログ・スペースを超えるスペースが必要になる可能性があります。2 次ログ・ファイルでは永続ファイル・スペースを必要としないので、こうした状況では利点になります。

関連資料:

- 452 ページの『logfilsiz - ログ・ファイルのサイズ』
- 454 ページの『logprimary - 1 次ログ・ファイル数』
- 468 ページの『logretain - ログ保存使用可能』
- 473 ページの『userexit - ユーザー出口使用可能』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 461 ページの『overflowlogpath - オーバーフロー・ログ・パス』

max_log - トランザクション当たりの最大ログ

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
伝搬クラス	即時
デフォルト[範囲]	0 [0 — 100]
単位	パーセント

この値が 0 ではない場合、このパラメーターは、 1 つのトランザクションで消費できるアクティブ・ログ・スペースのパーセントを示します。

値を 0 に設定すると、 1 つのトランザクションで消費できるスペース (アクティブ・ログ・スペース総量のパーセントとして) に制限がなくなります。これは、バージョン 8 より前のトランザクションの動作です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 461 ページの『num_log_span - 番号ログ幅』

mirrorlogpath - ミラー・ログ・パス

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	NULL [有効なパスまたは装置]

このパラメーターによって、ストリングを 242 バイトまでミラー・ログ・パスに指定することができます。ストリングがパス名を示す必要があり、これは相対パス名ではなく、完全修飾パス名でなければなりません。

注: パーティション・データベース環境では、ノード番号が自動的にパスに付加されます。このことは、複数論理ノード構成のパスの固有性を維持するために行われます。

mirrorlogpath が構成されている場合は、DB2 は、ログ・パスとミラー・ログ・パスの両方にアクティブ・ログ・ファイルを作成します。ログ・データはすべて両方のパスに書き込まれます。ミラー・ログ・パスには、アクティブ・ログ・ファイルのセットの複製があるので、ディスク・エラーや人間によるエラーで一方のパスにあるアクティブ・ログ・ファイルが破棄された場合でも、データベースはそのまま機能し続けることができます。

ミラー・ログ・パスが変更された場合は、ログ・ファイルが古いログ・パスにある可能性があります。こうしたログ・ファイルはアーカイブされていない場合があるので、これらのログ・ファイルは、手動でアーカイブしておく必要があります。また、このデータベース上で複製を実行する場合、ログ・パスが変更される前からのログ・ファイルを必要とします。 Yes に設定された「ユーザー出口使用可能 (*userexit*)」データベース構成パラメーターでデータベースが構成されている場合、およびすべてのログ・ファイルが DB2 によって自動的に、または手操作によってアーカイブされている場合、DB2 はログ・ファイルを検索して複製処理を完了することができるようになります。それ以外の場合、ファイルは、古いミラー・ログ・パスから新しいミラー・ログ・パスにコピーできます。

logpath または *newlogpath* で、ログ・ファイルの保管されているロケーションとしてロー・デバイスを指定する場合、ミラー・ロギング (*mirrorlogpath* によって示される) を行うことはできません。 *logpath* または *newlogpath* で、ログ・ファイルの保管されているロケーションとしてファイル・パスを指定する場合、ミラー・ロギングは可能であり、 *mirrorlogpath* でファイル・パスを指定する必要もあります。

推奨: ログ・ファイルの場合とまったく同様に、ミラー・ログ・ファイルの場合も、入出力の多くない物理ディスクに置く必要があります。

このパスは、1 次ログ・パスとは別の装置に置くことをお勧めします。

データベース・システム・モニターを使用すると、データベース・ロギングに関連する入出力の数を追跡できます。

以下のデータ・エレメントは、データベース・ロギングに関連した入出力活動の量を戻します。 他のディスク入出力アクティビティーに関する情報を収集し、入出力アクティビティーの 2 つのタイプを比較するには、オペレーティング・システムのモニター・ツールを使用できます。

- *log_reads* (読み取られたログ・ページの数)。
- *log_writes* (書き込まれたログ・ページの数)。

関連資料:

- 453 ページの『logpath - ログ・ファイルのロケーション』
- 459 ページの『newlogpath - データベース・ログ・パスの変更』
- 「システム・モニター *ガイドおよびリファレンス*」の『log_reads 読み取られたログ・ページの数 : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『log_writes 書き込まれたログ・ページの数 : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 461 ページの『overflowlogpath - オーバーフロー・ログ・パス』

newlogpath - データベース・ログ・パスの変更

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	NULL [有効なパスまたは装置]

このパラメーターによって、ストリングを 242 バイトまで指定して、ログ・ファイルが保管されるロケーションを変更することができます。ストリングによってパス名またはロー・デバイスを示すことができます。ストリングがパス名を示す場合は、相対パス名ではなく、完全修飾パス名でなければなりません。

注: パーティション・データベース環境では、ノード番号が自動的にパスに付加されます。このことは、複数論理ノード構成のパスの固有性を維持するために行われます。

レプリケーションを使用する場合で、ログ・パスがロー・デバイスであるときは、*overflowlogpath* 構成パラメーターを構成する必要があります。

装置を指定するには、オペレーティング・システムが装置として認識するストリングを指定します。 例:

- Windows NT 上では、`¥¥.¥d:` または `¥¥.¥PhysicalDisk5`

注: ログを装置に書き込めるようにするには、Windows NT バージョン 4.0 サービス・パック 3 またはそれ以降のリリースがインストールされていなければなりません。

- UNIX ベースのプラットフォームでは /dev/rdblog8

注: AIX、 Windows 2000、 Windows NT、 Solaris オペレーティング環境、HP-UX、および Linux プラットフォームの装置のみを指定できます。

新しい設定は、以下の両方が発生するまで *logpath* の値にはなりません:

- データベースが、*database_consistent* パラメーターによって示される整合状態にあります。
- すべてのユーザーがデータベースから切断された。

データベースへの最初の新しい接続が行われると、データベース・マネージャーはログを *logpath* によって指定されたロケーションに移動します。

古いログ・パスにログ・ファイルがある可能性があります。これらのログ・ファイルは、アーカイブされていないことがあります。これらのログ・ファイルは手操作でアーカイブする必要があります。また、このデータベース上でレプリケーションを実行する場合、ログ・パスが変更される前からのログ・ファイルを必要とします。 Yes に設定された「ユーザー出口使用可能 (*userexit*)」データベース構成パラメーターでデータベースが構成されている場合、およびすべてのログ・ファイルが DB2 によって自動的に、または手操作によってアーカイブされている場合、DB2 はログ・ファイルを検索してレプリケーション処理を完了することができるようになります。それ以外の場合は、ファイルを古いログ・パスから新しいログ・パスにコピーすることができます。

logpath または *newlogpath* で、ログ・ファイルの保管されているロケーションとしてロー・デバイスを指定する場合、ミラー・ロギング (*mirrorlogpath* によって示される) を行うことはできません。 *logpath* または *newlogpath* で、ログ・ファイルの保管されているロケーションとしてファイル・パスを指定する場合、ミラー・ロギングは可能であり、 *mirrorlogpath* でファイル・パスを指定する必要もあります。

推奨: 理想としては、ログ・ファイルは他のディスク I/O が**多くない**物理ディスクに置かれることが望まれます。たとえば、ログをオペレーティング・システムまたは大きなボリュームのデータベースと同じディスクに置くことは避けてください。これにより、入出力待ちなどのオーバーヘッドを最小にし、効率的なロギング活動が得られます。

データベース・システム・モニターを使用すると、データベース・ロギングに関連する入出力の数を追跡できます。

モニター・エレメント *log_reads* (読み取られたログ・ページの数) および *log_writes* (書き込まれたログ・ページの数) では、データベース・ロギングに関連する入出力アクティビティーの量を戻します。他のディスク入出力アクティビティーに関する情報を収集し、入出力アクティビティーの 2 つのタイプを比較するには、オペレーティング・システムのモニター・ツールを使用できます。

関連資料:

- 453 ページの『logpath - ログ・ファイルのロケーション』

- 501 ページの『database_consistent - データベースの整合性』
- 「システム・モニター ガイドおよびリファレンス」の『log_reads 読み取られたログ・ページの数 : モニター・エレメント』
- 「システム・モニター ガイドおよびリファレンス」の『log_writes 書き込まれたログ・ページの数 : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

num_log_span - 番号ログ幅

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
伝搬クラス	即時
デフォルト[範囲]	0 [0 — 65 535]
単位	カウンター

この値が 0 ではない場合、このパラメーターは、 1 つのアクティブ・トランザクションで展開できるアクティブ・ログ・ファイルの数を示します。

値が 0 に設定される場合、 1 つのトランザクションが使用できるログ・ファイル数に制限はありません。これは、バージョン 8 より前のトランザクションの動作です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 457 ページの『max_log - トランザクション当たりの最大ログ』

overflowlogpath - オーバーフロー・ログ・パス

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
伝搬クラス	即時
デフォルト[範囲]	NULL [任意の有効なパス]

このパラメーターは、ロギング要件に応じていくつかの関数で使用できます。

- このパラメーターを使用すると、ロールフォワード操作に必要なログ・ファイルを、DB2 が検索するロケーションを指定できます。これは、ROLLFORWARD コ

マンドの OVERFLOW LOG PATH コマンドに似ています。OVERFLOW LOG PATH の場合は、ROLLFORWARD コマンドごとにいつも指定するのに対して、この構成パラメーターは一度設定するだけです。ただし、両方使用すると、その特定のロールフォワード操作に関しては、OVERFLOW LOG PATH オプションが *overflowlogpath* 構成パラメーターを上書きします。

- *logsecond* が -1 に設定されている場合は、*overflowlogpath* を使用すると、アーカイブから取り出されたアクティブ・ログ・ファイルを、DB2 が保管するディレクトリーを指定できます (アクティブ・ログ・ファイルがアクティブ・ログ・パスになくなっている場合は、ロールバック操作のために、アクティブ・ログ・ファイルを取り出す必要があります)。overflowlogpath が指定されていない場合は、DB2 はログ・ファイルを取り出してアクティブ・ログ・パスに入れます。*overflowlogpath* を使用すると、取り出されたログ・ファイルを、DB2 が保管する追加のリソースを用意できます。利点としては、入出力コストが異なるディスクに拡散することや、アクティブ・ログ・パスに保管できるログ・ファイルが増えることなどがあります。
- db2ReadLog API (DB2 V8 より前では、db2ReadLog は sqlurlog と呼ばれていた) を、たとえば、レプリケーションを使用する場合で、*overflowlogpath* を使用すると、この API 用として必要なログ・ファイルを、DB2 が検索するロケーションを指定できます。ログ・ファイルが (アクティブ・ログ・パスにもオーバーフロー・ログ・パスにも) 見つからない場合、データベースが *userexit* を使用可能にして構成されていると、DB2 はログ・ファイルを検索します。*overflowlogpath* を使用すると、DB2 が取り出されたログ・ファイルを保管するためのディレクトリーも指定できます。利点としては、アクティブ・ログ・パスの入出力コストの削減、およびアクティブ・ログ・パスに保管できるログ・ファイル数の増加などがあります。
- アクティブ・ログ・パス用としてロー・デバイスを構成してある場合は、*logsecond* を -1 に設定するとき、または db2ReadLog API を使用するときに *overflowlogpath* を構成する必要があります。

overflowlogpath を設定するには、最大 242 バイトのストリングを指定します。ストリングがパス名を示す必要があり、これは相対パス名ではなく、完全修飾パス名でなければなりません。パス名はディレクトリーでなければならず、ロー・デバイスではありません。

注: パーティション・データベース環境では、ノード番号が自動的にパスに付加されます。このことは、複数論理ノード構成のパスの固有性を維持するために行われます。

関連資料:

- 456 ページの『logsecond - 2 次ログ・ファイル数』
- 「*管理 API リファレンス*」の『db2ReadLog - ログの非同期読み取り』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『ROLLFORWARD DATABASE コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

データベース・ログ・アクティビティー

次のパラメーターは、データベース・ロギングのタイプおよびパフォーマンスを制御します。

- 『archretrydelay - エラー時のアーカイブ再試行遅延』
- 464 ページの『blk_log_dsk_ful - ログ・ディスク・フル時のブロック』
- 464 ページの『failarchpath - フェイルオーバー・ログ・アーカイブ・パス』
- 465 ページの『logarchmeth1 - 1 次ログ・アーカイブ方式』
- 466 ページの『logarchmeth2 - 2 次ログ・アーカイブ方式』
- 466 ページの『logarchopt1 - 1 次ログ・アーカイブ・オプション』
- 467 ページの『logarchopt2 - 2 次ログ・アーカイブ・オプション』
- 467 ページの『logindexbuild - 作成済み索引ページのログ』
- 468 ページの『logretain - ログ保存使用可能』
- 469 ページの『mincommit - グループ化するコミット数』
- 470 ページの『numarchretry - エラー時の再試行数』
- 471 ページの『softmax - リカバリー範囲およびソフト・チェックポイント・インターバル』
- 473 ページの『userexit - ユーザー出口使用可能』
- 474 ページの『vendoropt - ベンダー・オプション』

archretrydelay - エラー時のアーカイブ再試行遅延

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] 20 [0 - 65 535]

このパラメーターでは、ログ・ファイルのアーカイブを試行して失敗してから、再びアーカイブを試行するまでに待機する秒数を指定します。再試行が有効であるためには、 *numarchretry* データベース構成パラメーターの値が少なくとも 1 でなければなりません。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

blk_log_dsk_ful - ログ・ディスク・フル時のブロック

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

伝搬クラス 即時

デフォルト[範囲] No [Yes; No]

この構成パラメーターは、DB2 がアクティブ・ログ・パス内に新規ログ・ファイルを作成できないときに、ディスク・フル・エラーが生成されないように設定できます。ただし、DB2 は、ログ・ファイルが正常に作成できるまで、5 分ごとにその作成を試みます。それぞれの試みの後で、DB2 は管理通知ログにメッセージを書き込みます。アプリケーションがログ・ディスク・フル状態のためハングしていることを確認するには、管理通知ログをモニターする以外に方法はありません。ログ・ファイルが正常に作成されるまでは、表データを更新しようとするユーザー・アプリケーションはトランザクションをコミットできません。読み取り専用照会が直接影響を受ける可能性はありませんが、照会が更新要求によってロックされているデータ、または更新アプリケーションによってバッファー・プール内に固定されているデータにアクセスする必要がある場合は、読み取り専用照会もハングするように見えます。

blk_log_dsk_ful を *yes* に設定すると、DB2 がログ・ディスク・フル・エラーを検出したときにアプリケーションがハングするので、ユーザーがエラーを解決し、トランザクションを完了することができます。ディスク・フル状態は、古いログ・ファイルを別のファイル・システムに移動するか、ファイル・システムを拡張して、ハングしているアプリケーションが完了できるようにすることで解決できます。

blk_log_dsk_ful が *no* に設定されている場合は、ログ・ディスク・フル・エラーを受け取ったトランザクションは失敗し、ロールバックされます。状態によっては、トランザクションがログ・ディスク・フル・エラーの原因である場合は、データベースがダウンする場合があります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

failarchpath - フェイルオーバー・ログ・アーカイブ・パス

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] Null [ ]

このパラメーターでは、メディアの問題の影響で 1 次と 2 次 (設定されている場合) のいずれのアーカイブ宛先にもログ・ファイルをアーカイブできない場合に、DB2 がログ・ファイルのアーカイブを試行する宛先のパスを指定します。ここに指定するパスは、ディスクを参照している必要があります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

logarchmeth1 - 1 次ログ・アーカイブ方式

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] オフ[ ]

このパラメーターでは、アーカイブ済みログの 1 次宛先のメディア・タイプを指定します。

関連概念:

- 685 ページの『付録 E. db2adutl コマンドと logarchopt1 および vendoropt データベース構成パラメーターを使ったノード間リカバリー』

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

logarchmeth2 - 2 次ログ・アーカイブ方式

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] オフ[ ]

このパラメーターでは、アーカイブ済みログの 2 次宛先のメディア・タイプを指定します。このパスが指定されている場合、ログ・ファイルはこの宛先と *logarchmeth1* データベース構成パラメーターに指定されている宛先の両方にアーカイブされます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

logarchopt1 - 1 次ログ・アーカイブ・オプション

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] Null [ ]

このパラメーターでは、アーカイブ済みログの 1 次宛先のオプション・フィールドを指定します (必要な場合)。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

logarchopt2 - 2 次ログ・アーカイブ・オプション

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] Null [ ]

このパラメーターでは、アーカイブ済みログの 2 次宛先のオプション・フィールドを指定します (必要な場合)。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

logindexbuild - 作成済み索引ページのログ

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] オフ [オン; オフ]

このパラメーターでは、索引の作成、再作成、または再編成の操作をログに記録しておき、 DB2 ロールフォワード操作中または高可用性災害時リカバリー (HADR) ログ再生プロシージャー中に索引を再構成できるようにするかどうかを指定します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

logretain - ログ保存使用可能

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	No [Recovery; No]

値は以下のとおりです。

- No は、ログが保存されないことを示します。
- Recovery は、ログが保存され、順方向リカバリーに使用できることを示します。

logretain が Recovery に設定されているか、または *userexit* が Yes に設定されている場合は、アクティブ・ログ・ファイルは保存され、ロールフォワード・リカバリーで使用するためのオンライン・アーカイブ・ログ・ファイルになります。これは、ログ保存ロギングと呼ばれます。

logretain が Recovery に設定された後、または *userexit* が Yes に設定された後 (あるいはその両方) は、データベースの全バックアップを行う必要があります。この状態は、*backup_pending* フラグ・パラメーターによって示されます。

logretain が No に設定され、しかも *userexit* が No に設定されている場合は、ログが保存されていないため、データベースに関するロールフォワード・リカバリーは使用不可です。この状態では、データベース・マネージャーはオンライン・アーカイブ・ログ・ファイルを含めて、 *logpath* ディレクトリーのすべてのログ・ファイルを削除し、新しいアクティブ・ログ・ファイルを割り振り、循環ロギングに戻ります。

関連資料:

- 501 ページの『log_retain_status - ログ保存状況標識』
- 473 ページの『userexit - ユーザー出口使用可能』
- 500 ページの『backup_pending - バックアップ・ペンディング標識』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

mincommit - グループ化するコミット数

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
伝搬クラス	即時
デフォルト[範囲]	1 [1 - 25]
単位	カウンター

このパラメーターを使用すると、最小数のコミットが実行されるまで、ディスクへのログ・レコードの書き込みを遅らせることができます。この遅延により、ログ・レコードの書き込みに関連したデータベース・マネージャー・オーバーヘッドが低減できます。したがって、データベースに対して実行中のアプリケーションが複数あり、非常に短い時間内に数多くのコミットがアプリケーションによって要求されるときは、これによってパフォーマンスが上がります。

こうしたコミットのグループ化が行われるのは、このパラメーターの値が 1 より大きい場合で、かつデータベースに接続されているアプリケーションの数がこのパラメーターの値より大か等しい場合だけです。コミットのグループ化が実行されているときは、1 秒が経過するか、コミット要求の数がこのパラメーターの値に等しくなるか、いずれかが生じるまで、アプリケーションのコミット要求は保留される可能性があります。

このパラメーターは、少しずつ (1 ずつなど) 増分すべきです。また、マルチユーザー・テストも使用して、このパラメーターの値を増やして期待される結果が得られるかどうかを検証してください。

このパラメーターに指定された値に対する変更は、即時に有効になります。したがって、すべてのアプリケーションがデータベースから切断されるまで待つ必要はありません。

推奨: 複数の読み取り/書き込みアプリケーションで一般的に並行データベース・コミットが要求される場合は、このパラメーターをデフォルト値から大きくしてください。そうすれば、ロギングが行われる頻度が低くなり、ロギングが行われる度に書き込まれるログ・レコード数が増えるので、ファイル入出力のロギング効率が上がります。

1 秒当たりのトランザクション数をサンプリングして、1 秒当たりのピーク・トランザクション数 (または、そのうちのかなり大きなパーセンテージ) に対処できるように、このパラメーターを調整することもできます。ピーク・アクティビティーに対処できれば、トランザクション集中期間のログ・レコード書き込みのオーバーヘッドが最小化されます。

mincommit を大きくする場合は、*logbufsz* パラメーターも値を大きくして、トランザクション集中期間にログ・バッファーがいっぱいになって、強制的に書き込みが行われることになるのを避ける必要もあります。この場合は、*logbufsz* が次の値に等しいことが必要です。

mincommit * (平均して 1 つのトランザクションで使用されるログ・スペース)

データベース・システム・モニターを使用すると、次の方法でこのパラメーターを調整できます。

- 1 秒当たりのピーク・トランザクション数を計算します。

  普通の 1 日を通してモニター・サンプルを取り、トランザクション集中期間を判別することができます。次のモニター・エレメントを加えることによって、合計トランザクション数を計算できます。

  – *commit_sql_stmts (試みられたコミット・ステートメント数)*
  – *rollback_sql_stmts (試みられたロールバック・ステートメント数)*

  この情報と使用可能なタイム・スタンプを使用して、1 秒当たりのトランザクション数を計算できます。
- 1 トランザクション当たりで使用されるログ・スペースを計算します。

  ある期間にわたって、トランザクション数をサンプリングする技法を使用して、次のモニター・エレメントによって、使用されるログ・スペースの平均を計算できます。

  – *log_space_used (使用される作業単位ログ・スペース)*

関連資料:

- 「システム・モニター *ガイドおよびリファレンス*」の『uow_log_space_used 作業単位ログ・スペース : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『commit_sql_stmts 試行されたコミット・ステートメント : モニター・エレメント』
- 「システム・モニター *ガイドおよびリファレンス*」の『rollback_sql_stmts 試行されたロールバック・ステートメント : モニター・エレメント』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

numarchretry - エラー時の再試行数

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] 5 [0 - 65 535]

このパラメーターでは、DB2 が、ログ・ファイルをフェイルオーバー・ディレクトリーにアーカイブする前に、ログ・ファイルを 1 次または 2 次アーカイブ・ディレクトリーにアーカイブしようと試行する回数を指定します。このパラメーターは、*failarchpath* データベース構成パラメーターが設定されている場合にのみ使用できます。 *numarchretry* が設定されていない場合、 DB2 は 1 次または 2 次ログ・パスへのアーカイブの試行を継続します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

softmax - リカバリー範囲およびソフト・チェックポイント・インターバル

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	100 [1 - 100 * *logprimary*]
単位	1 つの 1 次ログ・ファイルのサイズに占めるパーセンテージ

このパラメーターは、次の場合に使用します。

- クラッシュ (たとえば、電源障害など) の後でリカバリーする必要のあるログの数に影響を与える場合。たとえば、デフォルト値が使用されている場合、データベース・マネージャーは、リカバリーする必要のあるログの数を 1 に保持しようとします。このパラメーターの値として 300 を指定した場合、データベース・マネージャーは、リカバリーする必要のあるログの数を 3 に保持しようとします。

  クラッシュ・リカバリーに必要なログの数に影響を与えるために、データベース・マネージャーは、このパラメーターを使用してページ・クリーナーを起動して、指定されたリカバリー・ウィンドウよりも古いページについては、確実にディスクに書き込まれているようにします。
- ソフト・チェックポイントの頻度を決定する場合。

電源障害などのようなイベントの結果、データベース障害が発生した時点では、データベースに次のような変更がもたらされている可能性があります。

- コミットされてはいないが、バッファー・プールのデータを更新してしまった変更
- コミットされているが、バッファー・プールからディスクに書き込まれてはいない変更
- コミットされ、しかもバッファー・プールからディスクに書き込まれてしまった変更

データベースが再始動されると、ログ・ファイルを使用して、データベースのクラッシュ・リカバリーが実行されます。これにより、データベースが整合状態のまま

に置かれる (つまり、コミット済みトランザクションはすべてがデータベースに適用され、非コミット・トランザクションはすべてがデータベースに適用されていない) ことが保証されます。

ログ・ファイルからデータベースに適用される必要のあるレコードを判別するために、データベース・マネージャーはログ・コントロール・ファイルを使用します。このログ・コントロール・ファイルは、定期的にディスクに書き込まれ、このイベントの頻度に応じて、データベース・マネージャーがコミット済みトランザクションのログ・レコードを適用したり、すでにバッファー・プールからディスクに書き込まれてしまった変更を記述するログ・レコードを適用したりします。これらのログ・レコードがデータベースに影響を及ぼすことはありませんが、ログ・レコードを適用することによって、データベース再始動処理に多少のオーバーヘッドが発生します。

ログ・コントロール・ファイルは、ログ・ファイルがいっぱいになったとき、およびソフト・チェックポイント時には、必ずディスクに書き込まれます。この構成パラメーターを使用して、追加のソフト・チェックポイントを起動することができます。

ソフト・チェックポイントのタイミングは、*logfilsiz* に占めるパーセンテージとして与えられる、「現行の状態」と「記録された状態」の間の差に基づいて決まります。「記録された状態」が、ディスク上のログ・コントロール・ファイルに示されている最も古い有効ログ・レコードによって決まるのに対して、「現行の状態」は、メモリー内のコントロール情報によって決まります (最も古い有効ログ・レコードが、リカバリー処理で最初に読み込まれるログ・レコードになります)。ソフト・チェックポイントが取られるのは、次の公式による計算値がこのパラメーターの値より大か等しい場合です。

```
( (記録された状態と現行の状態の間のスペース) / logfilsiz ) * 100
```

推奨: このパラメーターの値は、許容リカバリー・ウィンドウが 1 つのログ・ファイルより大きいか小さいかによって、大きくしたり小さくしたりすることができます。このパラメーターの値を小さくすると、データベース・マネージャーがページ・クリーナーを起動する回数が増え、ソフト・チェックポイントを取る頻度も高くなります。これらのアクションによって、処理する必要のあるログ・レコードの数と、クラッシュ・リカバリー時に処理される重複ログ・レコードの数が、両方とも減らせます。

ただし、ページ・クリーナーの起動回数が増え、ソフト・チェックポイントの頻度が増すと、データベース・ロギングに関連したオーバーヘッドが増加し、データベース・マネージャーのパフォーマンスに影響する可能性があることに注意してください。また、ソフト・チェックポイントの頻度が高くなっても、次のような場合には、データベースの再始動に必要な時間が短縮できない可能性もあります。

- 非常に長いトランザクションで、コミット・ポイントが少ない場合。
- 非常に大きいバッファー・プールがあり、コミット済みトランザクションが入っているページがディスクに元どおり書き込まれる頻度があまり高くない場合 (非同期ページ・クリーナーを使用すると、この状態を回避できます)。

上記の事例のどちらでも、メモリーに保持されているコントロール情報が変更される頻度は高くないし、ログ・コントロール情報は、変更されていない限り、ディスクに書き込んでも利点はありません。

関連資料:

- 452 ページの『logfilsiz - ログ・ファイルのサイズ』
- 454 ページの『logprimary - 1 次ログ・ファイル数』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

userexit - ユーザー出口使用可能

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	No [Yes; No]

このパラメーターが使用可能な場合は、*logretain* パラメーターの設定にかかわらず、ログ保存ロギングが実行されます。このパラメーターは、ユーザー出口プログラムをログ・ファイルのアーカイブと検索のために使用する必要があることも示します。データベース・マネージャーがログ・ファイルをクローズすると、ログ・ファイルがアーカイブされます。ログ・ファイルは、ROLLFORWARD ユーティリティーがデータベースを格納するためにログ・ファイルの使用が必要になると、検索されます。

logretain、*userexit*、またはこれらのパラメーターの両方が使用可能になった後は、データベースのフル・バックアップを作成する必要があります。この状態は、*backup_pending* フラグ・パラメーターによって示されます。

これらのパラメーターの両方が選択解除されている場合は、ログが保存されないので、そのデータベースではロールフォワード・リカバリーが使用できなくなります。この場合は、データベース・マネージャーはオンライン・アーカイブ・ログ・ファイルを含めて、 *logpath* ディレクトリーのすべてのログ・ファイルを削除し、新しいアクティブ・ログ・ファイルを割り振り、循環ロギングに戻ります。

関連資料:

- 「*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*」の『データベース・リカバリー用のユーザー出口』
- 468 ページの『logretain - ログ保存使用可能』
- 502 ページの『user_exit_status - ユーザー出口状況標識』
- 500 ページの『backup_pending - バックアップ・ペンディング標識』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『ROLLFORWARD DATABASE コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

vendoropt - ベンダー・オプション

構成タイプ データベース

適用

- ローカルとリモート・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

デフォルト[範囲] Null [ ]

このパラメーターでは、バックアップ、リストア、またはロード・コピー操作中にストレージ・システムと通信するために DB2 が使用することのできる追加パラメーターを指定します。

関連概念:

- 685 ページの『付録 E. db2adutl コマンドと logarchopt1 および vendoropt データベース構成パラメーターを使ったノード間リカバリー』

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

リカバリー

次のパラメーターは、データベース・リカバリーのさまざまな要素を制御します。

- 475 ページの『autorestart - 自動再始動使用可能』
- 476 ページの『dft_loadrec_ses - ロード・リカバリー・セッションのデフォルト数』
- 481 ページの『indexrec - 索引再作成時点』
- 483 ページの『num_db_backups - データベース・バックアップの数』
- 484 ページの『rec_his_retentn - リカバリー履歴保存期間』
- 485 ページの『trackmod - 変更されたページの追跡使用可能化』

487 ページの『分散作業単位リカバリー』も参照してください。

次のパラメーターは、Tivoli Storage Manager (TSM) で作業する際に使用されます。

- 485 ページの『tsm_mgmtclass - Tivoli Storage Manager 管理クラス』
- 486 ページの『tsm_nodename - Tivoli Storage Manager ノード名』
- 486 ページの『tsm_owner - Tivoli Storage Manager 所有者名』
- 487 ページの『tsm_password - Tivoli Storage Manager パスワード』

以下は、高可用性災害時リカバリー (HADR) に関するパラメーターです。

- 476 ページの『hadr_db_role - HADR データベース役割』
- 477 ページの『hadr_local_host - HADR ローカル・ホスト名』
- 478 ページの『hadr_local_svc - HADR ローカル・サービス名』
- 478 ページの『hadr_remote_host - HADR リモート・ホスト名』
- 479 ページの『hadr_remote_inst - リモート・サーバーの HADR インスタンス名』
- 479 ページの『hadr_remote_svc - HADR リモート・サービス名』
- 480 ページの『hadr_syncmode - 対等状態にあるログ書き込みのための HADR 同期モード』
- 481 ページの『hadr_timeout - HADR タイムアウト値』

autorestart - 自動再始動使用可能

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

伝搬クラス 即時

デフォルト[範囲] On [On; Off]

このパラメーターがオンに設定されると、アプリケーションがデータベースに接続するときに必要であればデータベース・マネージャーは自動的にデータベース再始動ユーティリティーを呼び出します。クラッシュ・リカバリー は、データベース再始動ユーティリティーによって実行される操作です。この操作が実行されるのは、アプリケーションがデータベースに接続されている最中に、そのデータベースが異常終了した場合です。データベースの異常終了は、電源障害やシステム・ソフトウェア障害が原因で起こる場合があります。データベース・バッファー・プールに入っていても、障害の発生時にディスクに書き込まれていなかったコミット済みトランザクションには、すべてクラッシュ・リカバリーが適用されます。また、非コミット・トランザクションは、ディスクに書き込まれていたとしても、この操作によってすべてバックアウトされます。

autorestart が使用可能になっていない場合は、クラッシュ・リカバリーが実行される必要がある (再始動される必要がある) データベースに接続を試みたアプリケーションは、SQL1015N エラーを受け取ることになります。この場合は、そのアプリケーションでデータベース再始動ユーティリティーを呼び出すか、あるいはユーザーがリカバリー・ツールの再始動操作を選択することで、データベースを再始動することができます。

関連概念:

- 「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」の『クラッシュ・リカバリー』

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dft_loadrec_ses - ロード・リカバリー・セッションのデフォルト数

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

伝搬クラス 即時

デフォルト[範囲] 1 [1 - 30 000]

単位 カウンター

このパラメーターは、表ロードのリカバリー中に使用されるデフォルトのセッション数を指定します。この値は、ロード・コピーの検索に使用される、適正な入出力セッション数に設定してください。ロード・コピーの検索は、リストアに似た操作です。環境変数 DB2LOADREC で指定されたコピー・ロケーション・ファイル内の項目によって、このパラメーターをオーバーライドすることができます。

ロード検索に使用されるデフォルトのバッファー数は、このパラメーターの値に 2 を加えた数になります。コピー・ロケーション・ファイル内のバッファー数もオーバーライドが可能です。

このパラメーターは、ロールフォワードができる場合にのみ適用されます。

関連概念:

- 「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」の『ロードの概要』

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 607 ページの『その他の変数』

hadr_db_role - HADR データベース役割

構成タイプ データベース

適用

- ローカルとリモート・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 通知

このパラメーターでは、データベースがオンラインかオフラインかにかかわらず、データベースの現在の役割を示します。有効な値は STANDARD、PRIMARY、または STANDBY です。

注: GET SNAPSHOT FOR DATABASE コマンドを実行すると、データベースがオンラインの場合にのみ、高可用性災害時リカバリー (HADR) 状態を戻します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

hadr_local_host - HADR ローカル・ホスト名

構成タイプ データベース

適用

- ローカルとリモート・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 構成可能

デフォルト Null

このパラメーターでは、高可用性災害時リカバリー (HADR) TCP 通信のローカル・ホストを指定します。ホスト名または IP アドレスのいずれかを使用できます。ホスト名が指定されてホスト名が複数の IP アドレスにマップされると、エラーが戻され、HADR は開始しません。ホスト名が複数の IP アドレスにマップされる場合には (1 次およびスタンバイに同じホスト名を指定したとしても)、1 次およびスタンバイは結局このホスト名を異なる複数の IP アドレスにマップします。それは、一部の DNS サーバーが IP アドレス・リストを非 deterministic 順で戻すためです。

ホスト名の形式は myserver.ibm.com です。 IP アドレスの形式は ″12.34.56.78″ です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

hadr_local_svc - HADR ローカル・サービス名

構成タイプ データベース

適用

- ローカルとリモート・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 構成可能

デフォルト Null

このパラメーターでは、ローカルの高可用性災害時リカバリー (HADR) プロセスが接続を受け入れる TCP サービス名またはポート番号を指定します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

hadr_remote_host - HADR リモート・ホスト名

構成タイプ データベース

適用

- ローカルとリモート・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 構成可能

デフォルト Null

このパラメーターでは、リモートの高可用性災害時リカバリー (HADR) ノードの TCP/IP ホスト名または IP アドレスを指定します。 *hadr_local_host* と同様に、このパラメーターは必ず 1 つの IP アドレスにのみマップされます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

hadr_remote_inst - リモート・サーバーの HADR インスタンス名

構成タイプ データベース

適用

- ローカルとリモート・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 構成可能

デフォルト Null

このパラメーターでは、リモート・サーバーのインスタンス名を指定します。 DB2 コントロール・センターなどの管理ツールは、このパラメーターを使用してリモート・サーバーと接続します。高可用性災害時リカバリー (HADR) も、接続を要求しているリモート・データベースが宣言済みのリモート・インスタンスに属しているかどうかをチェックします。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

hadr_remote_svc - HADR リモート・サービス名

構成タイプ データベース

適用

- ローカルとリモート・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 構成可能

デフォルト Null

このパラメーターでは、リモートの高可用性災害時リカバリー (HADR) ノードが使用する TCP サービス名またはポート番号を指定します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

hadr_syncmode - 対等状態にあるログ書き込みのための HADR 同期モード

構成タイプ データベース

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] NEARSYNC [ASYNC; SYNC]

このパラメーターでは同期モードを指定します。システムどうしが対等状態にある場合に、 1 次ログ書き込みをスタンバイと同期する方法を指示します。有効な値は次のとおりです。

SYNC このモードはトランザクション損失に対する最大限の保護を提供しますが、トランザクション応答時間のコストも最大です。

NEARSYNC このモードは、トランザクション損失に対する保護が幾分緩くなる代わりに、 SYNC モードよりもトランザクション応答時間が短くなります。

ASYNC このモードは 1 次障害時にトランザクション損失の確率が最も高くなる代わりに、 3 種類のモードの中ではトランザクション応答時間が最も短くなります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

hadr_timeout - HADR タイムアウト値

構成タイプ データベース

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 120 [1 - 655 535]

このパラメーターでは、高可用性災害時リカバリー (HADR) プロセスが通信の試行が失敗したと判断するまでに待機する時間 (秒数) を指定します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

indexrec - 索引再作成時点

構成タイプ データベースおよびデータベース・マネージャー

適用

- ローカルとリモート・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]

UNIX データベース・マネージャー
restart [restart; access]

Windows データベース・マネージャー
restart [restart; access]

データベース Use system setting [system; restart; access]

このパラメーターでは、データベース・マネージャーが無効な索引の再作成を試みる時点を指示します。このパラメーターには、指定できる設定が次のように 3 つあります。

SYSTEM *システム設定値を使用* では、無効な索引は、データベース・マネージャー構成に指定されているタイミングで再作成されます (注: この設定は、データベース構成の場合にのみ有効です)。

ACCESS *索引アクセス時* では、無効な索引は、その索引が最初にアクセスされた時点で再作成されます。

RESTART *データベース再始動時* では、無効な索引は、RESTART DATABASE コマンドが明示的または暗黙的に発行された時点で再作成されます。なお、RESTART DATABASE コマンドが暗黙的に発行されるのは、*autorestart* パラメーターが使用可能になっている場合です。

索引が無効になるのは、致命的なディスク問題が発生したときです。これがデータ自体に生じた場合は、データが失われる恐れがあります。しかし、この問題が発生したのが索引である場合は、その索引を再作成することで、索引はリカバリーできます。ただし、ユーザーがデータベースに接続されているときに、索引が再作成されると、次のような 2 つの問題が生じる可能性があります。

- 索引ファイルの再作成に伴って、予期しない応答時間が低下する場合があります。ユーザーが表にアクセスして、この特定の索引を使用する場合は、索引が再作成されている間、ユーザーは待機することになります。
- 索引の再作成後、予期しないロックが保留される場合があります。特に、その索引の再作成の原因となったユーザー・トランザクションが COMMIT や ROLLBACK をまったく実行していなかった場合は、その可能性があります。

推奨: このオプションに関しては、ユーザー数の多いサーバーであり、しかも再始動のタイミングが重要でない場合は、クラッシュ後にデータベースをオンラインに戻す処理の一環として、DATABASE RESTART 時に索引が再作成されるようにするのが、最善の選択です。

このパラメーターを「ACCESS」に設定すると、索引の再作成中は、データベース・マネージャーのパフォーマンスが低下する結果になります。ユーザーがその特定の索引または表にアクセスした場合は、索引が再作成されるまで、ユーザーは待機する必要があります。

このパラメーターが「RESTART」に設定されている場合は、データベースの再始動に要する時間は、索引の再作成のせいで長くなりますが、データベースがオンラインに戻った後は、通常の処理に影響が生じることはありません。

注: データベース・リカバリー時には、リカバリー中のデータベースに属するファイル・システム上にある、すべての SQL プロシージャー実行可能ファイルが除去されます。 *indexrec* が RESTART に設定されている場合は、すべての SQL プロシージャー実行可能ファイルがデータベース・カタログから取り出され、次回データベースに接続するときにファイル・システムに書き戻されます。

indexrec が RESTART に設定されていない場合、SQL 実行可能ファイルは、その SQL プロシージャーが最初に実行されるときにだけファイル・システムに取り出されます。

関連タスク:

- 「*アプリケーション開発ガイド アプリケーションの構築および実行*」の『DB2 8.2 より前に作成された SQL プロシージャーのバックアップとリストア』

関連資料:

- 475 ページの『autorestart - 自動再始動使用可能』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

num_db_backups - データベース・バックアップの数

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
伝搬クラス	トランザクション境界
デフォルト[範囲]	12 [1 — 32 768]

このパラメーターは、データベースのために保存するデータベース・バックアップの数を指定します。指定したバックアップ数に達すると、リカバリー履歴ファイルで古いバックアップが有効期限切れとマークされます。有効期限が切れたデータベース・バックアップに関連する表スペース・バックアップおよびロード・コピー・バックアップのリカバリー履歴ファイル項目も有効期限切れとマークされます。バックアップに有効期限切れのマークが付けられると、物理バックアップは、その保管場所 (たとえば、ディスク、テープ、TSM など) から除去できます。次のデータベース・バックアップが行われると、有効期限が切れた項目がリカバリー履歴ファイルから除去されます。

データベース・バックアップが履歴ファイルで有効期限切れとマークされると、DB2 Data Links Manager を介してリンクされた対応するファイル・バックアップが、そのアーカイブ・サーバーから除去されます。

rec_his_retentn 構成パラメーターは、*num_db_backups* の値と互換性のある値に設定してください。たとえば、*num_db_backup* が大きい値に設定されている場合、*rec_his_retentn* は、バックアップ数をサポートするのに十分な大きさの値が設定されていなければなりません。

関連資料:

- 484 ページの『rec_his_retentn - リカバリー履歴保存期間』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

rec_his_retentn - リカバリー履歴保存期間

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	366 [-1; 0 — 30 000]
単位	日数

このパラメーターは、バックアップの履歴情報を保存する日数を指定します。リカバリー履歴ファイルがバックアップ、リストア、ロードの追跡に必要でない場合、このパラメーターの値を小さく設定することができます。

このパラメーターの値が -1 の場合、リカバリー履歴ファイルはコマンドまたは API を明示的に使用することによってのみ、縮小することができます。値が -1 でない場合、リカバリー履歴ファイルは、フル・データベース・バックアップを実行するたびに縮小されます。

このパラメーターの値は *num_db_backups* パラメーターの値をオーバーライドしますが、*rec_his_retentn* と *num_db_backups* は、一緒に作用する必要があります。*num_db_backups* の値が大きい場合は、*rec_his_retentn* の値には、その数のバックアップをサポートできるだけの十分な大きさが必要です。

保存期間がどんなに短くても、PRUNE ユーティリティーに FORCE オプションを指定していなければ、ほとんどのフル・データベース・バックアップとそのリストアセットは必ず保存されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『PRUNE HISTORY/LOGFILE コマンド』
- 483 ページの『num_db_backups - データベース・バックアップの数』

trackmod - 変更されたページの追跡使用可能化

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲] No [Yes, No]

このパラメーターが「Yes」に設定されていると、データベース・マネージャーがデータベースの変更を追跡するので、データベース・ページのどのサブセットを増分バックアップによって調べ、バックアップ・イメージに組み込む必要があるかを検出できます。このパラメーターを「Yes」に設定した後は、増分バックアップを取ることができるベースラインを決めるために、全データベース・バックアップを取る必要があります。また、このパラメーターが使用可能になっている場合、または表スペースが作成されている場合は、その表スペースを含むバックアップを取る必要があります。このバックアップは、データベース・バックアップと表スペース・バックアップのいずれかになります。このバックアップの後では、増分バックアップに表スペースを含めることができます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

tsm_mgmtclass - Tivoli Storage Manager 管理クラス

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲] NULL [任意のストリング]

Tivoli Storage Manager 管理クラスは、バックアップするオブジェクトのバックアップ・バージョンを TSM がどのように管理するかを判断します。

デフォルトでは、DB2 指定の管理クラスはありません。

TSM バックアップを実行すると、データベース構成パラメーターで指定した管理クラスを使用する前に、 TSM はまず TSM クライアント・オプション・ファイル内にある INCLUDE-EXCLUDE リストで指定した管理クラスに、バックアップ・オブジェクトをバインドしようとします。一致するものが見付からない場合には、 TSM サーバーで指定されたデフォルトの TSM 管理クラスが使用されます。次いで TSM はデータベース構成パラメーターによって指定された管理クラスに、バックアップ・オブジェクトを再バインドします。

ですから、デフォルトの管理クラスとデータベース構成パラメーターによって指定された管理クラスには、バックアップ・コピー・グループが含まれていなければなりません。含まれていないと、バックアップ操作は失敗します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「*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*」の『Tivoli Storage Manager』

tsm_nodename - Tivoli Storage Manager ノード名

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

伝搬クラス ステートメント境界

デフォルト[範囲] NULL [任意のストリング]

このパラメーターは、Tivoli Storage Manager (TSM) 製品に関連するノード名のデフォルト設定をオーバーライドするために使用されます。ノード名は、別のノードから TSM にバックアップされたデータベースのリストアを可能にするために必要です。

デフォルトでは、バックアップを行ったノード上に TSM からデータベースをリストアすることのみ可能になります。 DB2 でバックアップが行われている (たとえば、BACKUP DATABASE コマンドを使用して) 最中に、*tsm_nodename* がオーバーライドされることは可能です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

tsm_owner - Tivoli Storage Manager 所有者名

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

伝搬クラス ステートメント境界

デフォルト[範囲] NULL [任意のストリング]

このパラメーターは、Tivoli Storage Manager (TSM) 製品に関連する所有者のデフォルト設定をオーバーライドするために使用されます。所有者名は、別のノードから TSM にバックアップされたデータベースをリストアできるようにする場合に必要です。DB2 を介した (たとえば BACKUP DATABASE コマンドによる) バックアップ時に、*tsm_owner* をオーバーライドすることが可能です。

注: 所有者名には、大文字と小文字の区別があります。

デフォルトでは、バックアップを行ったノード上に TSM からデータベースをリストアすることのみ可能になります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

tsm_password - Tivoli Storage Manager パスワード

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
伝搬クラス	ステートメント境界
デフォルト[範囲]	NULL [任意のストリング]

このパラメーターは、Tivoli Storage Manager (TSM) 製品に関連するパスワードのデフォルト設定をオーバーライドするために使用されます。パスワードは、別のノードから TSM にバックアップされたデータベースのリストアを可能にするために必要です。

注: *tsm_nodename* が DB2 での (たとえば BACKUP DATABASE コマンドによる) バックアップ時にオーバーライドされる場合、*tsm_password* も設定する必要があります。

デフォルトでは、バックアップを行ったノード上に TSM からデータベースをリストアすることのみ可能になります。 DB2 でバックアップが行われている最中に、*tsm_nodename* がオーバーライドされることは可能です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

分散作業単位リカバリー

以下のパラメーターは、分散作業単位 (DUOW) トランザクションのリカバリーに影響します。

- 488 ページの『resync_interval - トランザクション再同期インターバル』
- 488 ページの『spm_log_file_sz - 同期点マネージャー・ログ・ファイル・サイズ』
- 489 ページの『spm_log_path - 同期点マネージャー・ログ・ファイル・パス』
- 490 ページの『spm_max_resync - 同期点マネージャー再同期エージェント限界』

- 490 ページの『spm_name - 同期点マネージャー名』
- 491 ページの『tm_database - トランザクション・マネージャー・データベース名』

resync_interval - トランザクション再同期インターバル

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 180 [1 - 60 000]

単位 秒

このパラメーターは、トランザクション・マネージャー (TM)、リソース・マネージャー (RM)、または同期点マネージャー (SPM) が、TM、RM、または SPM で検出された未解決の未確定トランザクションのリカバリーを再試行する時間インターバルを秒数で指定します。このパラメーターは、分散作業単位 (DUOW) 環境でトランザクションを実行している場合にのみ使用できます。このパラメーターは、フェデレーテッド・データベース・システムのリカバリーにも適用されます。

推奨: 現在の環境内で、未確定トランザクションがデータベースに対して実行される他のトランザクションを妨害しなければ、このパラメーターの値を増加してもかまいません。 DB2 Connect ゲートウェイを使用して DRDA2 アプリケーション・サーバーにアクセスしている場合は、たとえローカル・データ・アクセスを妨害することがないとしても、アプリケーション・サーバーで未確定トランザクションがもたらす影響を考慮する必要があります。未確定トランザクションがない場合、パフォーマンスの低下は最小限で済みます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

spm_log_file_sz - 同期点マネージャー・ログ・ファイル・サイズ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 256 [4 — 1 000]

単位 ページ (4 KB)

このパラメーターは、同期点マネージャー (SPM) ログ・ファイル・サイズを 4 KB ページ単位で識別します。ログ・ファイルは、sqllib の下の spmlog サブディレクトリーに含まれ、最初に SPM が始動したときに作成されます。

推奨: 同期点マネージャー・ログ・ファイル・サイズにはパフォーマンスを維持できるだけの十分な大きさが必要ですが、スペースの浪費を防げる程度の大きさにとどめておく必要もあります。必要なサイズは、同期点マネージャーを使用するトランザクションの数、および COMMIT または ROLLBACK の発行頻度によって異なります。

SPM ログ・ファイルのサイズを変更するには、以下を行ってください。

1. LIST DRDA INDOUBT TRANSACTIONS コマンドを使用して、未確定トランザクションが存在しないことを判別します。
2. 存在しない場合は、データベース・マネージャーを停止します。
3. 新しい SPM ログ・ファイル・サイズでデータベース・マネージャー構成を更新します。
4. $HOME/sqllib ディレクトリーに移動し、 rm -fr spmlog を発行して、現在の SPM ログを削除します。(注 : これは AIX コマンドを表しています。その他のシステムでは、別の除去コマンドまたは削除コマンドが必要になります。)
5. データベース・マネージャーを始動します。指定されたサイズの新しい SPM ログがデータベース・マネージャーの始動時に作成されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

spm_log_path - 同期点マネージャー・ログ・ファイル・パス

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] sqllib/spmlog [有効なパスまたは装置]

このパラメーターは、同期点マネージャー (SPM) ログが書き込まれるディレクトリーを指定します。デフォルトでは、ログは sqllib/spmlog ディレクトリーに書き込まれます。このディレクトリーは、大量のトランザクションを持つ環境では、入出力障害の原因になる可能性があります。このパラメーターを使用して、SPM ログ・ファイルが現在の sqllib/spmlog ディレクトリーではなく高速ディスクに置かれるようにします。これにより、SPM エージェント間の並列処理が高くなります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

spm_max_resync - 同期点マネージャー再同期エージェント限界

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] 20 [10 — 256]

このパラメーターは、再同期操作を同時に実行できるエージェントの数を指定します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

spm_name - 同期点マネージャー名

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト TCP/IP ホスト名から派生

このパラメーターは、データベース・マネージャーに対して同期点マネージャー (SPM) インスタンスの名前を識別します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

tm_database - トランザクション・マネージャー・データベース名

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] 1ST_CONN [任意の有効なデータベース名]

このパラメーターは、それぞれの DB2 インスタンスごとにトランザクション・マネージャー (TM) の名前を指定します。 TM データベースは、次のいずれかです。

- ローカル DB2 Universal Database データベース
- ホストまたは AS/400 システムにはないリモート DB2 Universal Database データベース
- DB2 for OS/390 V5 データベース (TCP/IP を介してアクセスされ、同期点マネージャー (SPM) が使用されない場合)

TM データベースとは、ロガーおよびコーディネーターとして使用され、未確定トランザクションのリカバリーを行うために使用されるデータベースのことです。

このパラメーターを **1ST_CONN** に設定すると、TM データベースを、ユーザーが最初に接続する最初のデータベースにすることができます。

推奨: 管理および運用を単純化するために、多数のインスタンスに対して少数のデータベースを作成し、これらのデータベースを TM データベースとして排他的に使用することができます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

データベース管理

データベースに関する情報を記述したりデータベースの管理を制御するためのパラメーターがいくつかあります。次のグループに分けることができます。

- 『Query Enabler』
- 493 ページの『属性』
- 496 ページの『DB2 Data Links Manager』
- 500 ページの『状況』
- 503 ページの『コンパイラーの設定』
- 510 ページの『自動保守』

Query Enabler

次のパラメーターは、 Query Enabler の制御を記述しています。

- 『dyn_query_mgmt - 動的 SQL 照会管理』

dyn_query_mgmt - 動的 SQL 照会管理

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 (DISABLE) [1(ENABLE), 0 (DISABLE)]

このパラメーターは DB2 Query Patroller がインストールされている場合に有効です。このパラメーターを「ENABLE」に設定すると、Query Patroller は、サブミッター ID や実行の見積もりコスト (オプティマイザーが計算) など、照会の情報を収集します。こうした値は、照会を Query Patroller で、ユーザー・レベルおよびシステム・レベルのしきい値を基にして管理すべきかどうかを判断するのに使用します。

このパラメーターを「DISABLE」に設定すると、Query Patroller はサブミットされた照会に関する情報を収集せず、照会管理は行われません。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

属性

次のパラメーターは、データベースの一般的な情報を記述します。

- 『alt_collate - 代替照合シーケンス』
- 494 ページの『codepage - データベースのコード・ページ』
- 494 ページの『codeset - データベース用コード・セット』
- 494 ページの『collate_info - 照合情報』
- 495 ページの『country - データベース・テリトリー・コード』
- 495 ページの『database_level - データベース・リリース・レベル』
- 495 ページの『release - 構成ファイル・リリース・レベル』
- 496 ページの『territory - データベース・テリトリー』

alt_collate 以外のパラメーターは、情報提供専用です。

alt_collate - 代替照合シーケンス

構成タイプ データベース

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] Null [IDENTITY_16BIT]

このパラメーターでは、非ユニコード・データベース内のユニコード表に使用する照合シーケンスを指定します。このパラメーターを設定しないうちは、ユニコードの表とルーチンを非ユニコード・データベース内に作成できません。このパラメーターを一度設定すると、変更もリセットもできません。

このパラメーターはユニコード・データベースには設定できません。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

codepage - データベースのコード・ページ

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターは、データベースを作成するために使用されたコード・ページを示します。 *codepage* パラメーターは、*codeset* パラメーターに基づいて派生されます。

関連資料:

- 494 ページの『codeset - データベース用コード・セット』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

codeset - データベース用コード・セット

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターは、データベースを作成するために使用されたコード・セットを示します。コード・セットは、*codepage* パラメーターの値を決定する場合に、データベース・マネージャーで使用されます。

関連資料:

- 494 ページの『codepage - データベースのコード・ページ』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

collate_info - 照合情報

このパラメーターを表示できるのは、db2CfgGet API を使用した場合だけです。コマンド行プロセッサーやコントロール・センターでは表示**できません**。

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターは、260 バイトのデータベース照合情報を提供します。最初の 256 バイトでデータベース照合シーケンスを指定するのに対して、バイト「n」には、データベースのコード・ページで基本 10 進表記が「n」になっている、コード・ポイントのソートに対する重みづけが入ります。

最後の 4 バイトには、照合シーケンスのタイプについての内部情報が入ります。これは、データベースのプラットフォームに適用できる整数として扱うことができます。次の 3 つの値があります。

- **0** - シーケンスに非ユニークの重みが含まれる

- **1** - シーケンスにユニークの重みすべてが含まれる
- **2** - シーケンスは ID シーケンスで、ストリングがバイトごとに比較される

この内部タイプ情報を使用する場合は、別のプラットフォームにあるデータベースに関する情報を検索するときに、バイト反転を考慮する必要があります。

照合シーケンスは、データベース作成時に指定できます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

country - データベース・テリトリー・コード

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターは、データベースを作成するために使用されたテリトリー・コードを示します。

関連資料:

- 496 ページの『territory - データベース・テリトリー』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

database_level - データベース・リリース・レベル

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターは、データベースを使用できるデータベース・マネージャーのリリース・レベルを示します。移行が不完全であったり、失敗したりした場合は、このパラメーターに反映されるのは移行されなかったデータベースのリリース・レベルであり、 *release* パラメーター (データベース構成ファイルのリリース・レベル) とは異なる可能性があります。それ以外の場合は、*database_level* の値は、*release* パラメーターの値と同一になります。

関連資料:

- 495 ページの『release - 構成ファイル・リリース・レベル』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

release - 構成ファイル・リリース・レベル

構成タイプ データベース・マネージャー、データベース

適用

- ローカルとリモート・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 通知

このパラメーターは、構成ファイルのリリース・レベルを指定します。

関連資料:

- 495 ページの『database_level - データベース・リリース・レベル』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』

territory - データベース・テリトリー

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターは、データベースを作成するために使用されたテリトリーを示します。 *territory* は、テリトリー・コード (*territory*) パラメーターの値を判別する場合に、データベース・マネージャーで使用されます。

関連資料:

- 495 ページの『country - データベース・テリトリー・コード』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

DB2 Data Links Manager

次のパラメーターは、 DB2 Data Links Manager と関連があります。

- 『datalinks - データ・リンク・サポート使用可能』
- 497 ページの『dl_expint - データ・リンク・アクセス・トークン有効期限インターバル』
- 497 ページの『dl_num_copies - データ・リンクのコピー数』
- 498 ページの『dl_time_drop - ドロップ後のデータ・リンク時間』
- 498 ページの『dl_token - データ・リンク・トークン・アルゴリズム』
- 499 ページの『dl_upper - 大文字のデータ・リンク・トークン』
- 499 ページの『dl_wt_iexpint - データ・リンク書き込みトークン初期有効期限インターバル』

datalinks - データ・リンク・サポート使用可能

構成タイプ データベース・マネージャー

パラメーター・タイプ 構成可能

デフォルト[範囲] NO [YES; NO]

このパラメーターはデータ・リンク・サポートが使用可能かどうかを指定します。「YES」の値は、ネイティブのファイル・システム (たとえば AIX の JFS) に保管

されたファイルにリンクしている Data Links Manager のデータ・リンク・サポートが使用可能であることを指定します。「NO」の値は、データ・リンク・サポートが使用可能でないことを指定します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

dl_expint - データ・リンク・アクセス・トークン有効期限インターバル

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
伝搬クラス	トランザクション境界
デフォルト[範囲]	60 [1 — 31 536 000]
単位	秒

このパラメーターによって、生成されたファイル・アクセス・コントロール・トークンが有効な時間のインターバル (秒) が指定されます。トークンが有効になる秒は、トークンが生成された時間から開始されます。データ・リンク・ファイル・システム・フィルターによって、この有効期限に対するトークンの妥当性がチェックされます。

このパラメーターは、「READ PERMISSION DB」を指定する DATALINK 列に適用されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dl_num_copies - データ・リンクのコピー数

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
伝搬クラス	即時
デフォルト[範囲]	0 [0 - 15]

このパラメーターは、データベースへのファイルのリンク時にアーカイブ・サーバー (TSM サーバーなど) に作成される、ファイルの追加コピーの数を指定します。

このパラメーターのデフォルト値はゼロ (0) です。

このパラメーターは、「Recovery=Yes」を指定する DATALINK 列に適用されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dl_time_drop - ドロップ後のデータ・リンク時間

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
伝搬クラス	即時
デフォルト[範囲]	1 [0 — 365]
単位	日数

このパラメーターは、DROP DATABASE の発行後、ファイルがアーカイブ・サーバー (たとえば、TSM サーバーなど) に保存される時間のインターバル (日数) を指定します。

このパラメーターのデフォルト値は 1 日です。値がゼロ (0) の場合は、DROP コマンドが発行されると、即時にファイルがアーカイブ・サーバーからドロップされることを意味します (ON UNLINK DELETE パラメーターが DATALINK 列に指定されない限り、実ファイルは削除されません)。

このパラメーターは、「Recovery=Yes」を指定する DATALINK 列に適用されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dl_token - データ・リンク・トークン・アルゴリズム

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
伝搬クラス	トランザクション境界
デフォルト[範囲]	MAC0 [MAC0; MAC1]

このパラメーターは、DATALINK ファイル・アクセス・コントロール・トークンの生成で使用されるアルゴリズムを指定します。 MAC1 (メッセージ認証コード) の値は MAC0 よりもさらに安定したメッセージ認証コードを生成しますが、パフォーマンス上のオーバーヘッドも増えます。

このパラメーターは、「READ PERMISSION DB」を指定する DATALINK 列に適用されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dl_upper - 大文字のデータ・リンク・トークン

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

伝搬クラス トランザクション境界

デフォルト[範囲] NO [YES; NO]

パラメーターは、ファイル・アクセス・コントロール・トークンが大文字を使用するかどうかを指示します。「YES」の値は、アクセス・コントロール・トークンのすべての文字が大文字であることを指定します。「NO」の値は、トークンに大文字と小文字の両方を含めることが可能であることを指定します。

このパラメーターは、「READ PERMISSION DB」を指定する DATALINK 列に適用されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dl_wt_iexpint - データ・リンク書き込みトークン初期有効期限インターバル

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

伝搬クラス トランザクション境界

デフォルト[範囲] 60 [1 - 31 536 000]

単位 秒

このパラメーターは、DATALINK 列から生成される書き込みトークンの初期有効期限インターバルを指定します。初期有効期限インターバルとは、トークンが生成された時点とそれがファイルのオープンに初めて使用された時点の間の期間のことです。

このパラメーターは、WRITE PERMISSION ADMIN を指定して定義された DATALINK 列の場合にのみ適用されます。

推奨: この値には、書き込みトークンが取り出された時点からファイルのオープンに使用される時点までの時間に対応できる、十分な大きさが必要です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

状況

次のパラメーターは、データベースの状態を記述しています。

- 『backup_pending - バックアップ・ペンディング標識』
- 501 ページの『database_consistent - データベースの整合性』
- 501 ページの『log_retain_status - ログ保存状況標識』
- 501 ページの『multipage_alloc - マルチページ・ファイル割り振り使用可能』
- 502 ページの『restore_pending - リストア・ペンディング』
- 502 ページの『rollfwd_pending - ロールフォワード・ペンディング標識』
- 502 ページの『user_exit_status - ユーザー出口状況標識』

backup_pending - バックアップ・ペンディング標識

構成タイプ データベース

パラメーター・タイプ 通知

オンに設定されている場合、このパラメーターは、データベースにアクセスする前にデータベース全体のバックアップを行う必要があることを示します。このパラメーターはデータベースをリカバリー不能からリカバリー可能にするために、データベース構成を変更した場合にのみオンになります。 (つまり、最初に *logretain* および *userexit* パラメーターの両方が NO に設定されてから、これらのパラメーターのいずれかあるいは両方が YES に設定され、データベース構成への更新が受け入れられた場合です)。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

database_consistent - データベースの整合性

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターは、データベースが整合状態にあるかどうかを示します。

YES では、すべてのトランザクションがコミットまたはロールバックされているので、データは整合性がとれていることを示します。データベースの整合性がとれているときに、システムが「クラッシュ」した場合は、データベースを使用可能にするために、特殊なアクションを行う必要はありません。

NO では、トランザクションがペンディング中であるか、データベース上でペンディング中のタスクが他にあって、この時点でデータに整合性がないことを示します。データベースに整合性がないときに、システムが「クラッシュ」した場合は、RESTART DATABASE コマンドを使用してデータベースを再始動し、データベースを使用可能にする必要があります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

log_retain_status - ログ保存状況標識

構成タイプ データベース

パラメーター・タイプ 通知

設定された場合、このパラメーターは、ログ・ファイルがロールフォワード・リカバリーで使用するために保存されていることを示します。

このパラメーターが設定されるのは、*logretain* パラメーターの設定が「Recovery」に等しいときです。

関連資料:

- 468 ページの『logretain - ログ保存使用可能』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

multipage_alloc - マルチページ・ファイル割り振り使用可能

構成タイプ データベース

パラメーター・タイプ 通知

マルチページ・ファイル割り振りは、挿入パフォーマンスを上げる場合に使用します。これは、SMS 表スペースにしか適用されません。使用可能にすると、すべての SMS 表スペースに影響します。つまり、個々の SMS 表スペースごとに選択することはできません。

このパラメーターは、デフォルトでは「Yes」です。つまり、マルチページ・ファイル割り振りが使用可能です。

データベース作成後にこのパラメーターを「No」に設定することはできません。マルチページ・ファイル割り振りを一度使用可能にすると、使用不可にはできません。マルチページ・ファイル割り振りを望まない場合には、データベース作成前にDB2_NO_MPFA_FOR_NEW_DB DB2 レジストリー変数を適切に設定する必要があります。 **db2empfa** ツールを使用すると、マルチページ・ファイル割り振りが現在使用不可になっているデータベースに対して、マルチページ・ファイル割り振りが使用できます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 591 ページの『パフォーマンス変数』

restore_pending - リストア・ペンディング

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターは、RESTORE PENDING 状況がデータベース内に存在するかどうかを示します。

関連資料:

- 473 ページの『userexit - ユーザー出口使用可能』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

rollfwd_pending - ロールフォワード・ペンディング標識

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターでは、次の状態のいずれか 1 つを指定できます。

- **DATABASE**。ロールフォワード・リカバリー手順がこのデータベースに必要であることを示します。
- **TABLESPACE**。1 つ以上の表スペースがロールフォワードする必要があることを示します。
- **NO**。データベースが使用可能であり、ロールフォワード・リカバリーは必要ないことを意味します。

リカバリー (ROLLFORWARD DATABASE の使用による) が完了しないうちは、データベースにも表スペースにもアクセスできません。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

user_exit_status - ユーザー出口状況標識

構成タイプ データベース

パラメーター・タイプ 通知

このパラメーターが Yes に設定されている場合は、データベース・マネージャーがロールフォワード・リカバリーのために使用可能になっていて、ユーザー出口プログラムは、データベース・マネージャーによって呼び出されると、ログ・ファイルのアーカイブおよび検索に使用されることを示します。

関連資料:

- 473 ページの『userexit - ユーザー出口使用可能』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』

コンパイラーの設定

次のパラメーターは、コンパイラーに影響する情報を提供します。

- 『dft_degree - デフォルト多重度』
- 504 ページの『dft_mttb_types - 最適化用デフォルト保守表タイプ』
- 504 ページの『dft_queryopt - デフォルト照会最適化クラス』
- 505 ページの『dft_refresh_age - デフォルト・リフレッシュ経過時間』
- 505 ページの『dft_sqlmathwarn - 演算例外時継続』
- 507 ページの『num_freqvalues - 保存される頻度値の数』
- 508 ページの『num_quantiles - 列の変位値の数』

dft_degree - デフォルト多重度

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
伝搬クラス	即時
デフォルト[範囲]	1 [-1, 1 - 32 767]

このパラメーターは、CURRENT DEGREE 特殊レジスターおよび DEGREE BIND オプションのデフォルト値を指定します。

デフォルト値は 1 です。

値が 1 では、パーティション内並列処理がないことを意味します。値が -1 では、オプティマイザーがプロセッサーの数と照会のタイプに基づいて、パーティション内並列処理の多重度を決定することを示します。

SQL ステートメントの並列処理の多重度は、CURRENT DEGREE 特殊レジスターまたは DEGREE BIND オプションを使用して、ステートメントのコンパイル時に指定されます。アクティブ・アプリケーションのパーティション内並列処理の最大実行時の多重度は、SET RUNTIME DEGREE コマンドを使用して指定します。「照会の最大並列処理多重度」 (*max_querydegree*) 構成パラメーターでは、すべての SQL 照会のパーティション内の照会最大並列処理多重度を指定します。

実行時に使用される実際の多重度は、次のうちで最も低い値になります。

- *max_querydegree* 構成パラメーター
- アプリケーション実行時の多重度
- SQL ステートメント・コンパイル時の多重度

関連資料:

- 525 ページの『max_querydegree - 照会の最大並列処理多重度』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dft_mttb_types - 最適化用デフォルト保守表タイプ

構成タイプ	データベース
パラメーター・タイプ	構成可能
デフォルト[範囲]	SYSTEM [ALL、NONE、FEDERATED_TOOL、SYSTEM、USER、または値のリスト]

このパラメーターは、CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターのデフォルト値を指定します。このレジスターの値は、照会の最適化中に使用される REFRESH DEFERRED マテリアライズ照会表のタイプを決定します。

値のリストをコンマで区切って指定できます。たとえば、'USER,FEDERATED_TOOL' などです。 ALL または NONE を他の値を一緒にリストすることはできません。同じ値を複数回指定することもできません。

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスター』

dft_queryopt - デフォルト照会最適化クラス

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
伝搬クラス	即時
デフォルト[範囲]	5 [0 — 9]
単位	照会最適化クラス (以下を参照)

照会最適化クラスは、SQL 照会のコンパイルに別々の段階の最適化を使用するように、オプティマイザーに指示するために使用します。このパラメーターでは、SET CURRENT QUERY OPTIMIZATION ステートメントも BIND コマンドの QUERYOPT オプションも使用されていないときに使用される、デフォルトの照会最適化クラスを設定することによって、さらに柔軟性が得られます。

照会最適化クラスは、現在のところ次のように定義されています。

- 0 - 最小照会最適化。
- 1 - DB2 バージョン 1 とほぼ同等。
- 2 - 低レベルの最適化。

- 3 - 中レベルの照会最適化。
- 5 - アクセス・プランの選択に費やされる労力を制限する、経験的による重要な照会最適化。これはデフォルトです。
- 7 - かなりの照会最適化。
- 9 - 最大照会最適化。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT QUERY OPTIMIZATION ステートメント』
- 「コマンド・リファレンス」の『BIND コマンド』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『LIST DRDA INDOUBT TRANSACTIONS コマンド』

dft_refresh_age - デフォルト・リフレッシュ経過時間

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲] 0 [0, 99999999999999 (ANY)]

このパラメーターでは、CURRENT REFRESH AGE 特殊レジスターが指定されていない場合は、REFRESH AGE にデフォルト値が使用されます。このパラメーターは DECIMAL(20,6) というデータ・タイプのタイム・スタンプ期間値を指定します。この時刻期間では、REFRESH TABLE ステートメントが特定の REFRESH DEFERRED マテリアライズ照会表に対して処理されてから、そのサマリー表を使用して照会の処理を最適化できる最大期間を表します。CURRENT REFRESH AGE の値が 99999999999999 (ANY) で、QUERY OPTIMIZATION クラスの値が 2、または 5 以上である場合は、REFRESH DEFERRED マテリアライズ照会表が、動的 SQL 照会の処理を最適化すると見なされます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

dft_sqlmathwarn - 演算例外時継続

構成タイプ データベース

パラメーター・タイプ 構成可能

デフォルト[範囲] No [No, Yes]

このパラメーターは、演算エラーおよび検索変換エラーを SQL ステートメントのコンパイル時のエラーまたは警告として処理することを決定する、デフォルト値を設定します。静的 SQL ステートメントの場合は、このパラメーターの値は、バインド時にパッケージに関連付けられます。動的 SQL DML ステートメントの場合は、このパラメーターの値は、ステートメントの準備時に使用されます。

重要: データベースの *dft_sqlmathwarn* 値を変更した場合は、算術式が組み込まれているチェック制約、トリガー、およびビューの振る舞いが変更されることがあります。この結果、データベースのデータ保全性に影響を及ぼす場合があります。したがって、データベースの *dft_sqlmathwarn* の設定を変更するのは、新しい演算例外処理の振る舞いがチェック制約、トリガー、およびビューにどのように影響するかを入念に評価した後のみに限る必要があります。一度変更すると、その後の変更でも同じく入念な評価が必要です。

例として、割り算の算術演算が組み込まれている、次のチェック制約を考察してみましょう。

```
A/B > 0
```

dft_sqlmathwarn が「No」であるとき、B=0 で INSERT を試みると、ゼロによる割り算は演算エラーとして処理されます。DB2 が制約をチェックできないため、挿入操作は失敗します。 *dft_sqlmathwarn* が「Yes」に変更された場合は、ゼロによる割り算は、演算警告として処理され、NULL という結果になります。NULL という結果では、「>」述部が UNKNOWN に評価され、挿入操作は正常に行われます。*dft_sqlmathwarn* が変更されて元の「No」に戻った場合は、ゼロによる割り算のために DB2 が制約を評価できないため、同じ行を挿入する試みは失敗します。*dft_sqlmathwarn* が「Yes」であったときに B=0 で挿入された行は、表内にとどまり、選択できます。行に対する更新は、制約を評価させる更新の場合は失敗するが、制約の再評価を必要としない更新の場合は、正常に行われます。

dft_sqlmathwarn を「No」から「Yes」に変更する場合は、算術式から NULL を明示的に処理するために、あらかじめ制約の書き直しを考慮しておく必要があります。たとえば、

```
( A/B > 0 ) AND ( CASE
                    WHEN A IS NULL THEN 1
                    WHEN B IS NULL THEN 1
                    WHEN A/B IS NULL THEN 0
                    ELSE 1
                    END
                  = 1 )
```

が使用できるのは、A と B が両方とも NULL 可能な場合です。そして、A または B が NULL 可能でない場合は、対応する IS NULL WHEN 文節は除去できます。

dft_sqlmathwarn を「Yes」から「No」に変更するときは、まずその前に、たとえば、次のような述部を使用して、不整合になる可能性のあるデータがないかどうかチェックしておく必要があります。

```
WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

不整合の行の切り分けができたら、*dft_sqlmathwarn* を変更する前に、適切なアクションを行って不整合を訂正しておく必要があります。また、変更後の算術式を使用

して、制約を手動で再チェックすることもできます。これを実行するには、まず最初に、影響を受けた表を (SET CONSTRAINTS ステートメントの OFF 文節によって) チェック・ペンディング状態に置き、次に表が (SET CONSTRAINTS ステートメントの IMMEDIATE CHECKED 文節によって) チェックされるよう要求します。不整合データが演算エラーによって示されるので、制約は評価されません。

推奨: 演算例外が組み込まれている照会の処理が特に必要でない限り、デフォルト設定の No を使用してください。その上で、値 Yes を使用します。この状態が生じる可能性があるのは、他のデータベース・マネージャーで、演算例外が発生するかどうかに関係なく結果が得られる、SQL ステートメントを処理している場合です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

num_freqvalues - 保存される頻度値の数

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
伝搬クラス	即時
デフォルト[範囲]	10 [0 — 32 767]
単位	カウンター

このパラメーターを使用すると、 WITH DISTRIBUTION オプションが RUNSTATS コマンドで指定されたときに収集される、「最も頻度の高い値」の数を指定できます。このパラメーターの値を大きくすると、統計の収集時に使用される統計ヒープの量 (*stat_heap_sz*) が増大します。

「最も頻度の高い値」統計は、オプティマイザーに列内でのデータ値の分散を理解させるのに役立ちます。値を大きくすれば、SQL オプティマイザーで使用可能な情報が増える結果になりますが、追加のカタログ・スペースが必要になります。 0 が指定されていると、たとえ分散統計の収集を要求しても、高頻度値統計が保存されることはありません。

また、表レベルまたは列レベルでの RUNSTATS コマンドの一部として、保存される高頻度値の数を指定することもできます。何も指定されなかった場合は、*num_freqvalues* 構成パラメーター値が使用されます。

このパラメーターを更新すると、一様に分布していないデータについて、オプティマイザーが一部の述部 (=、<、>、IS NULL、IS NOT NULL) の選択性をより正確に見積もることができます。選択性計算の正確性が増せば、より効率的なアクセス・プランが選択できる可能性があります。

このパラメーターの値を変更した場合は、その後で次のことを行う必要があります。

- すべてのユーザーがデータベースから切断され、ユーザーがデータベースに再接続された後で、RUNSTATS コマンドを実行します。
- 静的 SQL が含まれているパッケージがあれば、すべて再バインドします。

RUNSTATS コマンドでは、NUM_REQVALUES オプションを使用することによって、保存される高頻度値の数を指定できます。保存される高頻度値の数を RUNSTATS コマンドによって変更する方が、 *num_freqvalues* データベース構成パラメーターを使用して変更を行うよりも簡単です。

RUNSTATS を使用すると、表レベルと列レベルの両方で収集される高頻度値の数を制限できます。したがって、カタログが活用できなかった列に関する分散統計を削減し、しかもなお、重要な列に関する情報は使用して、カタログ内で占有されているスペースについて最適化ができます。

推奨: このパラメーターを更新するためには、一般的に選択述部がある (最も重要な表内の) 最も重要な列における非均等度を決定する必要があります。これには、列内におけるそれぞれの値の出現回数の順序付きランキングを提供する、SQL SELECT ステートメントを使用することができます。均等に分布する値をもったもの、ユニークなもの、または、LONG や LOB 列を考慮する必要はありません。このパラメーターの妥当な実用値は、10 ～ 100 の範囲にあります。

高頻度値統計を収集する処理には、かなりの量の CPU およびメモリー (*stat_heap_sz*) リソースが必要です。

関連資料:

- 508 ページの『num_quantiles - 列の変位値の数』
- 411 ページの『stat_heap_sz - 統計ヒープ・サイズ』
- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

num_quantiles - 列の変位値の数

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
伝搬クラス	即時
デフォルト[範囲]	20 [0 - 32 767]
単位	カウンター

このパラメーターでは、WITH DISTRIBUTION オプションが RUNSTATS コマンドで指定されたときに収集される変位値の数をコントロールします。このパラメーターの値を大きくすると、統計の収集時に使用される統計ヒープの量 (*stat_heap_sz*) が増大します。

「変位値」統計は、オプティマイザーに列内でのデータ値の分散を理解させるのに役立ちます。値を大きくすれば、SQL オプティマイザーで使用可能な情報が増える結果になりますが、追加のカタログ・スペースが必要になります。 0 または 1 が指定されていると、たとえ分散統計の収集を要求しても、変位値統計が保存されることはありません。

また、表レベルまたは列レベルでの RUNSTATS コマンドの一部として、収集される変位値の数を指定することもできます。何も指定されなかった場合は、*num_quantiles* 構成パラメーター値が使用されます。

このパラメーターを更新すると、一様に分布していないデータについて、範囲述部の選択性をより正確に見積もることができます。オプティマイザーの決定の中でも特に、この情報は、索引スキャンと表スキャンのどちらが選択されるかについて、強い影響を及ぼします (頻繁に発生する範囲の値にアクセスするには、表スキャンを使用する方が効率的であり、発生頻度の低い範囲の値の場合は、索引スキャンを使用する方が効率的です)。

このパラメーターの値を変更した場合は、その後で次のことを行う必要があります。

- すべてのユーザーがデータベースから切断され、ユーザーがデータベースに再接続された後で、RUNSTATS コマンドを実行します。
- 静的 SQL が含まれているパッケージがあれば、すべて再バインドします。

RUNSTATS コマンドでは、NUM_QUANTILES オプションを使用することによって、収集される変位値の数を指定できます。収集される変位値の数を RUNSTATS コマンドによって変更する方が、*num_quantiles* データベース構成パラメーターを使用して変更を行うよりも簡単です。

RUNSTATS を使用すると、表レベルと列レベルの両方で収集される変位値の数を制限できます。したがって、カタログが活用できなかった列に関する分散統計を削減し、しかもなお、重要な列に関する情報は使用して、カタログ内で占有されているスペースについて最適化ができます。

推奨: このパラメーターのこのデフォルト値では、片側述部 (>、>=、<、または <=) の場合は、最大で約 2.5% の見積エラー、BETWEEN 述部の場合は、最大で 5% のエラーです。変位値の数を概算する簡単な方法は、次のとおりです。

- 範囲照会の行数の見積もりで許容できる最大エラー P をパーセント単位で求めます。
- 変位値の数は、述部のほとんどが BETWEEN 述部の場合は、およそ 100/P で、述部のほとんどがそれ以外のタイプの範囲述部 (<、<=、>、または >=) の場合は、50/P であることが必要です。

たとえば、変位値の数が 25 であれば、最大見積エラーは、BETWEEN 述部の場合は 4%、″>″ 述部の場合は、2% という結果になる必要があります。このパラメーターの妥当な実用値は、10 〜 50 の範囲にあります。

関連資料:

- 507 ページの『num_freqvalues - 保存される頻度値の数』
- 411 ページの『stat_heap_sz - 統計ヒープ・サイズ』

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

自動保守

次のパラメーターを使用すると、幾つかの DB2 ユーティリティーの自動保守アクティビティーを制御することができます。

- 『autonomic_switches - 自動保守スイッチ』

autonomic_switches - 自動保守スイッチ

構成タイプ データベース

適用

- ローカルとリモート・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 すべてのスイッチがオフ

このパラメーターを使用して、パラメーターの各ビットで内部的に表されるスイッチを設定できます。これらのスイッチは、次のパラメーターを設定することによって、それぞれ別個に独立して更新できます。

auto_maint このパラメーターは、他のすべての自動保守データベース構成パラメーター (*auto_db_backup*、*auto_tbl_maint*、 *auto_runstats*、*auto_stats_prof*、*auto_prof_upd*、および *auto_reorg*) の親パラメーターです。このパラメーターを使用不可にすると、データベース構成ファイルに記録されている子パラメーターの設定を変更しなくても、すべての子パラメーターが使用不可になります。この親パラメーターを使用可能にすると、記録されている子パラメーターの値が有効になります。このようにして、自動保守を一括で使用不可または使用不能にすることができます。

auto_db_backup この自動保守パラメーターは、データベースの自動バックアップ操作を使用可能または使用不可にします。自動化された動作を指定するために、バックアップ・ポリシー (ルールまたはガイドラインの定義

	済みセット) を使用できます。バックアップ・ポリシーの目的は、必ず定期的にデータベースがバックアップされるようにすることです。 DB2 ヘルス・モニターを実行しておくことにより、データベースのバックアップ・ポリシーは自動的に作成されます。このパラメーターを使用可能にするには、このパラメーターをオンに設定し、親パラメーターも使用可能にすることが必要です。
auto_tbl_maint	このパラメーターは、すべての表保守パラメーター (*auto_runstats*、*auto_stats_prof*、 *auto_prof_upd*、and *auto_reorg*) の親パラメーターです。このパラメーターを使用不可にすると、データベース構成ファイルに記録されている子パラメーターの設定を変更しなくても、すべての子パラメーターが使用不可になります。この親パラメーターを使用可能にすると、記録されている子パラメーターの値が有効になります。このようにして、表保守を一括で使用不可または使用不能にすることができます。
auto_runstats	この自動表保守パラメーターは、データベースの自動表 RUNSTATS 操作を使用可能または使用不可にします。自動化された動作を指定するために、RUNSTATS ポリシー (ルールまたはガイドラインの定義済みセット) を使用できます。 RUNSTATS ユーティリティーによって収集された統計は、物理データにアクセスするための最も効率的なプランを判別するために、オプティマイザーによって使用されます。このパラメーターを使用可能にするには、このパラメーターをオンに設定し、親パラメーターも使用可能にすることが必要です。
auto_stats_prof	この自動表保守パラメーターを使用可能にすると、統計プロファイル生成がオンになります。統計プロファイル生成は、複数の表に対する複雑な照会、多数の述部、結合、およびグループ化操作を含むワークロードを持つアプリケーションを改善するために設計されたものです。このパラメーターを使用可能にするには、このパラメーターをオンに設定し、親パラメーターも使用可能にすることが必要です。
auto_prof_upd	この自動表保守パラメーター (*auto_stats_prof* の子パラメーター) を使用可能にすると、推奨値にしたがった RUNSTATS プロファイルの更新が指定されます。このパラメーターを使用不可にすると、推奨値は opt_feedback_ranking 表に保管されます。RUNSTATS プロファイルを手動で更新するときに、この推奨値を調べることができます。このパラメーターを使用可能にするには、このパラメーターをオンに設定し、親パラメーターも使用可能にすることが必要です。

auto_reorg

この自動表保守パラメーターは、データベースの表および索引の自動再編成を使用可能または使用不可にします。自動化された動作を指定するために、再編成ポリシー (ルールまたはガイドラインの定義済みセット) を使用できます。このパラメーターを使用可能にするには、このパラメーターをオンに設定し、親パラメーターも使用可能にすることが必要です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

通信

次のパラメーター・グループは、クライアント/サーバー環境での DB2 の使用状況について記述しています。

- 『通信プロトコルの設定』
- 515 ページの『DB2 ディスカバリー機能』

通信プロトコルの設定

次のパラメーターは、データベース・クライアントおよびデータベース・サーバーを構成する場合に使用します。

- 『nname - NetBIOS ワークステーション名』
- 513 ページの『svcename - TCP/IP サービス名』
- 514 ページの『tpname - APPC トランザクション・プログラム名』

nname - NetBIOS ワークステーション名

構成タイプ

データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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

構成可能

デフォルト

Null

このパラメーターを使用すると、NetBIOS LAN 環境内のワークステーション上のデータベース・インスタンスに、ユニークな名前を割り当てることができます。この *nname* は、ワークステーションの NetBIOS に登録される実際の NetBIOS 名の基本になります。

NetBIOS プロトコルは、これらの NetBIOS 名を使用して接続を確立するため、*nname* パラメーターをクライアントとサーバーの両方に設定する必要があります。

クライアント・アプリケーションは、アクセスするデータベースを含むサーバーの *nname* を認識している必要があります。サーバーの *nname* は、CATALOG NETBIOS NODE コマンドを使用して、クライアントのノード・ディレクトリーに「server-nname」パラメーターとしてカタログしなければなりません。

サーバー・ノードの *nname* を新しい名前に変更する場合は、そのサーバーのデータベースをアクセスするすべてのクライアントが、サーバーのこの新しい名前をカタログする必要があります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『CATALOG NETBIOS NODE コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

svcename - TCP/IP サービス名

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト Null

このパラメーターには、データベース・サーバーがリモート・クライアント・ノードからの通信を待つために使用する、 TCP/IP ポートの名前が含まれています。 この名前は、データベース・マネージャーが使用するために予約した 2 つの連続するポートの内の最初のポートに指定する必要があり、 2 番目のポートは、下位レベル・クライアントからの割り込み要求を処理するために使用されます。

TCP/IP を使用して、データベース・クライアントからの接続要求を受け入れるためには、データベース・サーバーがそのサーバーに指定されたポートで待機中でなければなりません。データベース・サーバーのシステム管理者は、ポートを予約して

(番号 *n*)、サーバーのサービス・ファイルにその関連する TCP/IP サービス名を定義する必要があります。データベース・サーバーが、下位レベル・クライアントからの要求をサポートする必要がある場合は、2 番目のポート (番号 *n+1*、割り込み要求用) をサーバーのサービス・ファイルに定義する必要があります。

データベース・サーバー・ポート (番号 *n*) とその TCP/IP サービス名は、データベース・クライアントのサービス・ファイルに定義する必要があります。下位レベル・クライアントの割り込みポート (*n+1*) も、クライアントのサービス・ファイルに定義する必要があります。

UNIX ベースのシステムでは、サービス・ファイルは `/etc/services` にあります。

svcename パラメーターは、データベース・サーバーの始動時に、着信接続要求を listen するポートを判別できるように、主接続ポートに関連したサービス名に設定する必要があります。下位レベル・クライアントをサポートまたは使用している場合、割り込みポートのサービス名は、構成ファイルに保管されません。割り込みポート番号は、主接続ポート番号に基づいて派生できます (*割り込みポート番号 = 主接続ポート番号 + 1*)。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

tpname - APPC トランザクション・プログラム名

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト Null

このパラメーターは、APPC 通信プロトコルの使用中に、データベース・クライアントがデータベース・サーバーに対して割り振り要求を出すときに使用する必要がある、リモート・トランザクション・プログラムの名前を定義します。このパラメーターは、データベース・サーバーの構成ファイルに設定する必要があります。

このパラメーターは、SNA トランザクション・プログラム定義に構成されているトランザクション・プログラムと同じ名前にしてください。

推奨: この名前に使用できる文字は、以下のとおりです。

- 英字 (A から Z; または a から z)
- 数字 (0 から 9)
- ドル記号 ($)、番号記号 (#)、アットマーク (@)、およびピリオド (.)

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

DB2 ディスカバリー機能

DB2 ディスカバリー機能を設定するために、以下のパラメーターを使用できます。

- 『discover - ディスカバリー・モード』
- 516 ページの『discover_db - データベースのディスカバリー』
- 516 ページの『discover_inst - サーバー・インスタンスのディスカバリー』

discover - ディスカバリー・モード

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] SEARCH [DISABLE, KNOWN, SEARCH]

クライアント側の観点からは、次のいずれかが行われます。

- *discover* = SEARCH の場合、クライアントは、ネットワーク上の DB2 サーバー・システムを見つけるための SEARCH ディスカバリー要求を発行できます。SEARCH ディスカバリーは、KNOWN ディスカバリーが提供する関数のスーパーセットを提供します。 *discover* = SEARCH の場合、クライアントは SEARCH ディスカバリー要求および KNOWN ディスカバリー要求の両方を発行することができます。
- *discover* = KNOWN の場合、クライアントは KNOWN ディスカバリー要求のみを発行することができます。特定のシステム上の Administration Server に関する接続情報を指定すると、DB2 システム上のすべてのインスタンスおよびデータベース情報がクライアントに返されます。

- *discover* = DISABLE の場合、クライアントではディスカバリーが使用不可になります。

デフォルトのディスカバリー・モードは SEARCH です。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 579 ページの『通信変数』

discover_db - データベースのディスカバリー

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

伝搬クラス 即時

デフォルト[範囲] Enable [Disable, Enable]

このパラメーターは、サーバーでディスカバリー要求が受け取られたときに、データベースについての情報がクライアントに戻されないようにするために使用します。

このパラメーターがデフォルトの状態では、このデータベースに対するディスカバリーが使用可能です。

このパラメーター値を「Disable」に変更することによって、機密データを持つデータベースをディスカバリー処理から隠すことができます。これは、データベースに対する他のセキュリティー・コントロールに追加して実行することができます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE CONFIGURATION コマンド』

discover_inst - サーバー・インスタンスのディスカバリー

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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] ENABLE [ENABLE, DISABLE]

このパラメーターは、このインスタンスが DB2 ディスカバリーによって検出できるかどうかを指定します。デフォルトでは「enable」で、インスタンスが検出できる指定であるのに対して、「disable」を指定すると、インスタンスは検出できません。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

パーティション・データベース環境

次のパラメーター・グループは、並列操作およびパーティション・データベース環境についての情報を記述しています。

- 『通信』
- 525 ページの『並列処理』

通信

次のパラメーターは、パーティション・データベース環境での通信について記述しています。

- 『conn_elapse - 接続経過時間』
- 518 ページの『fcm_num_anchors - FCM メッセージ・アンカー数』
- 519 ページの『fcm_num_buffers - FCM バッファー数』
- 520 ページの『fcm_num_connect - FCM 接続項目数』
- 521 ページの『fcm_num_rqb - FCM 要求ブロック数』
- 522 ページの『max_connretries - ノード接続再試行回数』
- 523 ページの『max_time_diff - ノード間最大時差』
- 524 ページの『start_stop_time - タイムアウトの開始および停止』

conn_elapse - 接続経過時間

構成タイプ データベース・マネージャー

適用 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] 10 [0-100]

単位 秒

このパラメーターは、TCP/IP 接続が 2 つのデータベース・パーティション・サーバー間に設定される必要がある時間 (秒数) を指定します。このパラメーターによって指定された時間内に接続が完了すれば、通信が確立されます。接続に失敗した場合は、通信を確立する試みがもう一度行われます。 *max_connretries* パラメーターによって指定された回数だけ接続が試みられ、しかもそのすべてがタイムアウトになった場合は、エラーになります。

関連資料:

- 522 ページの『max_connretries - ノード接続再試行回数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

fcm_num_anchors - FCM メッセージ・アンカー数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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
- ローカル・クライアントを持つサテライト・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] -1 [-1, 128 — *fcm_num_rqb*]

非パーティションのデータベース・システムでは、*intra_parallel* パラメーターがアクティブでないと、*fcm_num_anchors* を使用できません。

このパラメーターは、FCM メッセージ・アンカー数を指定します。エージェントは、自分たち同士でメッセージを送信するためにメッセージ・アンカーを使用します。デフォルト値 (-1) は、*fcm_num_rqb* で指定された値の 75% を指定します。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『高速コミュニケーション・マネージャー (FCM) 通信』

関連資料:

- 519 ページの『fcm_num_buffers - FCM バッファー数』
- 520 ページの『fcm_num_connect - FCM 接続項目数』
- 521 ページの『fcm_num_rqb - FCM 要求ブロック数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

fcm_num_buffers - FCM バッファー数

構成タイプ

データベース・マネージャー

適用

- ローカルとリモート・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]

32 ビット・プラットフォーム

512、1 024、または 4 096 [128 — *65 300*]

64 ビット・プラットフォーム

512、1 024、または 4 096 [128 — *524 288*]

- ローカルとリモート・クライアントを持つデータベース・サーバー。デフォルトは、1 024。
- ローカル・クライアントを持つデータベース・サーバー。デフォルトは、512。
- 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。デフォルトは、4 096。

単一パーティション・データベース・システムでは、*intra_parallel* パラメーターがアクティブでない場合、このパラメーターは使用されません。

このパラメーターは、データベース・サーバー間とデータベース・サーバー内の両方での内部通信 (メッセージ) 用として使用される 4 KB バッファーの数を指定します。

同じマシン上に複数の論理ノードがある場合は、このパラメーターの値を大きくする必要がある場合もあります。また、システム上のユーザーの数、システム上のデータベース・パーティション・サーバーの数、または複雑なアプリケーションのためにメッセージ・バッファーを使い尽くした場合は、このパラメーターの値を大きくしなければならない場合があります。

複数の論理ノードを非 AIX システムで使用している場合は、*fcm_num_buffers* バッファーの 1 つのプールが同じマシン上の論理ノードのすべてで共有されるのに対して、AIX では次のようになります。

- データベース・マネージャーで使用されている汎用メモリーに十分な余裕がある場合、 FCM バッファー・ヒープはそのメモリーから割り振られます。この状態では、それぞれのデータベース・パーティション・サーバーは独自の *fcm_num_buffers* バッファーを持ちます。データベース・パーティション・サーバーは、 FCM バッファー (これは、DB2 バージョン 5 では新規でした) のプールを共有しません。
- データベース・マネージャーが使用する汎用メモリーに十分な余裕がない場合、FCM バッファー・ヒープは独立メモリー域 (AIX 共用メモリー・セット) から割り振られます。その独立メモリー域は、同じマシン上のすべての論理ノードに共有されています。 *fcm_num_buffers* の 1 つのプールが、同じマシン上のすべての論理ノードに共有されることになります。これがすべての非 AIX プラットフォームの場合のデフォルト構成です。

使用中の値をもう一度調べて、複数の論理ノードがあるマシン上で割り振られる FCM バッファーの合計数を考慮してください。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『高速コミュニケーション・マネージャー (FCM) 通信』

関連資料:

- 520 ページの『fcm_num_connect - FCM 接続項目数』
- 518 ページの『fcm_num_anchors - FCM メッセージ・アンカー数』
- 521 ページの『fcm_num_rqb - FCM 要求ブロック数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

fcm_num_connect - FCM 接続項目数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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
- ローカル・クライアントを持つサテライト・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] -1 [-1, 128 — *fcm_num_rqb*]

非パーティションのデータベース・システムでは、*intra_parallel* パラメーターがアクティブでないと、*fcm_num_connect* を使用できません。

このパラメーターは、FCM *接続項目*数を指定します。エージェントは自分たち同士でデータを受け渡しするために接続項目を使用します。デフォルト値 (-1) は、*fcm_num_rqb* で指定された値の 75% を指定します。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『高速コミュニケーション・マネージャー (FCM) 通信』

関連資料:

- 519 ページの『fcm_num_buffers - FCM バッファー数』
- 518 ページの『fcm_num_anchors - FCM メッセージ・アンカー数』
- 521 ページの『fcm_num_rqb - FCM 要求ブロック数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

fcm_num_rqb - FCM 要求ブロック数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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
- ローカル・クライアントを持つサテライト・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]

UNIX 32 ビット・プラットフォーム
256、512、2 048 [128 — 120 000]

UNIX 64 ビット・プラットフォーム
256、512、2 048 [128 — 524 288]

Windows NT 32 ビット
10 000 [250 — 2 097 152]

Windows NT 64 ビット
256、512、2 048 [128 — 524 288]

- ローカルとリモート・クライアントを持つデータベース・サーバーの場合、デフォルトは 512 です。
- ローカル・クライアントを持つデータベース・サーバーの場合、デフォルトは 256 です。
- 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・サーバーの場合、デフォルトは 2 048 です。

非パーティションのデータベース・システムでは、*intra_parallel* パラメーターがアクティブでないと、*fcm_num_rqb* を使用できません。

このパラメーターは、FCM *要求ブロック*数を指定します。要求ブロックとは、FCM デーモンとエージェントの間、あるいはエージェント同士で情報を受け渡すためのメディアです。

要求ブロックの要件は、システム上のユーザー数、システム内のデータベース・パーティション・サーバー数、照会の複雑さによって異なります。最初はデフォルト値を使用し、データベース・システム・モニターからの結果を使用して、このパラメーターを微調整してください。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『高速コミュニケーション・マネージャー (FCM) 通信』

関連資料:

- 519 ページの『fcm_num_buffers - FCM バッファー数』
- 520 ページの『fcm_num_connect - FCM 接続項目数』
- 518 ページの『fcm_num_anchors - FCM メッセージ・アンカー数』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

max_connretries - ノード接続再試行回数

構成タイプ データベース・マネージャー

適用	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]	5 [0-100]

2 つのデータベース・パーティション・サーバー間に通信を確立する試みが失敗した (たとえば、*conn_elapse* パラメーターによって指定された値に達した) 場合は、*max_connretries* で、データベース・パーティション・サーバーに対して実行できる接続再試行回数を指定します。このパラメーターに指定された値を超えた場合は、エラーが戻されます。

関連資料:

- 517 ページの『conn_elapse - 接続経過時間』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

max_time_diff - ノード間最大時差

構成タイプ	データベース・マネージャー
適用	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	構成可能
デフォルト[範囲]	60 [1-1 440]
単位	分

各データベース・パーティション・サーバーには、それぞれ固有のシステム・クロックがあります。このパラメーターは、ノード構成ファイルにリストされているデータベース・パーティション・サーバー間に許容されている最大時差を分数で指定します。

複数のデータベース・パーティション・サーバーが 1 つのトランザクションに関連付けられていて、それらのクロックがこのパラメーターで指定されている時間内で同期していないと、そのトランザクションはリジェクトされ、SQLCODE が戻されます。 (トランザクションがリジェクトされるのは、データ変更がトランザクションに関連している場合だけです)。

DB2 では *協定世界時* (UTC) を使用しているので、このパラメーターを設定すると、異なる時間帯は考慮事項になりません。協定世界時とは、グリニッジ標準時と同じものです。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

start_stop_time - タイムアウトの開始および停止

構成タイプ	データベース・マネージャー
適用	ローカルとリモート・クライアントを持つデータベース・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]	10 [1 — 1 440]
単位	分

このパラメーターは時間を分単位で指定するため、すべてのデータベース・パーティション・サーバーは DB2START または DB2STOP コマンドに応答する必要があります。また、ADD DBPARTITIONNUM 操作時にタイムアウト値としても使用されます。

指定された時間内に DB2START コマンドに応答しないデータベース・パーティション・サーバーは、インスタンスのホーム・ディレクトリーの `sqllib` サブディレクトリーの `log` サブディレクトリーにある `db2start` エラー・ログにメッセージを送信します。それらを再始動する前に、これらのノードで DB2STOP を出す必要があります。

指定された時間内に DB2STOP コマンドに応答しないデータベース・パーティション・サーバーは、インスタンスのホーム・ディレクトリーの `sqllib` サブディレクトリーの `log` サブディレクトリーにある `db2stop` エラー・ログにメッセージを送信します。応答しないそれぞれのデータベース・パーティション・サーバーごと、またはそのすべてに対して、DB2STOP を出すことができます。 (すでに停止しているサーバーは、停止していることを示す記述を返します。)

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『ADD DBPARTITIONNUM コマンド』

並列処理

次のパラメーターは、並列処理について記述しています。

- 『intra_parallel - パーティション内並列処理機能の使用可能化』
- 『max_querydegree - 照会の最大並列処理多重度』

intra_parallel - パーティション内並列処理機能の使用可能化

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] NO (0) [SYSTEM (-1), NO (0), YES (1)]

値が -1 の場合、データベース・マネージャーが動作しているハードウェアに基づき、パラメーター値を「YES」または「NO」に設定します。

このパラメーターは、データベース・マネージャーでパーティション内並列処理を使用できるかどうかを指定します。

このパラメーターが「YES」の場合、並列操作によってパフォーマンスを改善できる操作の中には、データベース照会と索引作成が含まれます。

注: このパラメーター値を変更すると、パッケージがデータベースに再バインドされることがあり、パフォーマンスが低下する場合もあります。

関連資料:

- 525 ページの『max_querydegree - 照会の最大並列処理多重度』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

max_querydegree - 照会の最大並列処理多重度

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 ステートメント境界

デフォルト[範囲] -1 (ANY) [ANY, 1 — 32 767] (ANY はシステムによる判別を意味する)

このパラメーターは、データベース・マネージャーのこのインスタンスで実行中の SQL ステートメントで使用される、パーティション内並列処理の最大多重度を指定します。 SQL ステートメントの実行中に、そのステートメントによってパーティション内でこの数より多くの並列操作が行われることはありません。 *intra_parallel* 構成パラメーターを、「YES」に設定し、パーティション内並列処理を使用するデータベース・パーティションを使用可能にする必要があります。

この構成パラメーターのデフォルト値は -1 です。この値は、オプティマイザーによって決められた並列処理の多重度がシステムで使用されることを意味します。それ以外の場合は、ユーザー指定の値が使用されます。

注: SQL ステートメントの並列処理の多重度は、CURRENT DEGREE 特殊レジスターまたは DEGREE BIND オプションを使用して、ステートメントのコンパイル時に指定することができます。

アクティブなアプリケーションの照会の最大並列処理多重度は、SET RUNTIME DEGREE コマンドを使用して変更することができます。実際に実行時に使用される多重度は、次の値より低くなります。

- *max_querydegree* 構成パラメーター
- アプリケーション実行時の多重度
- SQL ステートメント・コンパイル時の多重度

実際の照会の並列処理多重度の判別方法に関する例外として、索引の作成時があります。この場合は、*intra_parallel* が「YES」で、表が複数のプロセッサーによる利点を生かせるだけの十分な大きさがあれば、索引の作成で、オンライン・プロセッサー: プロセッサー (最大で 6 つ) の数に 1 を加えた数を使用します。上記の他のパラメーター、BIND オプション、または特殊レジスターは関係ありません。

関連資料:

- 525 ページの『intra_parallel - パーティション内並列処理機能の使用可能化』
- 503 ページの『dft_degree - デフォルト多重度』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

インスタンス管理

データベース・マネージャー・インスタンスを管理するためのパラメーターがいくつかあります。この種のパラメーターは、次のカテゴリーに分けることができます。

- 『診断』
- 531 ページの『データベース・システム・モニター・パラメーター』
- 532 ページの『システム管理』
- 542 ページの『インスタンス管理』

診断

次のパラメーターは、データベース・マネージャーで利用できる診断情報を制御します。

- 『diaglevel - 診断エラー・キャプチャー・レベル』
- 528 ページの『diagpath - 診断データ・ディレクトリー・パス』
- 529 ページの『health_mon - ヘルス・モニター』
- 530 ページの『notifylevel - 通知レベル』

diaglevel - 診断エラー・キャプチャー・レベル

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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] 3 [0 — 4]

このパラメーターは、db2diag.log ファイルに記録される診断エラーのタイプを指定します。有効な値は次のとおりです。

0 - キャプチャーされる診断データなし

1 - 重大エラーのみ

2 - すべてのエラー

3 - すべてのエラーおよび警告

4 - すべてのエラー、警告、および通知メッセージ

diagpath 構成パラメーターを使って、エラー・ファイル、イベント・ログ・ファイル (Windows NT 上のみ)、アラート・ログ・ファイル、および *diaglevel* パラメーターの値に基づいて生成されるダンプ・ファイルが置かれるディレクトリーを指定します。

推奨: 問題の解決に役立つ追加の問題判別データを収集したい場合は、このパラメーターの値を大きくすることができます。

関連資料:

- 528 ページの『diagpath - 診断データ・ディレクトリー・パス』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

diagpath - 診断データ・ディレクトリー・パス

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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] Null [有効な任意のパス名]

このパラメーターを使用して、DB2 診断情報のための完全修飾パスを指定できます。このディレクトリーには、ご使用のプラットフォームに応じて、ダンプ・ファイル、トラップ・ファイル、エラー・ログ・ファイル、通知ファイルおよびアラート・ログ・ファイルが入れられます。

このパラメーターが null の場合は、診断情報が、以下のいずれかのディレクトリーまたはフォルダーにあるファイルに書き込まれます。

- サポートする Windows 環境
  - DB2INSTPROF 環境変数またはキーワードが**設定されていない**場合は、`x:¥SQLLIB¥DB2INSTANCE` に情報が書き込まれます。 `x:¥SQLLIB` は DB2PATH レジストリー変数または環境変数に指定されたドライブ参照およびディレクトリー、`DB2INSTANCE` はインスタンスの名前です。

    注: ディレクトリーに SQLLIB という名前を付ける必要はありません。

– DB2INSTPROF 環境変数またはキーワードが指定されている場合は、x:¥DB2INSTPROF¥DB2INSTANCE に情報が書き込まれます (DB2INSTPROF はインスタンス・プロファイル・ディレクトリーの名前、DB2INSTANCE はインスタンスの名前です)。
- UNIX ベースの環境の場合は、 INSTHOME/sqllib/db2dump です。INSTHOME は、インスタンス所有者のホーム・ディレクトリーです。

推奨: デフォルトを使用するか、複数のインスタンスの場合は、diagpath をひとつの場所にまとめて使用してください。

パーティション・データベース環境では、指定するパスはファイル共有システムにあることが必要です。

関連資料:

- 527 ページの『diaglevel - 診断エラー・キャプチャー・レベル』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

health_mon - ヘルス・モニター

構成タイプ	データベース・マネージャー
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]	オフ [オン; オフ]
関連パラメーター	

このパラメーターを使用すると、インスタンス、その関連データベース、およびデータベース・オブジェクトを健全性を表すさまざまな指標に従ってモニターするかどうか指定できます。 *health_mon* をオンにすると、ユーザーが選択したオブジェクトの健全性についての情報を、エージェントが収集します。ユーザーが設定したしきい値を基にして、オブジェクトが不健全な位置にあると見なされた場合は、通知が送信され、自動的にアクションが取られます。 *health_mon* をオフにした場合 (デフォルト)、オブジェクトの健全性はモニターされません。

ヘルス・センターまたは CLP を使用して、モニターしたいインスタンスおよびデータベース・オブジェクトを選択できます。また、ヘルス・モニターによって収集されたデータに基づいて、通知の送信先をどこにし、どのようなアクションを取る必要があるかについても指定できます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

notifylevel - 通知レベル

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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] 3 [0 — 4]

このパラメーターは、管理通知ログに書き込まれる管理通知メッセージのタイプを指定します。 UNIX プラットフォームでは、管理通知ログは、*instance*.nfy と呼ばれるテキスト・ファイルです。 Windows では、管理通知メッセージはイベント・ログに書き込まれます。エラーは DB2、ヘルス・モニター、キャプチャー・プログラムとアプライ・プログラム、およびユーザー・アプリケーションによって書き込むことができます。

このパラメーターの有効な値は、次のとおりです。

0 — 通知メッセージはキャプチャーされません (この設定は推奨できません)。

1 — 致命的エラーまたはリカバリー不能エラー。致命的エラーまたはリカバリー不能エラーだけがログに記録されます。これらの状態の中には、リカバリーのために DB2 サービスの援助が必要なものもあります。

2 — 即時アクションが必要です。システム管理者やデータベース管理者による即時アテンションを要する状態がログに記録されます。状態が解決されない場合は、致命的エラーに至る恐れがあります。エラーではないが、非常に大きなアクティビティー (たとえば、リカバリー) の通知もこのレベルでログに記録される場合があります。このレベルでは、ヘルス・モニター・アラームをキャプチャーします。

3 — 重要な情報であるが、即時アクションの必要はありません。即時アクションが必要とされるほど重大ではないが、最適なシステムではないことを示す状態がログに記録されます。このレベルでは、ヘルス・モニター・アラーム、ヘルス・モニター警告、およびヘルス・モニター・アテンションをキャプチャーします。

4 — 通知メッセージ。

管理通知ログには、値 *notifylevel* 以下 (この値を含む) の値をもつメッセージが組み込まれます。たとえば、*notifylevel* を 3 に設定すると、レベル 1、2、および 3 に該当するメッセージが管理通知ログに組み込まれることになります。

ユーザー・アプリケーションから通知ファイルまたは Windows イベント・ログに書き込みできるようにするためには、db2AdminMsgWrite API を呼び出す必要があります。

推奨: 問題の解決に役立つ追加の問題判別データを収集したい場合は、このパラメーターの値を大きくすることができます。また、ヘルス・モニターの構成で定義されている連絡先に通知を送信するためには、*notifylevel* を 2 以上の値に設定する必要があります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

データベース・システム・モニター・パラメーター

次のパラメーターを使用すると、データベース・システム・モニターの各種の設定を制御することができます。

- 『dft_monswitches - デフォルトのデータベース・システム・モニター・スイッチ』

dft_monswitches - デフォルトのデータベース・システム・モニター・スイッチ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 デフォルトでオンの dft_mon_timestamp を除いて、すべてのスイッチがオフ

このパラメーターは、パラメーターの各ビットで内部的に表されるスイッチを設定できる、唯一のパラメーターです。これらのスイッチは、次のパラメーターを設定することによって、それぞれ別個に独立して更新できます。

dft_mon_uow スナップショット・モニターの作業単位 (UOW) スイッチのデフォルト値

dft_mon_stmt スナップショット・モニターのステートメント・スイッチのデフォルト値

dft_mon_table	スナップショット・モニターの表スイッチのデフォルト値
dft_mon_bufpool	スナップショット・モニターのバッファー・プール・スイッチのデフォルト値
dft_mon_lock	スナップショット・モニターのロック・スイッチのデフォルト値
dft_mon_sort	スナップショット・モニターのソート・スイッチのデフォルト値
dft_mon_timestamp	スナップショット・モニターのタイム・スタンプ・スイッチのデフォルト値

推奨: どのスイッチも (ただし、dft_mon_timestamp を除く)、それを ON にした場合は、そのスイッチに関連するモニター・データの収集を、データベース・マネージャーに指示します。多くのモニター・データを収集すると、データベース・マネージャーのオーバーヘッドが増加し、システム・パフォーマンスに影響を与えます。 dft_mon_timestamp スイッチを OFF にすることが、CPU 使用率が 100% に近付くにつれて重要になります。こうなったときは、タイム・スタンプを発行するのに必要な CPU 時間が大幅に増大します。さらに、タイム・スタンプ・スイッチを OFF にすると、モニター・スイッチのコントロール下にある他のデータの合計コストが大幅に削減されます。

すべてのモニター・アプリケーションは、アプリケーションがその最初のモニター要求を出したときに、これらのデフォルト・スイッチ設定を継承します (たとえば、スイッチを設定し、イベント・モニターを活動化し、スナップショットを取る)。構成ファイル内のスイッチは、データベース・マネージャーの始動時からデータの収集を開始したい場合にのみ、オンに設定してください。 (オンに設定しない場合、各モニター・アプリケーションはそれ自体のスイッチを設定することができ、収集したデータはそのスイッチが設定された時刻に関連付けられるようになります。)

関連資料:

- 「コマンド・リファレンス」の『GET MONITOR SWITCHES コマンド』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

システム管理

次のパラメーターは、システム管理と関連があります。

- 533 ページの『comm_bandwidth - 通信スピード』
- 534 ページの『cpuspeed - CPU 速度』
- 535 ページの『dft_account_str - デフォルト・チャージバック・アカウント』

- 536 ページの『federated - フェデレーテッド・データベース・システム・サポート』
- 536 ページの『jdk_path - Software Developer's Kit for Java インストール・パス』
- 537 ページの『nodetype - マシン・ノード・タイプ』
- 537 ページの『numdb - ホストおよび iSeries データベースを含めた並行アクティブ・データベースの最大数』
- 539 ページの『tp_mon_name - トランザクション・プロセッサー・モニター名』
- 541 ページの『util_impact_lim - インスタンス影響ポリシー』

comm_bandwidth - 通信スピード

構成タイプ	データベース・マネージャー
適用	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	ステートメント境界
デフォルト[範囲]	-1 [.1 - 100 000] 値が -1 の場合は、パラメーター値がデフォルトにリセットされます。デフォルト値は、高速スイッチが使用されているかどうかに基づいて計算されます。
単位	MB/秒

通信スピードの計算値 (MB/秒) は、パーティション・データベース・システムのデータベース・パーティション・サーバー間で特定の操作を実行するコストを見積もる場合に、SQL オプティマイザーで使用されます。オプティマイザーがクライアントとサーバーの間の通信のコストをモデル化することはないので、このパラメーターには、データベース・パーティション・サーバー間の公称帯域幅のみが反映される必要があります。

この値を明示的に設定すると、テスト・システム上に実稼働環境をモデル化したり、ハードウェアのアップグレードの影響を調査したりすることができます。

推奨: 異なる環境をモデル化する場合にのみ、このパラメーターを調整する必要があります。

通信スピードは、オプティマイザーがアクセス・パスの決定にあたって使用します。このパラメーターを変更した場合は、アプリケーションの再バインド (REBIND PACKAGE コマンドを使用) を考慮してください。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

cpuspeed - CPU 速度

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 ステートメント境界

デフォルト[範囲] -1 [1^{-10} — 1] 値を -1 にすると、パラメーター値は、測定プログラムの実行に基づいてリセットされます。

単位 秒

CPU 速度 (ミリ秒/命令) は、ある操作を実行するコストを見積もるために、 SQL オプティマイザーが使用します。このパラメーターの値は、CPU 速度を計測するために設計されたプログラムからの出力に基づいて、データベース・マネージャーのインストール時に自動的に設定されます。ベンチマーク結果が次のいずれかの理由で利用できない場合は、プログラムが実行されます。

- プラットフォームが `db2spec.dat` ファイルをサポートしない場合。
- `db2spec.dat` ファイルが**見つからない**場合。
- IBM RISC システム/6000 モデル 530H 用のデータがファイルにない場合。
- このマシン用のデータがファイルにない場合。

この値を明示的に設定すると、テスト・システム上に実稼働環境をモデル化したり、ハードウェアのアップグレードの影響を調査したりすることができます。この値を -1 に設定すると、*cpuspeed* が再計算されます。

推奨: 異なる環境をモデル化する場合にのみ、このパラメーターを調整する必要があります。

CPU 速度は、オプティマイザーが、アクセス・パスの決定で使用します。このパラメーターを変更した場合は、アプリケーションの再バインド (REBIND PACKAGE コマンドを使用) を考慮してください。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

dft_account_str - デフォルト・チャージバック・アカウント

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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] Null [任意の有効ストリング]

アプリケーション接続要求のそれぞれについて、DB2 Connect が生成した接頭部とユーザーが指定した接尾部からなるアカウント ID が、アプリケーション・リクエスターから DRDA アプリケーション・サーバーへ送信されます。このアカウント情報は、リソースの使用状況を各ユーザー・アクセスに関連付けるためのメカニズムとして、システム管理者に提供されます。

注: このパラメーターは DB2 Connect にのみ適用できます。

接尾部は、アプリケーション・プログラムが呼び出す `sqlesact()` API、またはユーザー設定の環境変数 DB2ACCOUNT によって与えられます。接尾部が API と環境変数のどちらからも与えられない場合、DB2 Connect はデフォルトの接尾部としてこのパラメーターの値を使用します。このパラメーターが特に有用なのは、会計情報ストリングを DB2 Connect に転送できる機能のない下位データベース・クライアント (バージョン 2 よりも前のクライアント) の場合です。

推奨: 以下を使用して、この会計情報ストリングを設定してください。

- 英字 (A から Z)
- 数字 (0 から 9)
- 下線 (_)

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

federated - フェデレーテッド・データベース・システム・サポート

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] No [Yes; No]

このパラメーターは、データ・ソース (DB2 ファミリーおよび Oracle など) によって管理されるデータに対する分散要求をアプリケーションが実行できるようにする機能を使用可能または使用不能にします。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

jdk_path - Software Developer's Kit for Java インストール・パス

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] Null [有効パス]

このパラメーターは、Java ストアード・プロシージャーおよびユーザー定義関数を実行する場合に使用される、Software Developer's Kit for Java をインストールするディレクトリーを指定します。 Java インタープリターで使用される CLASSPATH や他の環境変数は、このパラメーターの値から計算されます。

関連資料:

- 421 ページの『java_heap_sz - Java インタープリター最大ヒープ・サイズ』

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

nodetype - マシン・ノード・タイプ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 通知

このパラメーターは、マシンにインストールされている DB2 製品についての情報を提供するものであり、したがって、データベース・マネージャー構成のタイプについての情報を提供します。このパラメーターによって戻される可能性のある値、および該当のノード・タイプに関連した製品が、以下に示してあります。

- **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** - ローカルおよびリモート・データベース・クライアントをサポートし、他のリモート・データベース・サーバーにアクセスできる DB2 サーバー製品。
- **クライアント** - リモート・データベース・サーバーにアクセスできるデータベース・クライアント。
- **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・データベース・クライアントをサポートし、他のリモート・データベース・サーバーにアクセスできる DB2 リレーショナル・データベース管理システム。
- **ローカルおよびリモート・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** - ローカルおよびリモート・データベース・クライアントをサポートし、他のリモート・データベース・サーバーにアクセスでき、パーティション並列処理に対応可能な DB2 サーバー製品。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』

numdb - ホストおよび iSeries データベースを含めた並行アクティブ・データベースの最大数

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲]

UNIX 8 [1 — 256]

ローカル・クライアントおよびリモート・クライアントを持つ Windows データベース・サーバー
8 [1 — 256]

ローカル・クライアントを持つ Windows データベース・サーバー
3 [1 — 256]

単位 カウンター

このパラメーターは、並行してアクティブにできる (つまり、アプリケーションを接続できる) ローカル・データベースの数、または DB2 Connect サーバー上にカタログできる異なるデータベース別名の最大数を指定します。各データベースはストレージを使い果たすと、アクティブなデータベースは新しい共用メモリー・セグメントを使用します。

推奨: 通常、この値の最良の設定は、増加を見込んで、データベース・マネージャーにすでに定義されている実際のデータベース数に 10% を加えた値にすることです。

numdb パラメーターを変更すると、割り振られたメモリーの合計量に影響を与えます。したがって、このパラメーターは頻繁に変更しないことをお勧めします。このパラメーターを更新するときは、データベースやそのデータベースに接続されているアプリケーションにメモリーを割り振ることができる、他の構成パラメーターについて考慮する必要があります。

関連資料:

- 398 ページの『app_ctl_heap_sz - アプリケーション・コントロール・ヒープ・サイズ』
- 410 ページの『sortheap - ソート・ヒープ・サイズ』
- 413 ページの『aslheapsz - アプリケーション・サポート層ヒープ・サイズ』
- 404 ページの『applheapsz - アプリケーション・ヒープ・サイズ』
- 390 ページの『locklist - ロック・リスト用最大ストレージ』
- 389 ページの『dbheap - データベース・ヒープ』
- 412 ページの『stmtheap - ステートメント・ヒープ・サイズ』
- 422 ページの『mon_heap_sz - データベース・システム・モニター・ヒープ・サイズ』
- 411 ページの『stat_heap_sz - 統計ヒープ・サイズ』

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 388 ページの『database_memory - データベース共用メモリー・サイズ』

tp_mon_name - トランザクション・プロセッサー・モニター名

構成タイプ

データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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

構成可能

デフォルト

デフォルトなし

有効値

- CICS
- MQ
- ENCINA
- CB
- SF
- TUXEDO
- TOPEND
- ブランクまたは何か別の値 (ただし、UNIX および Windows の場合であり、Solaris または SINIX の場合は、他に指定できる値はない)

このパラメーターでは、使用されているトランザクション処理 (TP) モニター製品の名前を識別します。

- アプリケーションが WebSphere Enterprise Edition CICS 環境で実行される場合は、このパラメーターは「CICS」に設定する必要があります。
- アプリケーションが WebSphere Enterprise Edition Encina 環境で実行される場合は、このパラメーターは「ENCINA」に設定する必要があります。
- アプリケーションが WebSphere Enterprise Edition Component Broker 環境で実行される場合は、このパラメーターは「CB」に設定する必要があります。
- アプリケーションが IBM MQSeries 環境で実行される場合は、このパラメーターは「MQ」に設定する必要があります。

- アプリケーションが BEA Tuxedo 環境で実行される場合は、このパラメーターは「TUXEDO」に設定する必要があります。
- アプリケーションが IBM San Francisco 環境で実行される場合は、このパラメーターは「SF」に設定する必要があります。

IBM WebSphere EJB および **Microsoft Transaction Server** のユーザーは、このパラメーターの値を構成する必要はありません。

上記の製品のいずれも使用されていない場合は、このパラメーターは構成しないで、ブランクのままにしておく必要があります。

以前のバージョンの DB2 Universal Database の Windows NT 版では、XA トランザクション・マネージャーの関数 *ax_reg* および *ax_unreg* が入っていた DLL のパスおよび名前が、このパラメーターに含まれていました。このフォーマットは、まだサポートされています。このパラメーターの値が上記の TP モニター名のいずれにも一致しない場合、値は、*ax_reg* および *ax_unreg* 関数が入っているライブラリー名であると見なされます。 UNIX および Windows NT 環境には、これが該当します。

TXSeries CICS および Encina のユーザー: 以前のバージョンのこの製品の Windows NT 版では、このパラメーターを「libEncServer:C」または「libEncServer:E」として構成する必要がありました、これはまだサポートされてはいますが、必須ではなくなりました。パラメーターを「CICS」または「ENCINA」として構成すれば十分です。

MQSeries ユーザー: 以前のバージョンのこの製品の Windows NT 版では、このパラメーターを「mqmax」として構成する必要がありました。これはまだサポートされてはいますが、必須ではなくなりました。パラメーターを「MQ」として構成すれば十分です。

Component Broker ユーザー: 以前のバージョンのこの製品の Windows NT 版では、このパラメーターを「somtrx1i」として構成する必要がありました。これはまだサポートされてはいますが、必須ではなくなりました。パラメーターを「CB」として構成すれば十分です。

San Francisco ユーザー: 以前のバージョンのこの製品の Windows NT 版では、このパラメーターを「ibmsfDB2」として構成する必要がありました。これはまだサポートされてはいますが、必須ではなくなりました。パラメーターを「SF」として構成すれば十分です。

このパラメーターに指定できるストリングの最大長は 19 文字です。

このパラメーターは、DB2 Universal Database の XA OPEN ストリングで構成することも可能です。複数のトランザクション処理モニターで単一の DB2 インスタンスを使用している場合は、この機能を使用する必要があります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 「*管理ガイド: プランニング*」の『xa_open ストリング形式』

util_impact_lim - インスタンス影響ポリシー

構成タイプ データベース・マネージャー

適用

- 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

デフォルト[範囲] 100 [1 - 100]

単位 許可するワークロードへの影響のパーセンテージ

データベース管理者 (DBA) は、このパラメーターを使用して、ワークロードに対するスロットル・ユーティリティーの性能の低下を制限することができます。次いで DBA は、重要な実動期間中にオンライン・ユーティリティーを実行し、実動作業のパフォーマンスへの影響が許容限度を超えないようにすることができます。

たとえば、*util_impact_lim* (影響ポリシー) の値を 10 に設定した場合、スロットルされたバックアップの起動は、ワークロードに 10% 以上の影響を与えることはありません。

util_impact_lim が 100 (デフォルト値) になっていると、ユーティリティーの起動はスロットルされません。この場合、ユーティリティーは、ワークロードに対して自由に影響を与えることができます (好ましくない)。 *util_impact_lim* を 100 以下の値に設定すると、ユーティリティーをスロットル・モードで起動できます。また、スロットル・モードで実行する場合は、ユーティリティーをゼロ以外の優先順位で起動する必要があります。

推奨事項: 多くの場合、*util_impact_lim* は低い値 (たとえば、1 ～ 10 程度) に設定したほうが益があります。

スロットルされたユーティリティーは、通常、スロットルされていないユーティリティーよりも完了に時間がかかります。ユーティリティーの実行に極端に長い時間がかかっている場合は、*util_impact_lim* の値を増やすか、*util_impact_lim* を 100 にして、スロットルを完全に無効にしてください。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

インスタンス管理

次のパラメーターは、データベース・マネージャー・インスタンスのセキュリティーおよび管理と関連があります。

- 『authentication - 認証タイプ』
- 543 ページの『catalog_noauth - 権限なしで許可されるカタログ』
- 544 ページの『clnt_krb_plugin - クライアント Kerberos プラグイン』
- 545 ページの『clnt_pw_plugin - クライアント・ユーザー ID パスワード・プラグイン』
- 545 ページの『dftdbpath - デフォルト・データベース・パス』
- 547 ページの『fed_noauth - フェデレーテッド・データベース認証をバイパス』
- 547 ページの『group_plugin - グループ・プラグイン』
- 548 ページの『local_gssplugin - ローカル・インスタンス・レベル許可に使用する GSS API プラグイン』
- 548 ページの『srvcon_auth - サーバーでの着信接続の認証タイプ』
- 549 ページの『srvcon_gssplugin_list - サーバーでの着信接続用の GSS API プラグインのリスト』
- 550 ページの『srvcon_pw_plugin - サーバーでの着信接続用のユーザー ID-パスワード・プラグイン』
- 551 ページの『srv_plugin_mode - サーバー・プラグイン・モード』
- 551 ページの『sysadm_group - システム管理権限グループ名』
- 552 ページの『sysctrl_group - システム制御権限グループ名』
- 553 ページの『sysmaint_group - システム保守権限グループ名』
- 554 ページの『sysmon_group - システム・モニター権限グループ名』
- 555 ページの『trust_allclnts - 全クライアントのトラステッド化』
- 556 ページの『trust_clntauth - トラステッド・クライアント認証』
- 557 ページの『use_sna_auth - SNA 認証の使用』

authentication - 認証タイプ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] SERVER [CLIENT; SERVER; SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

このパラメーターは、ユーザーの認証が行われる方法とその場所を指定します。

認証が SERVER の場合は、サーバー上で認証を行うために、ユーザー ID とパスワードがクライアントからサーバーに送信されます。値 SERVER_ENCRYPT は、ネットワークを介して送信されるパスワードが暗号化されることを除き、SERVER と同じように機能します。

注: 共通基準に準拠した構成の場合、サポートされる値は SERVER だけです。

値 CLIENT は、すべての認証がクライアントで行われることを示します。この場合はサーバーで認証を行う必要はありません。

KERBEROS の値は、認証が、Kerberos セキュリティー・プロトコルを使用して Kerberos サーバー上で行われることを意味します。 Kerberos セキュリティー・システムをサポートするサーバーおよびクライアントの認証タイプが KRB_SERVER_ENCRYPT の場合、有効なシステム認証タイプは KERBEROS です。クライアントが Kerberos セキュリティー・システムをサポートしない場合のシステム認証タイプは、事実上 SERVER_ENCRYPT に相当します。

値 GSSPLUGIN は、外部の GSSAPI ベースのセキュリティー・メカニズムを使用して認証が実行されることを意味します。 GSSPLUGIN セキュリティー・メカニズムをサポートするサーバーおよびクライアントの認証タイプが GSS_SERVER_ENCRYPT の場合、有効なシステム認証タイプは GSSPLUGIN です (つまり、クライアントがサーバーのプラグインのいずれかをサポートしている場合)。クライアントが GSSPLUGIN セキュリティー・メカニズムをサポートしない場合のシステム認証タイプは、事実上 SERVER_ENCRYPT に相当します。

推奨: 一般に、デフォルト値 (SERVER) が適切です。

関連資料:

- 「*コマンド・リファレンス*」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「*コマンド・リファレンス*」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「*コマンド・リファレンス*」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

catalog_noauth - 権限なしで許可されるカタログ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント

- 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]

ローカルとリモート・クライアントを持つデータベース・サーバー
NO [NO (0) — YES (1)]

クライアント; ローカル・クライアントを持つデータベース・サーバー
YES [NO (0) — YES (1)]

このパラメーターは、ユーザーがデータベースとノード、または DCS と ODBC ディレクトリーを SYSADM 権限なしでカタログおよびアンカタログできるようにするかを指定します。パラメーターのデフォルト値 (0) は、SYSADM 権限が必要なことを示します。このパラメーターが 1 (yes) に設定された場合、SYSADM 権限は必要なくなります。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

clnt_krb_plugin - クライアント Kerberos プラグイン

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] Null または IBMkrb5 [任意の有効ストリング]

このパラメーターでは、クライアント側の認証とローカル許可に使用するデフォルト Kerberos プラグイン・ライブラリーの名前を指定します。デフォルトの値は、

UNIX ベースのシステムでは NULL で、 Windows オペレーティング・システムでは IBMkrb5 です。このプラグインは、クライアントが KERBEROS 認証を使用して認証されるときに使用されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

clnt_pw_plugin - クライアント・ユーザー ID パスワード・プラグイン

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] Null [任意の有効ストリング]

このパラメーターでは、クライアント側の認証とローカル許可に使用するユーザー ID-パスワード・プラグイン・ライブラリーの名前を指定します。デフォルトの値は NULL で、DB2 提供のユーザー ID-パスワード・プラグイン・ライブラリーが使用されます。このプラグインは、クライアントが CLIENT、SERVER、または SERVER_ENCRYPT 認証を使用して認証されるときに使用されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

dftdbpath - デフォルト・データベース・パス

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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]

UNIX	インスタンス所有者のホーム・ディレクトリー [任意の既存のパス]
Windows	DB2 がインストールされているドライブ [任意の既存のパス]

このパラメーターには、データベース・マネージャーのもとでデータベースを作成するために使用されるデフォルト・ファイル・パスが含まれています。データベースの作成時にパスを指定しなかった場合、データベースは、*dftdbpath* パラメーターで指定されたパスに作成されます。

パーティション・データベース環境では、データベースが作成されるパスは、NFS マウント・パス (UNIX ベースのプラットフォームの場合) でもネットワーク・ドライブ (Windows 環境の場合) でもないようにする必要があります。指定のパスが各データベース・パーティション・サーバーに物理的に存在している必要があります。混乱を避けるためには、それぞれのデータベース・パーティション・サーバー上にローカルにマウントされているパスを指定するのが一番良い方法です。パスの最大長は 205 文字です。システムは、パスの終わりにノード名を追加します。

データベースは巨大なサイズになる場合があり、また多くのユーザーがデータベースを作成することも (環境と目的による) あるため、すべてのデータベースを指定のロケーションに作成して格納することができるようにすると便利です。また、保全性やバックアップおよびリカバリーのためにも、データベースをアプリケーションとデータから分離しておくことも有益です。

UNIX 環境の場合、*dftdbpath* 名の長さは、215 文字以下の有効で絶対的なパス名でなければなりません。Windows の場合は、*dftdbpath* にはドライブ名が指定でき、オプションで後にコロンを続けても構いません。

推奨: 可能であれば、ボリュームの大きいデータベースは、オペレーティング・システム・ファイルやデータベース・ログなど、頻繁にアクセスされるデータとは異なるディスクに配置してください。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

fed_noauth - フェデレーテッド・データベース認証をバイパス

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 [Yes; No]

fed_noauth が *yes* に設定され、 *authentication* が *server* または *server_encrypt* に設定され、*federated* が *yes* に設定されていると、インスタンスでの認証はバイパスされます。認証はデータ・ソースで行われると見なされます。 *fed_noauth* が *yes* に設定されているときは、注意してください。認証はクライアントでも、DB2 でも行われません。SYSADM 認証名を知っているユーザーであれば、このフェデレーテッド・サーバーの SYSADM 権限を持つことができます。

関連資料:

- 542 ページの『authentication - 認証タイプ』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』
- 536 ページの『federated - フェデレーテッド・データベース・システム・サポート』

group_plugin - グループ・プラグイン

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] Null [任意の有効ストリング]

このパラメーターでは、グループ・プラグイン・ライブラリーの名前を指定します。デフォルトの値は NULLで、DB2 はオペレーティング・システムのグループ・ルックアップを使用します。このプラグインはすべてのグループ参照数に使用されます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

local_gssplugin - ローカル・インスタンス・レベル許可に使用する GSS API プラグイン

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト[範囲] Null [任意の有効ストリング]

このパラメーターでは、*authentication* データベース・マネージャー構成パラメーターの値が GSSPLUGIN または GSS_SERVER_ENCRYPT に設定されている場合に、インスタンス・レベル・ローカル許可に使用するデフォルトの GSS API プラグイン・ライブラリーの名前を指定します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

srvcon_auth - サーバーでの着信接続の認証タイプ

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] Null [CLIENT; SERVER; SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

このパラメーターでは、サーバーで着信接続を処理する際に、ユーザー認証を実行する方法と場所を指定します。このパラメーターは現行の認証タイプをオーバーライドします。値を指定しない場合、DB2 は *authentication* データベース・マネージャー構成パラメーターの値を使用します。

各認証タイプについては、542 ページの『authentication - 認証タイプ』 を参照してください。

関連資料:

- 「*コマンド・リファレンス*」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「*コマンド・リファレンス*」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「*コマンド・リファレンス*」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

srvcon_gssplugin_list - サーバーでの着信接続用の GSS API プラグインのリスト

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] Null [任意の有効ストリング]

このパラメーターでは、データベース・サーバーがサポートする GSS API プラグイン・ライブラリーを指定します。デフォルトでは、この値は Null です。認証タイプが GSSPLUGIN で、このパラメーターが NULL であると、エラーが戻されます。認証タイプが KERBEROS で、このパラメーターが NULL であると、 DB2 提供の Kerberos モジュールまたはライブラリーが使用されます。別の認証タイプが使用されている場合には、このパラメーターは使用されません。

認証タイプが KERBEROS であり、このパラメーターの値が NULL ではない場合、リストに含まれる Kerberos プラグインは 1 つでなければならず、このプラグインが認証に使用されます (リスト内の他のすべての GSS プラグインは無視されます)。複数の Kerberos プラグインがある場合は、エラーが戻されます。

GSS API プラグイン名どうしはコンマ (,) で区切ります。コンマの前後にスペースは不要です。プラグイン名は優先する順序にリストします。このパラメーターは、*srvcon_auth* パラメーターに KERBEROS、KRB_SERVER_ENCRYPT、 GSSPLUGIN または GSS_SERVER_ENCRYPT が指定されている場合、または *srvcon_auth* が指定されておらず authentication に KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN または GSS_SERVER_ENCRYPT が指定されている場合に、サーバーでの着信接続を処理します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

srvcon_pw_plugin - サーバーでの着信接続用のユーザー ID-パスワード・プラグイン

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] Null [任意の有効ストリング]

このパラメーターでは、サーバー側の認証に使用するデフォルトのユーザー ID-パスワード・プラグイン・ライブラリーの名前を指定します。デフォルトの値は NULL で、DB2 提供のユーザー ID-パスワード・プラグイン・ライブラリーが使用されます。

このパラメーターは、 *srvcon_auth* パラメーターに SERVER または SERVER_ENCRYPT が指定されている場合、または *srvcon_auth* が指定されておらず *authentication* に CLIENT、SERVER、または SERVER_ENCRYPT が指定されている場合に、サーバーでの着信接続を処理します。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

srv_plugin_mode - サーバー・プラグイン・モード

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] UNFENCED

このパラメーターでは、fenced モードまたは unfenced モードでプラグインを実行するかどうかを指定します。 unfenced モードのみがサポートされています。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

sysadm_group - システム管理権限グループ名

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト Null

システム管理 (SYSADM) 権限はデータベース・マネージャー内の最高レベルの権限であり、すべてのデータベース・オブジェクトをコントロールします。このパラメーターは、データベース・マネージャー・インスタンスの SYSADM 権限を持つグループ名を定義します。

SYSADM 権限は、特定の運用環境で使用されるセキュリティー機能によって決定されます。

- Windows 98 オペレーティング・システムでは、 SYSADM グループは NULL でなければならない。

  システム・セキュリティーを使用する場合は、 Windows 98 クライアントではこのパラメーターは「NULL」である必要があります。これは、 Windows 98 オペレーティング・システムではグループ情報を保管しないため、ユーザーが指定された SYSADM グループのメンバーであるかどうかを判別する方法がないからです。 グループ名が指定される場合、ユーザーはそのメンバーにはなれません。
- Windows NT および Windows 2000 オペレーティング・システムでは、このパラメーターを 8 文字以下の名前を持つローカル・グループに設定することができ、Windows NT および Windows 2000 セキュリティー・データベースで定義されます。このパラメーターに「NULL」を指定する場合、管理者グループのすべてのメンバーは SYSADM 権限を持っています。
- UNIX ベースのシステムでは、このパラメーターの値として「NULL」を指定する場合、SYSADM グループのデフォルトは、インスタンス所有者の 1 次グループです。

  値が「NULL」でない場合、SYSADM グループは有効な UNIX グループ名です。

パラメーターをそのデフォルト (NULL) 値にリストアするため UPDATE DBM CFG USING SYSADM_GROUP NULL を使用してください。 キーワード「NULL」は大文字で指定する必要があります。 DB2 コントロール・センターの「インスタンスの構成 (Configure Instance)」ノートブックを使用することもできます。

関連資料:

- 552 ページの『sysctrl_group - システム制御権限グループ名』
- 553 ページの『sysmaint_group - システム保守権限グループ名』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

sysctrl_group - システム制御権限グループ名

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト Null

このパラメーターは、システム・コントロール権限 (SYSCTRL) を持つグループ名を定義します。 SYSCTRL には、システム・リソースに影響を与える操作を許可する特権がありますが、データへの直接アクセスはできません。

警告: システム・セキュリティーが使用される (すなわち、認証が CLIENT、SERVER、 DCS、またはほかの有効な認証である) 場合、 Windows 98 クライアントでは、このパラメーターは Null にする必要があります。これは、 Windows 98 オペレーティング・システムはグループ情報を保管しないため、ユーザーが、指定された SYSCTRL グループのメンバーであるかどうかを判別する方法がないからです。グループ名が指定される場合、ユーザーはそのメンバーにはなれません。

パラメーターをそのデフォルト (NULL) 値にリストアするため UPDATE DBM CFG USING SYSCTRL_GROUP NULL を使用してください。キーワード「NULL」は大文字で指定する必要があります。 DB2 コントロール・センターの「インスタンスの構成 (Configure Instance)」ノートブックを使用することもできます。

関連資料:

- 551 ページの『sysadm_group - システム管理権限グループ名』
- 553 ページの『sysmaint_group - システム保守権限グループ名』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

sysmaint_group - システム保守権限グループ名

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト Null

このパラメーターは、システム保守 (SYSMAINT) 権限を持つグループ名を定義します。 SYSMAINT は、データに直接アクセスすることなしに、インスタンスに関連するすべてのデータベースでの保守操作を実行する特権を持っています。

警告: システム・セキュリティーが使用される (すなわち、認証が CLIENT、SERVER、 DCS、またはほかの有効な認証である) 場合、 Windows 98 クライアントでは、このパラメーターは Null にする必要があります。これは、 Windows 98 オペレーティング・システムはグループ情報を保管しないため、ユーザーが、指定された SYSMAINT グループのメンバーであるかどうかを判別する方法がないからです。グループ名が指定される場合、ユーザーはそのメンバーにはなれません。

パラメーターをそのデフォルト (NULL) 値にリストアするため UPDATE DBM CFG USING SYSMAINT_GROUP NULL を使用してください。キーワード「NULL」は大文字で指定する必要があります。 DB2 コントロール・センターの「インスタンスの構成 (Configure Instance)」ノートブックを使用することもできます。

関連資料:

- 551 ページの『sysadm_group - システム管理権限グループ名』
- 552 ページの『sysctrl_group - システム制御権限グループ名』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

sysmon_group - システム・モニター権限グループ名

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・クライアントを持つデータベース・サーバー
- クライアント
- 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 構成可能

デフォルト Null

このパラメーターは、システム・モニター (SYSMON) 権限を持つグループ名を定義します。インスタンス・レベルで SYSMON 権限を持つユーザーは、データベース・マネージャー・インスタンスまたはそのデータベースのデータベース・システム・モニター・スナップショットを取ることができます。 SYSMON 権限では、以下のコマンドを使用できます。

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

ユーザーは SYSADM、SYSCTRL、または SYSMAINT 権限を持つと自動的に、データベース・システム・モニター・スナップショットを取ることも、上記のコマンドを使用することもできます。

警告: システム・セキュリティーが使用される (すなわち、認証が CLIENT、SERVER、 DCS、またはほかの有効な認証である) 場合、 Windows 98 クライアントでは、このパラメーターは Null にする必要があります。これは、 Windows 98 オペレーティング・システムはグループ情報を保管しないため、ユーザーが、指定された SYSMON グループのメンバーであるかどうかを判別する方法がないからです。グループ名が指定される場合、ユーザーはそのメンバーにはなれません。

パラメーターをそのデフォルト (NULL) 値にリストアするため UPDATE DBM CFG USING SYSMON_GROUP NULL を使用してください。キーワード「NULL」は大文字で指定する必要があります。 DB2 コントロール・センターの「インスタンスの構成 (Configure Instance)」ノートブックを使用することもできます。

関連資料:

- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

trust_allclnts - 全クライアントのトラステッド化

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] YES [NO, YES, DRDAONLY]

このパラメーターは、*authentication* パラメーターが CLIENT に設定されている場合のみ、アクティブになります。

このパラメーターと *trust_clntauth* は、データベース環境に対するユーザーの妥当性が検証される場所を決定する場合に使用されます。

このパラメーターのデフォルト「YES」を受け入れると、すべてのクライアントがトラステッド・クライアントとして扱われます。これは、セキュリティーのレベルをクライアントで利用できることをサーバーが前提にしていること、およびクライアントでユーザーを有効にできる可能性を示しています。

このパラメーターは、*authentication* パラメーターが CLIENT に設定されている場合、「NO」にしか変更できません。このパラメーターを「NO」に設定されている場合、非トラステッド・クライアントは、サーバーに接続するときにユーザー ID とパスワードの組み合わせを指定する必要があります。非トラステッド・クライアントは、認証ユーザーのセキュリティー・サブシステムを持たないオペレーティング・システム・プラットフォームです。

このパラメーターを「DRDAONLY」に設定すると、DB2 for OS/390 and z/OS、DB2 for VM/VSE、および DB2 for OS/400 からのクライアントを除く、すべてのクライアントに対して保護されます。 これらのクライアントは、クライアント側の認証を行うことしか承認されません。他のすべてのクライアントは、サーバーによって認証されるユーザー ID とパスワードを指定する必要があります。

trust_allclnts が「DRDAONLY」に設定されていると、*trust_clntauth* パラメーターは、クライアントが認証される場所を判別するために使用されます。 *trust_clntauth* が「CLIENT」に設定されている場合、認証がクライアントで発生します。 *trust_clntauth* が「SERVER」に設定されている場合、認証がクライアントで発生するのは、パスワードが指定されていない場合であり、パスワードが指定されている場合は、サーバーで発生します。

関連資料:

- 542 ページの『authentication - 認証タイプ』
- 556 ページの『trust_clntauth - トラステッド・クライアント認証』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

trust_clntauth - トラステッド・クライアント認証

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 構成可能

デフォルト[範囲] CLIENT [CLIENT, SERVER]

このパラメーターでは、クライアントが接続用のユーザー ID とパスワードの組み合わせを指定したときにトラステッド・クライアントがサーバーまたはクライアントで認証されているかどうかを指定します。このパラメーター (および *trust_allclnts*) は、*authentication* パラメーターが CLIENT に設定されている場合にのみアクティブです。ユーザー ID とパスワードが指定されていない場合、ユーザーは有効であるとクライアントは仮定し、サーバーでそれ以上の妥当性検査は行われません。

このパラメーターが CLIENT (デフォルト) に設定されている場合は、トラステッド・クライアントは、ユーザー ID とパスワードの組み合わせを示さなくても接続でき、オペレーティング・システムがユーザーをすでに認証していると想定されます。このパラメーターが SERVER に設定されている場合は、ユーザー ID およびパスワードの妥当性がサーバーで検査されます。

CLIENT の数値は 0 で、SERVER の数値は 1 です。

関連資料:

- 542 ページの『authentication - 認証タイプ』
- 555 ページの『trust_allclnts - 全クライアントのトラステッド化』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

use_sna_auth - SNA 認証の使用

構成タイプ データベース・マネージャー

適用

- ローカルとリモート・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 [Yes; No]

use_sna_auth が *yes* に設定され、*authentication* が *server* に設定されている場合は、セキュリティー・タイプが SAME または PROGRAM で SNA プロトコルを使用するサーバーへのインバウンド接続は、 SNA 層においてのみ認証され、DB2 によって認証されることはありません。

関連資料:

- 542 ページの『authentication - 認証タイプ』
- 「コマンド・リファレンス」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET DATABASE MANAGER CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

DB2 Administration Server

次のパラメーターは、DB2 Administration Server と関連があります。

- 『authentication - 認証タイプ DAS』
- 559 ページの『contact_host - 連絡先リストのロケーション』
- 560 ページの『das_codepage - DAS コード・ページ』
- 560 ページの『das_territory - DAS テリトリー』
- 561 ページの『dasadm_group - DAS 管理者権限グループ名』
- 561 ページの『db2system - DB2 サーバー・システムの名前』
- 562 ページの『discover - DAS ディスカバリー・モード』
- 563 ページの『exec_exp_task - 有効期限切れタスクの実行』
- 563 ページの『jdk_64_path - 64 ビット Software Developer's Kit for Java インストール・パス DAS』
- 564 ページの『jdk_path - Software Developer's Kit for Java インストール・パス DAS』
- 565 ページの『sched_enable - スケジューラー・モード』
- 565 ページの『sched_userid - スケジューラー・ユーザー ID』
- 566 ページの『smtp_server - SMTP サーバー』
- 567 ページの『toolscat_db - ツール・カタログ・データベース』
- 567 ページの『toolscat_inst - ツール・カタログ・データベース・インスタンス』
- 568 ページの『toolscat_schema - ツール・カタログ・データベース・スキーマ』

authentication - 認証タイプ DAS

構成タイプ DB2 Administration Server

適用 DB2 Administration Server

パラメーター・タイプ	構成可能
デフォルト[範囲]	SERVER_ENCRYPT [SERVER_ENCRYPT; KERBEROS_ENCRYPT]

このパラメーターは、ユーザーの認証が行われる方法とその場所を決定します。

認証が SERVER_ENCRYPT の場合は、ユーザー ID とパスワードがクライアントからサーバーに送信されるので、認証はサーバー上で行うことができます。ネットワークを通して送信されるパスワードは、暗号化されています。

値が KERBEROS_ENCRYPT の場合は、認証のための Kerberos セキュリティー・プロトコルを使用して、認証が Kerberos サーバーで実行されることを示します。

注: KERBEROS_ENCRYPT 認証タイプは、Windows 2000 が稼働しているサーバーでのみサポートされます。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』

contact_host - 連絡先リストのロケーション

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
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]	NULL [任意の有効なバージョン 8 DB2 Administration Server TCP/IP ホスト名]

このパラメーターは、スケジューラーおよびヘルス・モニターによる通知に使用される連絡先情報が保管されるロケーションを指定します。このロケーションは、DB2 Administration Server の TCP/IP ホスト名になるように定義されます。*contact_host* がリモート DAS に置くようにすると、複数の DB2 Administration Server 間での連絡先リストの共用がサポートされます。 *contact_host* が指定されていない場合、DAS は、連絡先情報がローカルに指定されているものと見なします。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』

das_codepage - DAS コード・ページ

構成タイプ DB2 Administration Server

適用 DB2 Administration Server

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] Null [任意の有効な DB2 コード・ページ]

このパラメーターは、DB2 Administration Server によって使用されるコード・ページを示します。このパラメーターが NULL の場合は、システムのデフォルト・コード・ページが使用されます。このパラメーターは、ローカル DB2 インスタンスのロケールと互換性のあることが必要です。互換性がない場合、DB2 Administration Server は DB2 インスタンスと通信できません。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 560 ページの『das_territory - DAS テリトリー』

das_territory - DAS テリトリー

構成タイプ DB2 Administration Server

適用 DB2 Administration Server

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] Null [任意の有効な DB2 Territory]

このパラメーターは、DB2 Administration Server によって使用される territory を示します。このパラメーターが NULL の場合は、システムのデフォルト territory が使用されます。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』

- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 560 ページの『das_codepage - DAS コード・ページ』

dasadm_group - DAS 管理者権限グループ名

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
パラメーター・タイプ	構成可能
デフォルト[範囲]	Null [任意の有効なグループ名]

DAS 管理 (DASADM) 権限は、DAS 内の最高レベルの権限です。このパラメーターは、DAS の DASADM 権限を持つグループ名を定義します。

DASADM 権限は、特定の運用環境で使用されるセキュリティー機能によって決定されます。

- Windows NT および Windows 2000 オペレーティング・システムでは、このパラメーターを 8 文字以下の名前を持つローカル・グループに設定することができ、Windows NT および Windows 2000 セキュリティー・データベースで定義されます。 このパラメーターに「NULL」を指定する場合、管理者グループのすべてのメンバーは DASADM 権限を持っています。
- UNIX ベースのシステムでは、このパラメーターの値として「NULL」を指定する場合、DASADM グループのデフォルトは、インスタンス所有者の 1 次グループです。

  値が「NULL」でない場合、DASADM グループは有効な UNIX グループ名です。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』

db2system - DB2 サーバー・システムの名前

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
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]	TCP/IP ホスト名 [任意の有効なシステム名]

このパラメーターは、ユーザーおよびデータベース管理者が DB2 サーバー・システムの識別に使用する名前を指定します。可能であれば、この名前はご使用のネットワーク内でユニークである必要があります。

この名前は、コントロール・センターのオブジェクト・ツリーのシステム・レベルで表示されるので、管理者がコントロール・センターから管理できるサーバー・システムを識別する場合に役立ちます。

構成アシスタントの「ネットワークの検索」機能を使用すると、DB2 ディスカバリーがこの名前を戻し、その結果がオブジェクト・ツリーのシステム・レベルで表示されます。 この名前は、ユーザーがアクセスしたいデータベースが入っているシステムを識別するのに役立ちます。*db2system* の値は、インストール時に次のように設定されます。

- Windows では、セットアップ・プログラムによって Windows システム用に指定されているコンピューター名に等しく設定されます。
- UNIX システムでは、 UNIX システムの TCP/IP ホスト名に等しい名前に設定されます。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 562 ページの『discover - DAS ディスカバリー・モード』

discover - DAS ディスカバリー・モード

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
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]	SEARCH [DISABLE; KNOWN; SEARCH]

Administration Server 側の観点から、この構成パラメーターは、DB2 Administration Server が始動したときに開始されるディスカバリー・モードのタイプを決定します。

- discover = SEARCH の場合、Administration Server はクライアントからの SEARCH ディスカバリー要求を処理します。SEARCH は、KNOWN ディスカバリーが提供する機能のスーパーセットを提供します。 discover = SEARCH の場合、Administration Server はクライアントからの SEARCH ディスカバリー要求と KNOWN ディスカバリー要求の両方を処理します。
- discover = KNOWN の場合、Administration Server はクライアントからの KNOWN ディスカバリー要求だけを処理します。
- discover = DISABLE の場合、Administration Server はどのタイプのディスカバリー要求も処理しません。このサーバー・システムの情報は基本的にクライアントから隠されます。

デフォルトのディスカバリー・モードは SEARCH です。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 561 ページの『db2system - DB2 サーバー・システムの名前』

exec_exp_task - 有効期限切れタスクの実行

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
パラメーター・タイプ	構成可能
デフォルト[範囲]	No [Yes; No]

このパラメーターは、過去にスケジュールされていたが、まだ実行されていないタスクを、スケジューラーが実行するかどうかを指定します。スケジューラーは、始動時にのみ、有効期限切れタスクを検出します。

たとえば、毎週土曜日に実行するようにスケジュールされたジョブがあり、スケジューラーが金曜日にオフにされ、月曜日に再始動された場合、土曜日にスケジュールされていたジョブは、過去にスケジュールされていたジョブになります。*exec_exp_task* が Yes に設定されている場合、土曜日のジョブは、スケジューラーが再始動した時点で実行されます。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 565 ページの『sched_enable - スケジューラー・モード』
- 567 ページの『toolscat_inst - ツール・カタログ・データベース・インスタンス』
- 567 ページの『toolscat_db - ツール・カタログ・データベース』
- 568 ページの『toolscat_schema - ツール・カタログ・データベース・スキーマ』
- 566 ページの『smtp_server - SMTP サーバー』
- 565 ページの『sched_userid - スケジューラー・ユーザー ID』

jdk_64_path - 64 ビット Software Developer's Kit for Java インストール・パス DAS

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
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] NULL [任意の有効なパス]

このパラメーターは、DB2 Administration Server 関数を実行するために使用する 64 ビット Software Developer's Kit (SDK) for Java がインストールされているディレクトリーを指定します。

注: これは、 32 ビットの SDK for Java を指定する *jdk_path* 構成パラメーターとは異なります。

Java インタープリターで使用される環境変数は、このパラメーターの値から計算されます。このパラメーターは、32 ビットと 64 ビットの両方のインスタンスをサポートするプラットフォームでのみ使用されます。これらのプラットフォームは 64 ビット・ハイブリッド・プラットフォームとしても知られており、 AIX、HP-UX、および Solaris オペレーティング環境が含まれます。それ以外のすべてのプラットフォームでは、*jdk_path* だけが使用されます。

このパラメーターにはデフォルト値がないため、 SDK for Java のインストール時にこのパラメーターの値を指定する必要があります。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』

jdk_path - Software Developer's Kit for Java インストール・パス DAS

構成タイプ DB2 Administration Server

適用 DB2 Administration Server

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] デフォルトの Java インストール・パス [任意の有効なパス]

このパラメーターは、DB2 Administration Server 関数を実行するために使用する Software Developer's Kit (SDK) for Java がインストールされているディレクトリーを指定します。 Java インタープリターで使用される環境変数は、このパラメーターの値から計算されます。

Windows オペレーティング・システムでは、 Java ファイル (必要な場合) は DB2 インストール時に sqllib ディレクトリー (java¥jdk 内にある) の下に置かれています。このとき、*jdk_path* 構成パラメーターは sqllib¥java¥jdk に設定されます。実際は、Java が Windows プラットフォームで DB2 によってインストールされること

はなく、Java ファイルはただ sqllib ディレクトリーに置かれるだけで、これは Java がすでにインストールされているかどうかにかかわりなく行われます。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』

sched_enable - スケジューラー・モード

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
パラメーター・タイプ	構成可能
デフォルト[範囲]	オフ [オン; オフ]

このパラメーターは、スケジューラーが Administration Server によって開始されるか、されないかを示します。スケジューラーを使用すると、タスク・センターなどのツールがタスクをスケジュールし、Administration Server で実行できます。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 567 ページの『toolscat_inst - ツール・カタログ・データベース・インスタンス』
- 567 ページの『toolscat_db - ツール・カタログ・データベース』
- 568 ページの『toolscat_schema - ツール・カタログ・データベース・スキーマ』
- 566 ページの『smtp_server - SMTP サーバー』
- 563 ページの『exec_exp_task - 有効期限切れタスクの実行』
- 565 ページの『sched_userid - スケジューラー・ユーザー ID』

sched_userid - スケジューラー・ユーザー ID

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
パラメーター・タイプ	通知
デフォルト[範囲]	NULL [任意の有効なユーザー ID]

このパラメーターは、スケジューラーがツール・カタログ・データベースに接続するのに使用するユーザー ID を指定します。このパラメーターは、ツール・カタログ・データベースが DB2 Administration Server からリモートにある場合にのみ有効です。

スケジューラーがリモート・ツール・カタログ・データベースに接続するのに使用するユーザー ID およびパスワードは、**db2admin** コマンドを使用して指定します。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 565 ページの『sched_enable - スケジューラー・モード』
- 567 ページの『toolscat_inst - ツール・カタログ・データベース・インスタンス』
- 567 ページの『toolscat_db - ツール・カタログ・データベース』
- 568 ページの『toolscat_schema - ツール・カタログ・データベース・スキーマ』
- 566 ページの『smtp_server - SMTP サーバー』
- 563 ページの『exec_exp_task - 有効期限切れタスクの実行』

smtp_server - SMTP サーバー

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
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]	Null [任意の有効な SMTP サーバー TCP/IP ホスト名]

このパラメーターは、スケジューラーおよびヘルス・モニターによって使用されます。

スケジューラーがオンの場合、このパラメーターは、スケジューラーから E メールまたはページャー通知を送信する場合に使用する、SMTP サーバーを示します。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 565 ページの『sched_enable - スケジューラー・モード』
- 567 ページの『toolscat_inst - ツール・カタログ・データベース・インスタンス』
- 567 ページの『toolscat_db - ツール・カタログ・データベース』
- 568 ページの『toolscat_schema - ツール・カタログ・データベース・スキーマ』
- 563 ページの『exec_exp_task - 有効期限切れタスクの実行』

- 565 ページの『sched_userid - スケジューラー・ユーザー ID』

toolscat_db - ツール・カタログ・データベース

構成タイプ DB2 Administration Server

適用 DB2 Administration Server

パラメーター・タイプ 構成可能

デフォルト[範囲] NULL [任意の有効なデータベース別名]

このパラメーターは、スケジューラーで使用されるツール・カタログ・データベースを示します。データベースは、*toolscat_inst* で指定されるインスタンスのデータベース・ディレクトリーにあることが必要です。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 565 ページの『sched_enable - スケジューラー・モード』
- 567 ページの『toolscat_inst - ツール・カタログ・データベース・インスタンス』
- 568 ページの『toolscat_schema - ツール・カタログ・データベース・スキーマ』
- 566 ページの『smtp_server - SMTP サーバー』
- 563 ページの『exec_exp_task - 有効期限切れタスクの実行』
- 565 ページの『sched_userid - スケジューラー・ユーザー ID』

toolscat_inst - ツール・カタログ・データベース・インスタンス

構成タイプ DB2 Administration Server

適用 DB2 Administration Server

パラメーター・タイプ 構成可能

デフォルト[範囲] NULL [任意の有効なインスタンス]

このパラメーターは、スケジューラーが toolscat_db および toolscat_schema とともに、ツール・カタログ・データベースを識別するために使用するインスタンス名を示します。ツール・カタログ・データベースには、タスク・センターおよびコントロール・センターによって作成されたタスク情報が含まれます。ツール・カタログ・データベースは、この構成パラメーターで指定されたインスタンスのデータベース・ディレクトリーにあることが必要です。データベースには、ローカル・エージェントとリモート・エージェントがあります。ツール・カタログ・データベースがローカルの場合は、インスタンスは、TCP/IP 用に構成する必要があります。データベースがリモートの場合は、データベース・ディレクトリーにカタログされるノードが TCP/IP ノードであることが必要です。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『ツール・カタログ・データベースおよび DAS スケジューラーのセットアップと構成』

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 565 ページの『sched_enable - スケジューラー・モード』
- 567 ページの『toolscat_db - ツール・カタログ・データベース』
- 568 ページの『toolscat_schema - ツール・カタログ・データベース・スキーマ』
- 566 ページの『smtp_server - SMTP サーバー』
- 563 ページの『exec_exp_task - 有効期限切れタスクの実行』
- 565 ページの『sched_userid - スケジューラー・ユーザー ID』

toolscat_schema - ツール・カタログ・データベース・スキーマ

構成タイプ	DB2 Administration Server
適用	DB2 Administration Server
パラメーター・タイプ	構成可能
デフォルト[範囲]	NULL [任意の有効なスキーマ]

このパラメーターは、スケジューラーで使用されるツール・カタログ・データベースのスキーマを示します。スキーマは、データベース内のツール・カタログ表およびビューのセットをユニークに識別する場合に使用します。

このパラメーターは、バージョン 8 コマンド行プロセッサー (CLP)からのみ更新可能です。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『RESET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 565 ページの『sched_enable - スケジューラー・モード』
- 567 ページの『toolscat_inst - ツール・カタログ・データベース・インスタンス』
- 567 ページの『toolscat_db - ツール・カタログ・データベース』
- 566 ページの『smtp_server - SMTP サーバー』
- 563 ページの『exec_exp_task - 有効期限切れタスクの実行』
- 565 ページの『sched_userid - スケジューラー・ユーザー ID』

第 4 部 付録

付録 A. DB2 レジストリー変数と環境変数

この章では、レジストリー変数と環境変数の使用法を説明し、それぞれの構文と使用法の説明と共に、カテゴリー別の変数のリストを記載しています。

DB2 レジストリー変数と環境変数

このセクションでは、始動と実行を行うときに知っている必要のある DB2® レジストリー変数と環境変数をリストします。各変数には短い説明があります。一部の変数は、ご使用の環境に適用されないことがあります。

サポートされているすべてのレジストリー変数を表示するには、以下のコマンドを実行してください。

```
db2set -lr
```

現行またはデフォルト・インスタンスの変数の値を変更するには、以下のコマンドを実行してください。

```
db2set registry_variable_name=new_value
```

DB2 環境変数 DB2INSTANCE、DB2NODE、DB2PATH、および DB2INSTPROF が DB2 プロファイル・レジストリーに保管されるかどうかは、オペレーティング・システムによって異なります。これらの環境変数を更新するには、`set` コマンドを使用します。これらの変更は、システムが次回リブートされるときに有効になります。 UNIX® プラットフォームでは、`export` コマンドを `set` コマンドの代わりに使用できます。

変更されたレジストリー変数の値は、 DB2START コマンドを実行する前に設定しなければなりません。

注: レジストリー変数が引き数としてブール値を必要とする場合、値の YES、1、および ON はすべて同等であり、値の NO、0、および OFF もすべて同等です。どの変数についても、適切な同等値のいずれかを指定することができます。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『環境変数およびプロファイル・レジストリー』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『LDAP 環境でのユーザー・ラベルでの DB2 レジストリー変数の設定』

関連資料:

- 572 ページの『汎用レジストリー変数』
- 575 ページの『システム環境変数』
- 579 ページの『通信変数』
- 583 ページの『コマンド行変数』

- 584 ページの『MPP 構成変数』
- 586 ページの『SQL コンパイラー変数』
- 591 ページの『パフォーマンス変数』
- 605 ページの『データ・リンク変数』
- 607 ページの『その他の変数』

カテゴリー別のレジストリー変数および環境変数

次のセクションでは、レジストリー変数と環境変数を、それらが制御するデータベース・マネージャーまたはデータベースの動作に応じてリストしています。

汎用レジストリー変数

表 46. 汎用レジストリー変数

変数名	オペレーティング・システム	値
説明		
DB2ACCOUNT	すべて	デフォルト = null
リモート・ホストに送信される会計ストリング。詳細は、「*DB2 Connect ユーザーズ・ガイド*」を参照してください。		
DB2BIDI	すべて	デフォルト = NO 値: YES または NO
この変数は双方向サポートを可能にします。 DB2CODEPAGE 変数は、使用するコード・ページを宣言する場合に使用します。双方向サポートについての追加詳細は、付録『各国語サポート』を参照してください。		
DB2CODEPAGE	すべて	デフォルト: オペレーティング・システムの指定どおりに言語 ID から得られます。
データベース・クライアント・アプリケーションのために DB2 に提示されるデータのコード・ページを指定します。 (設定するように) DB2 のマニュアルに明確に記述されているか、または DB2 サービスで求められない限り、DB2CODEPAGE は設定しないでください。 DB2CODEPAGE にオペレーティング・システムでサポートされていない値を設定すると、その結果は予測できなくなります。 通常、DB2 はコード・ページ情報を自動的にオペレーティング・システムから得るので、 DB2CODEPAGE を設定する必要はありません。		
DB2_COLLECT_TS_REC_INFO	すべて	デフォルト = OFF 値: YES または NO
この変数は、表スペースに影響するログ・レコードがログ・ファイルに含まれているかどうかに関わりなく、表スペースのロールフォワード時に DB2 がすべてのログ・ファイルを処理するかどうかを指定します。この表スペースに影響するいずれのログ・レコードも含まれていないログ・ファイルをスキップするには、この変数を ON に設定してください。 DB2_COLLECT_TS_REC_INFO はログ・ファイルを作成および使用する前に設定し、ログ・ファイルのスキップに必要な情報を収集できるようにします。		
DB2CONSOLECP	Windows	デフォルト = null 値: すべての有効なコード・ページ値
DB2 メッセージ・テキストを表示するためのコード・ページを指定します。指定すると、この値はオペレーティング・システムのコード・ページ設定より優先されます。		

表 46. 汎用レジストリー変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2COUNTRY	Windows	デフォルト = null 値: すべての有効な country、territory、または region コードの数値
クライアント・アプリケーションの country、territory、または region コードを指定します。指定すると、この値はオペレーティング・システムの設定より優先されます。		
DB2DBDFT	すべて	デフォルト = null
暗黙接続に使用するデータベースのデータベース別名を指定します。アプリケーションがデータベースに接続していないが SQL ステートメントが発行されている場合、デフォルト・データベースで DB2DBDFT 環境変数が定義されていれば、暗黙接続が行われます。		
DB2DBMSADDR	Windows 32 ビット・オペレーティング・システム	デフォルト= Windows NT では 0x20000000 値: 0x20000000 ～ 0xB0000000 (増分は 0x10000)
デフォルトのデータベース・マネージャー共用メモリーのアドレスを 16 進形式で指定します。共用メモリーのアドレス衝突のために *db2start* が失敗する場合、このレジストリー変数は強制的にデータベース・マネージャー・インスタンスに変更され、別のアドレスでその共用メモリーに割り当てられます。		
DB2DISCOVERYTIME	Windows オペレーティング・システム	デフォルト = 40 秒。 最低 = 20 秒
SEARCH ディスカバリーが DB2 システムを探索する時間を指定します。		
DB2_FORCE_APP_ON_MAX_LOG	すべて	デフォルト: TRUE 値: TRUE、FALSE
MAX_LOG 構成パラメーター値を超過したときの反応を指定します。 TRUE に設定すると、アプリケーションはデータベースを強制終了し、作業単位をロールバックします。 FALSE に設定すると、現行のステートメントは失敗します。アプリケーションは、作業単位内のそれ以前のステートメントで完了した作業をコミットできます。または、作業をロールバックして作業単位を取り消すこともできます。		
DB2GRAPHICUNICODESERVER	すべて	デフォルト = OFF 値: ON または OFF
このレジストリー変数は、GRAPHIC データを Unicode データベースに挿入するために作成された既存のアプリケーションを受け入れるために使用されます。このレジストリー変数を使用するのは、sqldbchar (GRAPHIC) データを、クライアントのコード・ページの代わりに Unicode で送信するアプリケーションにのみ必要です。(sqldbchar は、単一の 2 バイト文字を保持できる C および C++ の SQL データ・タイプでサポートされています。) 「ON」に設定すると、データベースに対して、GRAPHIC データが Unicode で送信されてくることを伝え、アプリケーションは GRAPHIC データを Unicode で受信することを期待します。		
DB2INCLUDE	すべて	デフォルト = 現行ディレクトリー
DB2 PREP 処理において、 SQL INCLUDE テキスト・ファイル・ステートメントの処理時に使用されるパスを指定します。これによって、INCLUDE ファイルが検出されるディレクトリーのリストが提供されます。いろいろなプリコンパイルを使用する言語において DB2INCLUDE が使用される方法については、「*アプリケーション開発ガイド*」を参照してください。		
DB2INSTDEF	Windows オペレーティング・システム	デフォルト = DB2

表 46. 汎用レジストリー変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2INSTANCE が定義されていない場合に使用される値を設定します。		
DB2INSTOWNER	Windows NT	デフォルト = null
インスタンスの初回作成時に DB2 プロファイル登録で作成されるレジストリー変数。この変数は、インスタンスを所有するマシンの名前に設定されます。		
DB2_LIC_STAT_SIZE	すべて	デフォルト = null 範囲: 0 ～ 32 767
このレジストリー変数は、システムのライセンス統計が入っているファイルの最大サイズ (MB 単位) を決定します。値がゼロの場合、ライセンスの統計収集はオフになります。認識または定義されていない場合、この変数はデフォルト (無制限) に設定されます。統計は、ライセンス・センターを使って表示されます。		
DB2LOCALE	すべて	デフォルト: NO 値: YES または NO
DB2 を呼び出した後にデフォルトの「C」ロケールがデフォルトの「C」ロケールにリストアされるかどうか、および DB2 機能を呼び出した後にプロセス・ロケールをリストアして元の「C」に戻すかどうかを指定します。元のロケールが「C」ではない場合、このレジストリー変数は無視されます。		
DB2NBDISCOVERRCVBUFS	すべて	デフォルト = 16 バッファー 最小値 = 16 バッファー
この変数は NetBIOS SEARCH ディスカバリーで使用されます。変数は、クライアントが同時に受信可能なディスカバリー応答数を指定します。クライアントがこの変数で指定した数よりも多くの応答を同時に受け取ると、超過した応答は NetBIOS 層によって廃棄されます。デフォルトは、16 の NetBIOS 受信バッファーです。デフォルト値よりも小さい値を選択すると、デフォルトが使用されます。		
DB2_OBJECT_TABLE_ENTRIES	すべて	デフォルト = 0 値: 0-50000
1 つの表スペースに入ることが予想されるオブジェクトの数を指定します。 DMS 表スペースに大量のオブジェクト (たとえば、1000 以上) が作成されることが分かっている場合は、表スペースを作成する前に、このレジストリー変数を適切な数値に設定します。これにより、表スペースの作成時のオブジェクト・メタデータ用に連続するストレージが予約されます。連続するストレージを予約すると、メタデータ内の項目を更新する操作 (たとえば、CREATE INDEX、IMPORT REPLACE) がオンライン・バックアップによってブロックされる可能性は減ります。また、表スペースの開始時にメタデータが保管されるので、表スペースのサイズ変更が容易になります。 表スペースの初期サイズが十分でないために連続するストレージを予約できない場合は、追加スペースの予約を指定せずに表スペース作成が続行します。		
DB2OPTIONS	すべて	デフォルト = null
コマンド行プロセッサー・オプションを設定します。		
DB2TERRITORY	すべて	デフォルト: オペレーティング・システムの指定どおりに言語 ID から得られます。
クライアント・アプリケーションの region および territory コードを指定します。これは、日付と時刻の形式に影響します。		
DB2_VIEW_REOPT_VALUES	すべて	デフォルト = NO 値: YES、NO

表 46. 汎用レジストリー変数 (続き)

変数名	オペレーティング・システム	値
説明		
この変数を使用すると、すべてのユーザーは、ステートメントを EXPLAIN するときに、再最適化された SQL ステートメントのキャッシュされた値を EXPLAIN_PREDICATE 表に保管できます。この変数を NO に設定すると、DBADM だけがその値を EXPLAIN_PREDICATE 表に保管できます。		

関連概念:

- 571 ページの『DB2 レジストリー変数と環境変数』

システム環境変数

表 47. システム環境変数

変数名	オペレーティング・システム	値
説明		
DB2CONNECT_IN_APP_PROCESS	すべて	デフォルト = YES 値: YES または NO
この変数を NO に設定すると、 DB2 Connect Enterprise Edition マシン上のローカル DB2 Connect クライアントはエージェント内で強制的に実行されます。 エージェント内で実行する利点としては、ローカル・クライアントをモニターできることと、ローカル・クライアントが SYSPLEX サポートを使用できることがあります。		
DB2DOMAINLIST	Windows NT サーバーのみ	デフォルト = null 値: コンマ (「,」) 区切りの Windows NT ドメイン・ネームのリスト
1 つ以上の Windows NT ドメインを定義します。リストには、要求側のユーザー ID を認証するドメインが定義されます。これらのドメインに属しているユーザーの接続要求のみが受け入れられます。 この変数は、CLIENT 認証がデータベース・マネージャー構成で設定されている場合にのみ有効であり、Windows NT ドメイン環境で Windows NT デスクトップからのシングル・サインオンが要求されている場合に必要です。 このレジストリー変数は、 DB2 Universal Database バージョン 7.1 (またはそれ以降) が稼働する DB2 サーバーおよびクライアントの純粋な Windows NT ドメイン環境下でのみ使用します。		
DB2ENVLIST	UNIX	デフォルト: null
ストアード・プロシージャーまたはユーザー定義関数の特定の変数名をリストします。デフォルトでは、**db2start** コマンドは、接頭部が **DB2** または **db2** になっているユーザー環境変数を除いて、すべてのユーザー環境変数をフィルターに掛けて除去します。特定の環境変数をストアード・プロシージャーまたはユーザー定義関数のどちらかに渡さなければならない場合、 DB2ENVLIST 環境変数にその変数名をリストできます。その場合、各変数を 1 つまたは複数のスペースで区切ります。		
DB2INSTANCE	すべて	デフォルト = DB2INSTDEF (Windows 32 ビット・オペレーティング・システムの場合)
デフォルト解釈によりアクティブになるインスタンスを指定するために使用される環境変数。 UNIX では、ユーザーは DB2INSTANCE に値を指定する必要があります。		
DB2INSTPROF	Windows オペレーティング・システム	デフォルト: null

表 47. システム環境変数 (続き)

変数名	オペレーティング・システム	値
説明		
Windows オペレーティング・システムにおいて、インスタンス・ディレクトリーが DB2PATH 以外の場所にある場合のその位置を示すために使用される環境変数。		
DB2LIBPATH	UNIX	デフォルト: null
DB2 はその共有ライブラリー・パスを構成します。 PATH をエンジンのライブラリー・パスに追加したい場合は、(たとえば、AIX ではユーザー定義関数には LIBPATH 内に特定項目が必要) DB2LIBPATH を設定する必要があります。実際の DB2LIBPATH の値は、DB2 構成共有ライブラリー・パスの最後に付加されます。		
DB2NODE	すべて	デフォルト: null 値: 1 ～ 999
接続する DB2 Enterprise Server Edition データベース・パーティション・サーバーのターゲット論理ノードを指定します。この変数を指定しない場合、ターゲット論理ノードはデフォルトとして、マシン上のポート 0 に定義された論理ノードに設定されます。		
DB2_PARALLEL_IO	すべて	デフォルト: null 値: • *[:ディスクの数] - すべての表スペースが並列入出力を使用可能にすることを意味します。コロンの後に値またはシンボルを指定し、すべての表スペースについて、コンテナーごとのディスクの数のデフォルトを定義します。ディスクの数を指定しないと、6 (RAID-5 装置の値) がデフォルトとして使用されます。 • TablespaceID[:ディスクの数],... - コンマで区切られた定義済み表スペースのリスト。この表スペースについて、コンテナーごとのディスクの数を定義するために、各表スペース ID の後にコロンを追加して値を指定します。この値は、数値または下の表に記述されているいずれかのシンボルにすることができます。 • 上記 2 つの値タイプの組み合わせ (コンマで値を区切る)。

表 47. システム環境変数 (続き)

変数名	オペレーティング・システム	値
説明		
このレジストリー変数は、DB2 が表スペースの入出力並列処理を計算する方法を変更するために使用します。入出力並列処理を有効にすると (複数のコンテナーを使用することにより暗黙的に、または DB2_PARALLEL_IO を設定することにより明示的に)、その結果としてプリフェッチ要求の正しい数を発行されます。各プリフェッチ要求は、ページのエクステントの要求です。 このレジストリー変数を設定しない場合、表スペースの並列処理の多重度は表スペースのコンテナーの数になります。たとえば、DB2_PARALLEL_IO が NULL に設定されており、表スペースに 4 つのコンテナーがある場合、 4 つのエクステント・サイズのプリフェッチ要求が発行されることになります。 このレジストリー変数を設定する場合、表スペースの並列処理の多重度は表スペースのプリフェッチ・サイズとエクステント・サイズの比率になります。たとえば、プリフェッチ・サイズが 160 ページでエクステント・サイズが 32 ページの表スペースに DB2_PARALLEL_IO を設定した場合、 5 つのエクステント・サイズ・プリフェッチ要求が発行されることになります。ワイルドカード「*」文字を使用することによって、すべての表スペースについて、同様の方法で入出力並列処理を計算するよう DB2 に指示できます。 各 DB2 コンテナーの下の物理スピンドルの除去をサポートする入出力サブシステムでは (たとえば、RAID 装置が付加されているもの)、表スペースのプリフェッチ・サイズを選択するときに、各 DB2 コンテナーの下のディスクの数を考慮に入れる必要があります。プリフェッチ・サイズは次の式を基にして計算します。 `Prefetch size = (number of containers) * (number of disks per container) * extent size` 表スペースのプリフェッチ・サイズが AUTOMATIC である場合、 DB2 は上記の式を使用して表スペースのプリフェッチ・サイズを自動的に計算します。 DB2_PARALLEL_IO レジストリー変数を使用して、各コンテナーのディスク数を DB2 に通知できます。たとえば、DB2_PARALLEL_IO=″1:4″ に設定し、表スペース 1 に 3 つのコンテナーがあり、エクステント・サイズ 32、プリフェッチ・サイズ AUTOMATIC である場合は、プリフェッチ・サイズは 3 * 4 * 32 = 384 ページと計算されます。この表スペースの入出力並列処理は 384 / 32 で 12 です。表スペースのプリフェッチ・サイズが AUTOMATIC ではない場合、各コンテナーのディスク数に関するこの情報は使用されません。 DB2_PARALLEL_IO に指定した表スペースは、レジストリー変数に他の指定がされていなければ、デフォルトで各コンテナーのディスク数を 6 と想定します。たとえば、DB2_PARALLEL_IO=*,1:3 の場合、表スペース 1 には各コンテナーのディスク数として 3 が使用されますが、それ以外のすべての表スペースは各コンテナーのディスク数に 6 を使用します。		
DB2PATH	Windows オペレーティング・システム	デフォルト: (オペレーティング・システムによって異なる)
この製品が Windows 32 ビットのオペレーティング・システム上に、インストールされるディレクトリーを指定するために使用される環境変数。		
DB2PROCESSORS	Windows オペレーティング・システム	デフォルト: null 値: 0-n-1 (ここで n はプロセッサーの数)
特定の db2syscs プロセスに対してプロセス Affinity Mask を設定します。複数の論理ノードを実行している環境では、論理ノードを 1 つのプロセッサーまたは複数のプロセッサーの集合に関連付けるためにこの変数が使用されます。 この変数が指定されると、DB2 は SetProcessAffinityMask() API を発行します。指定されない場合は、db2syscs プロセスがマシン上のすべてのプロセッサーに関連付けられます。		

表 47. システム環境変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2_USE_PAGE_CONTAINER_TAG	すべて	デフォルト: null 値: ON、null
デフォルトでは、DB2 はコンテナー・タグを各 DMS コンテナー (それがファイルでも装置でも) の最初のエクステントに保管します。コンテナー・タグは、コンテナーのメタデータです。 DB2 バージョン 8.1 より前には、コンテナー・タグは単一のページに保管されたので、コンテナー内でより少ないスペースだけを必要としました。継続してコンテナー・タグを単一のページに保管するには、 DB2_USE_PAGE_CONTAINER_TAG を ON に設定します。 しかし、コンテナーに RAID 装置を使用する場合にこのレジストリー変数を ON に設定すると、入出力パフォーマンスは低下することがあります。 RAID 装置ではエクステント・サイズが RAID ストライプ・サイズと等しいかその倍数の表スペースを作成するので、 DB2_USE_PAGE_CONTAINER_TAG を ON に設定すると、エクステントが RAID ストライプときれいに一致しなくなります。その結果、入出力要求は最適な場合よりも多くの物理ディスクにアクセスしなければならないことがあります。このレジストリー変数を使用可能にしないことを強くお勧めします。 このレジストリー変数に対する変更を有効にするには、 DB2STOP コマンドを出して DB2START コマンドを入力します。		
DB2_CLPHISTSIZE	すべて	デフォルト: 20 **注:** このレジストリー変数は、インストール時にデフォルト値に設定されません。このレジストリー変数が設定されていない場合、または有効な範囲外の値に設定されている場合、この変数を利用するコードがデフォルト値の 20 を使用します。値: 1 以上 500 以下
この変数は、CLP 対話式セッションでコマンド履歴に保管されるコマンドの数を決定します。コマンド履歴はメモリー内で保持されるため、この変数の値を高くしすぎると、セッション内で実行されるコマンドの数および長さによってはパフォーマンスが低下する場合があります。		
DB2_CLP_EDITOR	すべて	デフォルト: Windows プラットフォーム: 「Notepad」 UNIX: 「vi」 **注:** このレジストリーは、インストール時にデフォルト値に設定されません。このレジストリー変数が設定されていない場合、この変数を利用するコードがデフォルト値を使用します。値: オペレーティング・システム・パスにある任意の有効なエディター
この変数は、**EDIT** コマンドの実行時に使用するエディターを決定します。 CLP 対話式セッションから **EDIT** コマンドを実行すると、ユーザー指定コマンドでプリロード済みのエディターが起動し、編集と実行が可能になります。		

関連概念:

- 571 ページの『DB2 レジストリー変数と環境変数』

通信変数

表 48. 通信変数

変数名	オペレーティング・システム	値
説明		
DB2CHECKCLIENTINTERVAL	全、ただしサーバーのみ	Default=50 値: ゼロより大きい数値。
APPC および TCP/IP クライアント接続検査の頻度を指定します。照会完了まで待つのではなく、クライアント終了を早期に検出できるようにします。この変数が 0 に設定されている場合、検査は実行されません。 低い値ほどチェックが頻繁であることを表します。低い頻度の場合は 100、中位の頻度の場合は 50、高い頻度の場合は 10 を目安にしてください。データベース要求の実行中に、クライアントの状況を頻繁にチェックすればするほど、照会が完了するまでの時間が長くなります。 DB2 のワークロードが重い (内部要求が多い) 場合、DB2CHECKCLIENTINTERVAL に低い値を設定すると、ワークロードが軽く、DB2 がほとんどの時間待機している状況よりも、パフォーマンスに重大な影響があります。 DB2 UDB では、バージョン 8.1.4 で DB2CHECKCLIENTINTERVAL のデフォルト値は 50 です。バージョン 8.1.4 以前は、デフォルト値が 0 です。		
DB2COMM	全、ただしサーバーのみ	デフォルト = null 値: APPC、 IPXSPX、 NETBIOS、 NPIPE、 TCPIP の任意の組み合わせ
データベース・マネージャーを開始したときに開始されるコミュニケーション・マネージャーを指定します。これを指定しないと、サーバーではどの DB2 コミュニケーション・マネージャーも開始されません。		
DB2_FORCE_NLS_CACHE	AIX、HP_UX、Solaris オペレーティング環境	デフォルト = false 値: TRUE または FALSE
マルチスレッド・アプリケーションにおいてロック競合が起きないようにするために使用します。レジストリー変数が「TRUE」の場合、スレッドが初めてコード・ページとテリトリー・コードの情報にアクセスする際にそれらの情報が保管されます。その時点以降、この情報を要求する他のスレッドにこのキャッシュ情報が使用されます。したがってロック競合は除かれ、特定の状態下でパフォーマンスが向上することになります。アプリケーションにより接続間のロケール設定が変更される場合、この設定値は使用できません。マルチスレッド・アプリケーションでロケール設定を変更するのは「スレッド・セーフ」ではないので普通はこの変更は行われません。したがってこのような状態は考慮する必要がないと思われます。		
DB2JD_PORT_NUMBER	すべて	デフォルト = 6789 値: 1-65535
この設定により db2jd のデフォルト・ポート番号をオーバーライドできます。このレジストリー変数を設定すると、パラメーターを指定せずに db2jstrt を発行したときに、 db2jd はこの設定を db2jd の listen ポートとして使用することを試みます。このレジストリー変数を設定せず、ポート・パラメーターを指定しない場合、 db2jd はデフォルト・ポート 6789 で開始します。		
DB2NBADAPTERS	Windows	デフォルト = 0 範囲: 0 ～ 15。 複数の値を指定するときは、コンマで区切ります
DB2 NetBIOS LAN 通信に使用するローカル・アダプターを指定するために使用されます。各ローカル・アダプターをその論理アダプター番号で指定します。		

表 48. 通信変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2NBCHECKUPTIME	Windows サーバーのみ	デフォルト = 1 分 値: 1 ～ 720
NetBIOS プロトコルの検査手順の各呼び出しの間の時間間隔を指定します。検査時間は分の単位で指定します。 より小さい値を指定すると、 NetBIOS プロトコル検査はより頻繁に実行され、予期しない時点でエージェント/セッションが終了すると、残されているメモリーやその他のシステム・リソースは解放されます。		
DB2NBINTRLISTENS	Windows サーバーのみ	デフォルト = 1 値: 1 ～ 10 複数の値を指定するときは、コンマで区切ります
リモート・クライアントの割り込みに備えて非同期に出される NetBIOS 送信 listen コマンド (NCB) の数を指定します。 この機能は、「割り込みがアクティブ」な環境のために提供されており、このおかげでサーバーが他のリモート割り込みを扱っているときに、リモート・クライアントからの割り込み呼び出しが接続を確立できます。 DB2NBINTRLISTENS をより小さい値に設定すると、 NetBIOS セッションと NCB がサーバーで節約されます。 しかし、クライアント割り込みが多い環境では、 DB2NBINTRLISTENS をより大きな値に設定し、クライアントの割り込みに応答できるようにする必要があります。 **注:** この値は、それが指定されている位置に意味があります。すなわち、値は、DB2NBADAPTERS のそれぞれの対応位置の値に関連します。		
DB2NBRECVBUFFSIZE	Windows サーバーのみ	デフォルト = 4096 バイト 範囲: 4096 ～ 65536
DB2 NetBIOS プロトコル受信バッファーのサイズを指定します。これらのバッファーは、NetBIOS 受信 NCB に割り当てられます。より小さい値にするとサーバー・メモリーが節約され、一方、クライアント・データの転送が大きいときはより大きい値が必要になります。		
DB2NBBRECVNCBS	Windows サーバーのみ	デフォルト = 10 範囲: 1 ～ 99
操作中にサーバーが発行し、保守する NetBIOS ″receive_any″ コマンド (NCB) 数を指定します。この値は、サーバーが接続されているリモート・クライアント数に応じて調整されます。 より小さい値にすると、サーバー・リソースが節約されます。 **注:** DB2NBBRECVNCBS によって、使用中の各アダプターそれぞれにユニークな受信 NCB 値を指定することができます。すなわち、値が指定されている位置が意味を持ち、各値は、DB2NBADAPTERS のそれぞれの対応位置の値に関連します。		
DB2NBRESOURCES	Windows サーバーのみ	デフォルト = null
複数コンテキスト環境で DB2 使用のために割り振る NetBIOS リソース数を指定します。この変数は複数コンテキスト・クライアント操作の場合に限られます。		
DB2NBSENDNCBS	Windows サーバーのみ	デフォルト = 6 範囲: 1 ～ 720

表 48. 通信変数 (続き)

変数名	オペレーティング・システム	値
説明		
サーバーが使用するために予約する送信 NetBIOS コマンド (NCB) 数を指定します。 この値は、サーバーが接続されているリモート・クライアント数に応じて調整できます。 DB2NBSENDNCBS をより小さい値に設定すると、サーバー・リソースが節約されます。 ただし、他のすべての送信コマンドが使用中になっているときにサーバーがリモート・クライアントへの送信を待機しないようにするには、より大きい値を設定する必要があります。		
DB2NBSESSIONS	Windows サーバーのみ	デフォルト = null 範囲: 5 ～ 254
DB2 が DB2 使用のために予約しておくように要求するセッション数を指定します。 DB2NBSESSIONS の値は、DB2NBADAPTERS で指定される各アダプターに固有のセッションを要求するように設定できます。 **注:** この値は、それが指定されている位置に意味があります。すなわち、値は、DB2NBADAPTERS のそれぞれの対応位置の値に関連します。		
DB2NBXTRANCBS	Windows サーバーのみ	デフォルト = アダプター当たり 5 範囲: 5 ～ 254
db2start コマンドが出されたときにサーバーが予約する必要のある「余分の」NetBIOS コマンド (NCB) 数を指定します。 DB2NBXTRANCBS の値は、 DB2NBADAPTERS で指定される各アダプターに固有のセッションを要求するように設定できます。		
DB2RETRY	Windows	デフォルト = 0 範囲: 0～20 000
DB2 で APPC リスナーの再始動が試行される回数。サーバー/ゲートウェイで SNA サブシステムがダウンした場合、このプロファイル変数と DB2RETRYTIME を一緒に使用すると、 APPC リスナーを自動的に再始動できます。その際他のプロトコルを使用しているクライアント通信は中断しません。 このシナリオの場合、 APPC クライアント通信を復元するために DB2 を停止して再始動する必要はなくなります。		
DB2RETRYTIME	Windows	デフォルト = 1 分 範囲: 0～7 200 分
DB2 で APPC リスナーを開始するための連続再試行が行われる間隔を示す分数。増分は 1 分。サーバー/ゲートウェイで SNA サブシステムがダウンした場合、このプロファイル変数と DB2RETRY を一緒に使用すると、 APPC リスナーを自動的に再始動できます。その際他のプロトコルを使用しているクライアント通信は中断しません。 このシナリオの場合、 APPC クライアント通信を復元するために DB2 を停止して再始動する必要はなくなります。		
DB2SERVICETPINSTANCE	Windows、AIX、および Solaris オペレーティング環境	デフォルト = null
以下が原因で生じる問題を解決するために使用されます。 • 同じマシンで複数のインスタンスを実行している。 • 同じマシンでバージョン 6 またはバージョン 7 のインスタンスを実行しており、同じ TP 名を登録しようとしています。 **db2start** コマンドが呼び出されると、指定されたインスタンスが以下の TP 名の APPC listener を開始します。 • DB2DRDA • x'07'6DB		
DB2SORCVBUF	すべて	デフォルト = 65536
Windows オペレーティング・システムでの TCP/IP 受信バッファーの値を指定します。		

表 48. 通信変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2SOSNDBUF	すべて	デフォルト = 65536
Windows NT オペレーティング・システムでの TCP/IP 送信バッファーの値を指定します。		
DB2SYSPLEX_SERVER	Windows NT、および UNIX	デフォルト = null
SYSPLEX 機能を DB2 for OS/390 and z/OS に接続した際に使用できるかどうかを指定します。 このレジストリー変数を設定しなかったり非ゼロ値に設定したりすると、活用は使用可能です。このレジストリー変数をゼロ (0) に設定すると、機能は使用不可です。ゼロに設定すると、DCS データベース・カタログ項目の指定内容にかかわらず、ゲートウェイで SYSPLEX 機能は使用できません。詳細については、コマンド行プロセッサーの **CATALOG DCS DATABASE** コマンドを参照してください。		
DB2TCPCONNMGRS	すべて	デフォルト = 1 (シリアル・マシンの場合); 最大で 8 つの接続マネージャーに切り上げられたプロセッサー数の平方根 (SMP マシンの場合)。 値: 1 ～ 8
このレジストリー変数が設定されていない場合、デフォルトの数の接続マネージャーが作成されます。このレジストリー変数が設定されていれば、ここで割り当てた値がデフォルト値を上書きします。指定された数 (最大 8) の TCP/IP 接続マネージャーが作成されます。 1 より小さい値が指定された場合、 DB2TCPCONNMGRS には値 1 が設定され、値が範囲外であることを示す警告がログに記録されます。 8 より大きい値が指定された場合、 DB2TCPCONNMGRS には値 8 が設定され、値が範囲外であることを示す警告がログに記録されます。 1 から 8 の値を指定した場合、その値がそのまま使用されます。複数の接続マネージャーが作成されれば、複数のクライアント接続を同時に受け取る場合の接続スループットが向上するはずです。ユーザーが SMP マシン上で実行している場合、または DB2TCPCONNMGRS レジストリー変数を変更した場合、追加の TCP/IP 接続マネージャー・プロセス (UNIX の場合) またはスレッド (Windows オペレーティング・システムの場合) がある場合があります。追加のプロセスまたはスレッドでは、追加のストレージを必要とします。 **注:** 接続マネージャーの数を 1 に設定すると、多くのユーザーを持つ、または頻繁に接続と切断が繰り返される、あるいはその両方が原因でシステムのリモート接続上でパフォーマンスの低下が生じます。		
DB2_VI_ENABLE	Windows NT	デフォルト = OFF 値: ON または OFF
仮想インターフェース (VI) 体系通信プロトコルを使用するかどうかを指定します。このレジストリー変数が「ON」の場合、FCM では VI を使用してノード間通信を行います。このレジストリー変数が「OFF」の場合、FCM では TCP/IP を使用してノード間通信を行います。 **注:** このレジストリー変数は、インスタンス中のすべてのデータベース・パーティションで同じでなければなりません。		
DB2_VI_VIPL	Windows NT	デフォルト = `vipl.dll`
DB2 で使用される仮想インターフェース・プロバイダー・ライブラリー (VIPL) の名前を指定します。このライブラリーを正常にロードするには、このレジストリー変数で使用されるライブラリー名は PATH ユーザー環境変数にもなければなりません。 現在サポートされているすべてのインプリメンテーションでは、同じライブラリー名を使用します。		
DB2_VI_DEVICE	Windows NT	デフォルト = null 値: `nic0 / VINIC`

表 48. 通信変数 (続き)

変数名	オペレーティング・システム	値
説明		
装置のシンボリック名、もしくはネットワーク・インターフェース・カード (NIC) に関連した仮想インターフェース・プロバイダー・インスタンスを指定します。独立ハードウェア・ベンダー (IHV) はそれぞれ独自の NIC を作成しています。 Windows NT マシンで使用できる NIC は 1 つだけです。同一マシン上の複数の論理ノードは同一の NIC を共用します。装置シンボリック名「VINIC」は、Synfinity 相互接続で使用する場合、大文字かつ単一である必要があります。他の現在サポートされている、すべてのインプリメンテーションは、装置記号名として「nic0」を使用します。		

関連概念:

- 571 ページの『DB2 レジストリー変数と環境変数』

コマンド行変数

表 49. コマンド行変数

変数名	オペレーティング・システム	値
説明		
DB2BQTIME	すべて	デフォルト = 1 秒 最大値: 1 秒
コマンド行プロセッサー・フロントエンドが、バックエンド・プロセスがアクティブでフロントエンドへの接続を設定しているかどうかを検査する前にスリープする時間を指定します。		
DB2BQTRY	すべて	デフォルト = 60 再試行 最小値: 0 再試行
コマンド行プロセッサー・フロントエンド・プロセスが、バックエンド・プロセスがすでにアクティブであるかどうかを判別しようとする回数を指定します。 これは、DB2BQTIME と一緒に働きます。		
DB2_CLPPROMPT	すべて	デフォルト = なし 定義されない場合は、 “db2 =>” がデフォルト CLP 対話式プロンプトとして使用されます。 可能な値: %i、%d、%ia、%da、または %n のトークンを含み、 100 未満の長さの任意の文字列。デフォルト CLP 対話式プロンプト (db2 =>) を明示的に変更する必要がない限り、この変数をユーザーが設定する必要はありません。

表 49. コマンド行変数 (続き)

変数名	オペレーティング・システム	値
説明		
このレジストリー変数を使用すると、コマンド行プロセッサー (CLP) 対話式モードで使用されるプロンプトをユーザーが定義できます。変数は、%i、%d、%ia、%da、または %n のオプションのトークンを含み、 100 未満の長さの文字列に設定できます。 CLP 対話式モードで実行する場合、使用するプロンプトは DB2_CLPPROMPT レジストリー変数に指定された文字列を取り、トークン %i、%d、%ia、%da、または %n をそれぞれ、現行接続インスタンスのローカル別名、現行データベース接続のローカル別名、現行接続インスタンスの許可 ID、現行データベース接続の許可 ID、および改行 (すなわち復帰) で置き換えることによって構成されます。 **注:** 1. DB2_CLPPROMPT レジストリー変数が CLP 対話モード内で変更された場合、 CLP 対話モードが閉じて再オープンされるまで、 DB2_CLPPROMPT の新しい値は有効になりません。 2. インスタンス・アタッチが存在していない場合は、 %ia は空ストリングで置き換えられ、 %i は DB2INSTANCE レジストリー変数の値で置き換えられます。 Windows プラットフォームの場合のみ、DB2INSTANCE 変数が設定されていない場合、 %i が DB2INSTDEF レジストリー変数の値によって置き換えられます。これらの変数のいずれも設定されていない場合は、 %i が空ストリングで置き換えられます。 3. データベース接続が存在していない場合、 %da は空ストリングで置き換えられ、 %d は DB2DBDFT レジストリー変数の値で置き換えられます。 DB2DBDFT 変数が設定されていない場合、 %d は空ストリングで置き換えられます。 4. 対話式入力プロンプトは常に、許可 ID、データベース名、およびインスタンス名を大文字で表します。		
DB2IQTIME	すべて	デフォルト = 5 秒 最小値: 1 秒
コマンド行プロセッサー・バックエンド・プロセスが、フロントエンド・プロセスがコマンドを渡すのを入力キューで待機する時間を指定します。		
DB2RQTIME	すべて	デフォルト = 5 秒 最小値: 1 秒
コマンド行プロセッサー・バックエンド・プロセスが、フロントエンド・プロセスからの要求を待機する時間を指定します。		

関連概念:

- 571 ページの『DB2 レジストリー変数と環境変数』

MPP 構成変数

表 50. MPP 構成変数

変数名	オペレーティング・システム	値
DB2ATLD_PWFILE	DB2 UDB ESE (AIX、Solaris オペレーティング環境、および Windows NT の場合)	デフォルト = null 値: ファイル・パス式

表 50. MPP 構成変数 (続き)

変数名	オペレーティング・システム	値
オートローダー認証の際に使用されるパスワードを含むファイルへのパスを指定します。設定しないと、オートローダーは、その構成ファイルからパスワードを取り出すか、対話式で入力を促すプロンプトを出します。この変数を使用すると、パスワード・セキュリティーの問題が扱われ、認証情報からのオートローダー構成情報の分離を許可します。 このレジストリー変数は必要なくなったが、以前のバージョンとの互換性のために保持されています。		
DB2CHGPWD_EEE	DB2 UDB ESE (AIX および Windows NT の場合)	デフォルト = null 値: YES または NO
AIX または Windows NT ESE システムでのパスワード変更を他のユーザーに許可するかどうかを指定します。すべてのパーティションまたはノードのパスワードは、 Windows NT ドメイン・コントローラ (Windows NT の場合) または NIS (AIX の場合) を使って集中保守する必要があります。集中保守しないと、すべてのパーティションまたはノード間でパスワードが一致しなくなるおそれがあります。その結果、ユーザーが変更を加えるために接続するデータベース・パーティションでのみパスワードが変更される可能性があります。このグローバル・レジストリー変数の変更は、ルート・ディレクトリーおよび DAS インスタンスで行わなければなりません。 この変数は古い *db2atld* ユーティリティーを新しい LOAD ユーティリティーの代わりに使用する場合にのみ必要となります。		
DB2_FORCE_FCM_BP	AIX	デフォルト = No 値: Yes または No
このレジストリー変数は、複数の論理パーティションを使用する DB2 UDB ESE for AIX に適用できます。 DB2START を発行すると、 DB2 により FCM バッファーがデータベース・グローバル・メモリーから、または使用可能なグローバル・メモリーが足りない場合は分離した共用メモリー・セグメントから割り当てられます。これらのバッファーは、同じ物理マシン上にあるそのインスタンスのすべての FCM デーモンによって使用されます。割り振られるメモリーの種類は、*fcm_num_buffers* データベース・マネージャー構成パラメーターによって指定した作成される FCM バッファーの数に大きく依存します。 DB2_FORCE_FCM_BP 変数を Yes に設定した場合、 FCM バッファーは常に分離したメモリー・セグメント内に作成されて、同一ノード上の別々の論理パーティションの FCM デーモン間で共用メモリーを介して通信が行われます。グローバル・メモリーに作成される場合、同一ノード上の FCM デーモン間の通信は UNIX ソケットを介して行われます。共用メモリーを介する通信はより高速だが、特にデータベース・バッファー・プールなど他の用途に使用可能な共用メモリー・セグメントの数が 1 つ少なくなります。このように、DB2_FORCE_FCM_BP レジストリー変数を使用可能にすると、データベース・バッファー・プールの最大サイズは小さくなります。		
DB2_NUM_FAILOVER_NODES	すべて	デフォルト: 2 値: 0 ～ 論理ノードの数
高可用性環境でフェイルオーバー・ノードとして使用できるノードの数を指定します。高可用性環境であれば、ノードに障害が発生したときに、そのノードを別のホストで 2 番目の論理ノードとして再始動できます。この変数で使用する数から、フェイルオーバー・ノードの FCM リソース用に予約されるメモリーの量が判別されます。 たとえば、 1 と 2 という 2 つの論理ノードがあるホスト A と、 3 と 4 という 2 つの論理ノードがあるホスト B を想定します。 DB2_NUM_FAILOVER_NODES が 2 に設定されているものと想定します。 DB2START 中に、ホスト A とホスト B は FCM が必要とするだけのメモリーを予約するため、最大 4 つの論理ノードを管理することができます。そして、一方のホストで障害が発生した場合、もう一方のホストで、障害が発生したホスト用の論理ノードを再始動することができます。		

表 50. MPP 構成変数 (続き)

変数名	オペレーティング・システム	値
DB2_PARTITIONEDLOAD__DEFAULT	サポートされる ESE プラットフォームのすべて	デフォルト : YES 値の範囲 : YES/NO
DB2_PARTITIONEDLOAD_DEFAULT レジストリー変数によって、 ESE に特定の Load オプションを指定しない場合、ユーザーは ESE 環境内の Load ユーティリティーのデフォルト動作を変更できます。デフォルト値は YES であり、これは ESE 環境で ESE に特定の Load オプションを指定しない場合に、ロードがターゲット表が定義されているすべてのパーティション上で試行されることを指定します。 値が NO であると、ロードは Load ユーティリティーが現在接続されているパーティション上だけで試行されます。		
DB2PORTRANGE	Windows NT	値: nnnn:nnnn
この値は、FCM によって使用される TCP/IP ポート範囲に設定されるので、別のマシン上に作成される追加のパーティションも同じポート範囲になります。		

関連概念:

- 571 ページの『DB2 レジストリー変数と環境変数』

SQL コンパイラー変数

表 51. SQL コンパイラー変数

変数名	オペレーティング・システム	値
説明		
DB2_ANTIJOIN	すべて	デフォルト=NO (ESE 環境) デフォルト=YES (ESE 環境以外) 値: YES、NO または EXTEND
DB2 Universal Database ESE 環境の場合に「YES」が指定されていると、オプティマイザーは、機会があるたびに、「NOT EXISTS」副照会を DB2 がより効率的に処理できるアンチ結合に変換します。非 ESE 環境 で「No」が指定されていると、オプティマイザーは「NOT EXISTS」副照会をアンチ結合にほとんど変換しません。 ESE と NON-ESE の両方の環境で、EXTEND を指定すると、オプティマイザーは「NOT IN」と「NOT EXISTS」の両方の副照会をアンチ結合にトランスフォームする機会を探ります。		
DB2_CORRELATED_ PREDICATES	すべて	デフォルト = Yes 値: Yes または No
この変数のデフォルトは「Yes」です。結合内の相関列上にユニーク索引があり、このレジストリー変数が「Yes」の場合、このオプティマイザーは、結合述語の相関を検出し、補正しようとします。このレジストリー変数が「Yes」の場合、オプティマイザーは相関しているケースを検出するためユニーク索引統計の KEYCARD 情報を使用し、結合された、関連する述部の選び方を動的に調整します。こうすることで、結合するサイズやコスト面で、より正確な見積もりを立てることができます。単純な等価述部の相関で調整が行われます。たとえば、C1 および C2 に索引が存在する場合の `WHERE C1=5 AND C2=10` など。索引はユニークである必要はないが、等価述部列は索引内のすべての列を網羅していなければなりません。		
DB2_HASH_JOIN	すべて	デフォルト = YES 値: YES または NO

表 51. SQL コンパイラー変数 (続き)

変数名	オペレーティング・システム	値
説明		
アクセス・プランのコンパイル時に可能な結合方法としてハッシュ結合を指定します。		
DB2_INLIST_TO_NLJN	すべて	デフォルト = NO 値: YES または NO

状況によっては、SQL コンパイラーは IN リスト述部を結合に書き換えることができます。たとえば、次のような照会は、
 	
`SELECT *`
`FROM EMPLOYEE`
`WHERE DEPTNO IN ('D11', 'D21', 'E21')`
 	
次のように書くことができます。
 	
`SELECT *`
`FROM EMPLOYEE, (VALUES 'D11', 'D21', 'E21) AS V(DNO)`
`WHERE DEPTNO = V.DNO`
 	
DEPTNO に索引がある場合、この改訂はより良いパフォーマンスを提供することがあります。値のリストが最初にアクセスされて、結合述部に適用する索引を使用して NESTED LOOP 結合のある EMPLOYEE に結合されます。
 	
場合によっては、オプティマイザーが照会の再書き込みバージョンに最適な結合メソッドを判別するための正確な情報を持っていないことがあります。 IN リストにオプティマイザーがカタログ統計を使用して選択性を判別することを禁止するパラメーター・マーカーまたはホスト変数が含まれる場合に、このことが生じることがあります。このレジストリー変数はオプティマイザーが値のリストを結合するために、 IN リストを結合内の内部表として与える表を使用して、 NESTED LOOP 結合を優先的に使用するようにします。		

DB2_LIKE_VARCHAR	すべて	デフォルト =Y,Y

表 51. SQL コンパイラー変数 (続き)

変数名	オペレーティング・システム	値
説明		

サブエレメント統計の使用を制御します。これらの統計は、データにブランクで区切られた一連のサブフィールドまたはサブエレメント形式の構造がある場合、列内のデータ内容に関する統計です。サブエレメント統計の収集はオプションで、RUNSTATS コマンドまたは API 内のオプションによって制御されます。
 	
このレジストリー変数は、次の形式の述部をオプティマイザーが処理する方法に影響します。
 	
`COLUMN LIKE '%xxxxxx%'`
 	
xxxxxx は文字のストリングです。
 	
このレジストリー変数の使用方法を示す構文は次のとおりです。
 	
`db2set DB2_LIKE_VARCHAR=[Y|N|S|num1] [,Y|N|S|num2]`
 	
説明
• コンマの前にある用語、または述部の右にある唯一の用語は、 2 番目の用語が N に指定されているか、または列に正の値のサブエレメント統計がない場合にのみ、以下の意味になります。
– S - オプティマイザーは % 文字で囲まれたストリングの長さに基づいて、列を形成するために連結する一連のエレメントの各エレメントの長さを見積もる。
– Y - デフォルト。アルゴリズム・パラメーターのデフォルト値 1.9 を使用する。アルゴリズム・パラメーターで可変長サブエレメント・アルゴリズムを使用する。
– N - 固定長サブエレメント・アルゴリズムを使用する。
– num1 - 可変長サブエレメント・アルゴリズムにより、アルゴリズム・パラメーターとして num1 の値を使用する。
• コンマの後の用語は以下のような意味がありますが、正の値のサブエレメント統計を持つ列に対してのみです。
– N - サブエレメント統計を使用しない。最初の用語が有効になります。
– Y - デフォルト。正の値のサブエレメント統計を持つ列の場合に、アルゴリズム・パラメーターのデフォルト値 1.9 と一緒にサブエレメント統計を使用する可変長サブエレメント・アルゴリズムを使用する。
– num2 - 正の値のサブエレメント統計を持つ列の場合に、アルゴリズム・パラメーターとして num2 の値と一緒にサブエレメント統計を使用する可変長サブエレメント・アルゴリズムを使用する。		

DB2_MINIMIZE_LISTPREFETCH	すべて	デフォルト = NO 値: YES または NO
リスト・プリフェッチは、修飾するための RID を索引から検索して、それらをページ番号でソートしてからデータ・ページをプリフェッチすることを必要とする、特殊な表アクセス方式です。場合によっては、オプティマイザーがリスト・プリフェッチが良いアクセス方式であるかどうかを判別するための正確な情報を持っていないことがあります。述部選択性にオプティマイザーがカタログ統計を使用して選択性を判別することを禁止するパラメーター・マーカーまたはホスト変数が含まれる場合に、このことが生じることがあります。 このレジストリー変数は、そのような状況でオプティマイザーがリスト・プリフェッチを検討することを禁止します。		
DB2_SELECTIVITY	ALL	デフォルト = No 値: Yes または No

表 51. SQL コンパイラー変数 (続き)

変数名	オペレーティング・システム	値
説明		
このレジストリー変数は、 SQL ステートメント内の検索条件で、SELECTIVITY 文節が使用できる場所を制御します。 このレジストリー変数が「Yes」に設定された場合、以下の述部に SELECTIVITY 文節を設定可能です。 • 少なくとも 1 つの式がホスト変数を含む基本述部 • 一致条件、述部条件、またはエスケープ条件にホスト変数が含まれる LIKE 述部		
DB2_NEW_CORR_SQ_FF	すべて	デフォルト = OFF 値: ON または OFF

「ON」に設定すると、 SQL オプティマイザーが特定の副照会述部について計算した選択値に影響します。このパラメーターを使用すると、副照会の SELECT リストで MIN または MAX 統計関数を使用する等価副照会述部の選択値の正確度を高めることができます。例:
 	
`SELECT * FROM T WHERE`
`T.COL = (SELECT MIN(T.COL)`
`FROM T WHERE ...)`		

DB2_PRED_FACTORIZE	すべて	デフォルト = NO 値: YES または NO

オプティマイザーが、論理和から追加の述部を取り出す機会を探るかどうかを指定します。 状況によっては、追加の述部は中間の推定カーディナリティー、または結果セットを変更できます。以下の照会で、
 	
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`
 	
オプティマイザーは、以下の追加述部を生成できます。
 	
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE n1.lastname IN`
`('SMITH', 'JONES')`
`AND n2.lastname IN`
`('SMITH', 'JONES')`
`AND`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`		

DB2_REDUCED_OPTIMIZATION	すべて	デフォルト = NO 値: NO、YES、任意の整数、DISABLE

表 51. SQL コンパイラー変数 (続き)

変数名	オペレーティング・システム	値
説明		
このレジストリー変数によって、最適化機能を削減したり、最適化機能を指定した最適化レベルに固定して使用するように要求することができます。使用される最適化手法の数を削減する場合、最適化の際に使用される時間およびリソースも削減されます。 **注:** 最適化に使用される時間およびリソースは削減されることがあるが、最適ではないデータ・アクセス・プランが生成されるリスクは増加します。このレジストリー変数は、IBM またはそのパートナーから指示された場合にのみ使用します。 • NO に設定した場合 オプティマイザーは最適化手法を変更しません。 • YES に設定した場合 最適化レベルが 5 (デフォルト) 以下の場合、相当量の準備時間とリソースを消費するものの通常はより良いアクセス・プランを生成することのないいくつかの最適化手法をオプティマイザーは使用不可にします。 最適化レベルがちょうど 5 の場合、いくつかのその他の手法を縮小または使用不可にします。その結果、オプティマイザーによる最適化に必要な時間とリソースがさらに削減されることがあるが、同時に最適ではないアクセス・プランが生成されるリスクがさらに増大します。最適化レベルが 5 より低い場合、これらの技法のいくつかは最初から無効であることがあります。しかしそれらが有効であれば、有効のままとなります。 • 任意の整数に設定した場合 YES と同じ効果があり、さらにレベル 5 で最適化された動的に準備された照会のために以下の追加の動作が伴います。いずれかの照会ブロック内にある結合の合計数が設定値を超える場合、上記のレベル 5 最適化レベルについての説明で示したような、追加の最適化手法を使用不可にする代わりに、オプティマイザーは貪欲型結合列挙に切り替えます。これは、照会が最適化レベル 2 に類似したレベルで最適化されることを暗黙に示します。 • DISABLE に設定した場合 この DB2_REDUCED_OPTIMIZATION 変数によって拘束されないときのオプティマイザーの動作は、最適化レベル 5 での動的照会の最適化レベルを下げることがあります。この設定値はこの動作を使用不可にして、オプティマイザーがレベル 5 の最適化を完全に実行することを要求します。 最適化レベル 5 での動的な最適化レベルの引き下げは、 DB2_REDUCED_OPTIMIZATION を YES に設定したときの最適化レベルがちょうど 5 の場合について説明された動作、および整数の設定値について説明された動作よりも優先されることに注意してください。		

表 51. SQL コンパイラー変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2_SQLROUTINE_PREPOPTS	すべて	デフォルト = 空ストリング 値: • BLOCKING {UNAMBIG | ALL | NO} • DATETIME {DEF | USA | EUR | ISO | JIS | LOC} • DEGREE {1 | *並列処理の多重度* | ANY} • DYNAMICRULES {BIND | RUN} • EXPLAIN {NO | YES | ALL} • EXPLSNAP {NO | YES | ALL} • FEDERATED {NO | YES} • INSERT {DEF | BUF} • ISOLATION {CS | RR | UR | RS | NC} • QUERYOPT *最適化レベル* • VALIDATE {RUN | BIND}
DB2_SQLROUTINE_PREPOPTS レジストリー変数を使用すると、 SQL プロシージャーのプリコンパイル・オプションと BIND オプションをカスタマイズできます。		

関連概念:

- 81 ページの『最適化クラスのガイドライン』
- 179 ページの『最適の結合を選択する方法』
- 571 ページの『DB2 レジストリー変数と環境変数』

関連資料:

- 83 ページの『最適化クラス』

パフォーマンス変数

表 52. パフォーマンス変数

変数名	オペレーティング・システム	値
説明		
DB2AFFINITIES	AIX 5 以上、すべての Linux、ただし zSeries (32 ビット) を除く	デフォルト = 設定なし 値: 構成ファイルへの有効なパス

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		

リソース・ポリシーを定義し、DB2 が使用するオペレーティング・システム・リソースを制限できます。たとえば、AIX または Linux の場合、このレジストリー変数を使用して DB2 が使用する処理プログラムのセットを制限できます。
 	
AIX NUMA が使用可能なマシンでは、ポリシーを定義して、DB2 が使用するリソース・セットを指定できます。リソース・セット・バインディングを使用すると、各 DB2 プロセスが個別に特定のリソース・セットにバインドされます。パフォーマンス調整の状況によっては、この方法が役立つ場合があります。
 	
このレジストリー変数では、 DB2 プロセスをオペレーティング・システム・リソースにバインドするときに使用するポリシーが定義された構成ファイルへのパスを設定できます。リソース・ポリシーを使用することにより、 DB2 を制限するオペレーティング・システム・リソースのセットを指定できます。各 DB2 プロセスはこのセットの単一リソースにバインドされます。リソース割り当ては循環ラウンドロビン方式で行われます。
 	
構成ファイルの例:
 	
`Example 1: Bind all DB2 processes to either CPU 1 or 3.`
`  <RESOURCE_POLICY>`
`  <METHOD>CPU</METHOD>`
`  <RESOURCE>1</RESOURCE>`
`  <RESOURCE>3</RESOURCE>`
`  </RESOURCE_POLICY>`
 	
`Example 2: Bind DB2 processes to one of the following resource sets:`
`  sys/node.03.00000, sys/node.03.00001, sys/node.03.00002,`
`sys/node.03.00003`
`  <RESOURCE_POLICY>`
`  <METHOD>RSET</METHOD>`
`  <RESOURCE>sys/node.03.00000</RESOURCE>`
`  <RESOURCE>sys/node.03.00001</RESOURCE>`
`  <RESOURCE>sys/node.03.00002</RESOURCE>`
`  <RESOURCE>sys/node.03.00003</RESOURCE>`
`  </RESOURCE_POLICY>`
 	
注: RSET メソッドを使用するには CAP_NUMA_ATTACH 機能が必要であるため、このメソッドは Linux ではサポートされていません。		

DB2_ALLOCATION_SIZE	すべて	デフォルト = 8 MB 範囲: 32 KB ～ 256 MB
バッファー・プールのメモリー割り振りのサイズを指定します。 このレジストリー変数に高い値を設定することの潜在的な利点は、バッファー・プールに割り振られる希望のメモリー量に達するために、必要な割り振りの回数が少なくすむということです。 このレジストリー変数に高い値を設定する際にかかる潜在的コストとして、バッファー・プールが単一の割り振りサイズによって変更される場合にメモリーが無駄になる可能性があるという点が挙げられます。たとえば、DB2_ALLOCATION_SIZE の値が 8 MB で、バッファー・プールが 4 MB 削減される場合、8 MB セグメント全体は解放できないので、この 4 MB は無駄になります。		
DB2_APM_PERFORMANCE	すべて	デフォルト = OFF 値: ON、OFF

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
SQL キャッシュ (パッケージ・キャッシュ) の動作に影響するアクセス・プラン・マネージャー (APM) 内でパフォーマンス関連の変更を使用可能にするには、この変数を ON に設定します。これらの設定値は通常、実動システムには勧められません。これらはパッケージ外キャッシュ・エラー、メモリー使用量の増加、またはその両方など、いくらかの制限を生じさせます。 DB2_APM_PERFORMANCE を ON に設定すると、「パッケージ・ロックなし」モードも使用可能になります。このモードは、グローバル SQL キャッシュがパッケージ・ロックを使用しないで操作できるようにします。パッケージ・ロックは、キャッシュされたパッケージ項目が除去されないように保護する内部システム・ロックです。 「パッケージ・ロックなし」モードにより、パフォーマンスはいくらか改善されることがあるが、特定のデータベース操作は許可されなくなります。これらの禁止される操作には、パッケージを無効にする操作、パッケージを操作不能にする操作、 PRECOMPILE、BIND、および REBIND が含まれます。		
DB2ASSUMEUPDATE	すべて	デフォルト = OFF 値: ON、OFF
使用可能になっていると、DB2 は、UPDATE ステートメントで提供されるすべての固定長の列が実際に変更中であると想定することができます。これにより、DB2 が既存の列値と提供される新規値を比較して列が実際に変更中かどうかを判別する必要がなくなります。列が更新用に (たとえば SET 文節で) 提供されており、しかし実際にはその列が変更されていないときにこのレジストリー変数を使用すると、ロギングや索引の保守が余分に発生する可能性があります。 このレジストリー変数は更新時にチェックされます。		
DB2_AVOID_PREFETCH	すべて	デフォルト = OFF 値: ON または OFF
クラッシュ・リカバリーにおいて、プリフェッチを使用するかを指定します。 DB2_AVOID_PREFETCH=ON の場合は、プリフェッチは使用されません。		
DB2_AWE	Windows 2000	デフォルト = null 値: <entry>[;<entry>...]。 <entry>=<バッファー・プール ID>、<物理ページの数>、 <アドレス・ウィンドウの数>
32 ビットの Windows 2000 プラットフォームの DB2 UDB では、バッファー・プールに 64 GB までのメモリーを割り振ることができます。 Windows 2000 は、Address Windowing Extensions (AWE) バッファー・プールをサポートするように正しく構成されている必要があります。これには、『メモリー内のページのロック』権利をユーザーに関連付け、物理ページおよびアドレス・ウィンドウ・ページを割り振り、このレジストリー変数を設定する作業が含まれる。この変数を設定するには、 AWE サポートに使用するバッファー・プールのバッファー・プール ID を知っておく必要があります。バッファー・プールの ID は、 SYSCAT.BUFFERPOOLS システム・カタログ・ビューの BUFFERPOOLID 列にあります。 **注:** • AWE サポートが使用可能になっている場合には、拡張ストレージをデータベースのバッファー・プールに対して使用することはできません。 • このレジストリー変数を使用して参照されるバッファー・プールは、すでに SYSCAT.SYSBUFFERPOOLS に存在していなければなりません。 • AWE で使用可能なバッファー・プールは、ブロック・ベース I/O で使用可能なバッファー・プールより優先します。バッファー・プールが AWE とブロック・ベース I/O の両方に構成されている場合は、AWE がブロック・ベース I/O より優先します。		

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2_BINSORT	すべて	デフォルト = YES 値: YES または NO
ソートの CPU 時間と経過時間が減少する新しいソート・アルゴリズムを使用可能にします。この新アルゴリズムにより、DB2 UDB の非常に効率的な整数ソート技法が、あらゆるソート・データ・タイプ (BIGINT、CHAR、VARCHAR、FLOAT、DECIMAL、およびそれらを組み合わせたデータ・タイプ) に拡張されます。		
DB2BPVARS	各パラメーターに指定されたとおり	デフォルト = パス

2 種類のパラメーターを使用してバッファー・プールを調整できます。パラメーターの 1 つの種類は Windows だけで使用可能なもので、バッファー・プールが特定のタイプのコンテナーの分散読み取りを使用することを指定します。他の種類のパラメーターは、すべてのプラットフォームで使用可能なもので、プリフェッチ動作に影響を与えます。
 	
複数のパラメーターは ASCII ファイル内で、各行に 1 つずつ `parameter=value` の形式で指定できます。例として、`bpvars.vars` という名前のファイルには以下の行が含まれます。
 	
`NO_NT_SCATTER = 1`
`NUMPREFETCHQUEUES = 2`
 	
`bpvars.vars` が F:¥vars¥ に保管されていると想定すると、これらの変数を設定するには以下のコマンドを実行します。
 	
`db2set DB2BPVARS=F:¥vars¥bpvars.vars`
 	
分散読み取りパラメーター
 	
分散読み取りパラメーターは、それぞれのコンテナー・タイプに対する順次プリフェッチが大量に行われ、DB2NTNOCACHE をすでに ON に設定したシステムで推奨されます。これらのパラメーターは、Windows プラットフォームだけで使用可能であり、 NT_SCATTER_DMSFILE, NT_SCATTER_DMSDEVICE および NT_SCATTER_SMS です。 NO_NT_SCATTER パラメーターを指定すると、すべてのコンテナーで分散読み取りを明示的に禁止します。指定の種類のすべてのコンテナーで分散読み取りをオンに切り替えるには、特定のパラメーターを使用します。上記の個々のパラメーターのデフォルトはゼロ (OFF) で、可能な値はゼロ (OFF) および 1 (ON) です。
注: 分散読み取りをオンに切り替えることができるのは、 DB2NTNOCACHE を ON に設定して Windows ファイルのキャッシングをオフにした場合だけです。 DB2NTNOCACHE の設定が OFF に設定されているか未設定の場合は、管理通知ログに警告メッセージが書き込まれ、分散読み取りは使用不可のままになります。		

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
プリフェッチ調整パラメーター プリフェッチ調整パラメーターは、NUMPREFETCHQUEUES および PREFETCHQUEUESIZE です。これらのパラメーターはすべてのプラットフォームで使用可能であり、バッファー・プール・データ・プリフェッチを改善するために使用できます。たとえば、目的の PREFETCHSIZE が複数の PREFETCHSIZE/EXTENTSIZE プリフェッチ要求に分割されている順次プリフェッチについて考えます。この場合、要求は入出力サーバーが非同期入出力を実行するプリフェッチ・キューに入れられます。デフォルトでは、DB2 はデータベース・パーティションごとにサイズ max(100,2*NUM_IOSERVERS) の 1 つのキューを保守します。一部の環境では、キューの増加またはキューのサイズの変更、もしくはその両方により、パフォーマンスが改善されます。プリフェッチ・キューの数は最大で入出力サーバーの数の半分とします。これらのパラメーターを設定するとき、現行ユーザー数などのワークロード特性と同様に、 PREFETCHSIZE、EXTENTSIZE、NUM_IOSERVERS などのパラメーター、およびバッファー・プール・サイズを検討してください。 デフォルト値が環境に対して小さすぎると考えられる場合、最初に少しだけ値を増加します。たとえば、NUMPREFETCHQUEUES=4 および PREFETCHQUEUESIZE=200 と設定できます。これらのパラメーターの変更を制御された方法で行い、変更の効果を観察および評価できるようにします。 NUMPREFETCHQUEUES では、デフォルトは 1 で、値の範囲は 1 から NUM_IOSERVERS となります。NUMPREFETCHQUEUES を 1 未満に設定した場合、それは 1 に調整されます。その値を NUM_IOSERVERS よりも大きく設定した場合、それは NUM_IOSERVERS に調整されます。 PREFETCHQUEUESIZE では、デフォルト値は max(100,2*NUM_IOSERVERS) です。値の範囲は 1 〜 32767 です。PREFETCHQUEUESIZE を 1 未満に設定した場合、それはデフォルト値に調整されます。その値を 32767 より大きく設定した場合、それは 32767 に調整されます。		
DB2CHKPTR	すべて	デフォルト = OFF 値: ON または OFF
入力のポインター検査が必要であるかどうかを指定します。		
DB2CHKSQLDA	すべて	デフォルト = OFF 値: ON または OFF
入力の SQLDA 検査が必要であるかどうかを指定します。		
DB2_ENABLE_BUFPD	すべて	デフォルト = YES 値: ON または OFF
照会パフォーマンスを向上させるために DB2 で中間バッファリングを使用するかどうかを指定します。バッファリングによって、あらゆる環境で照会パフォーマンスが向上するわけではありません。それぞれの照会パフォーマンスが向上するかどうかを判別するために、テストを行う必要があります。		
DB2_EVALUNCOMMITTED	すべて	デフォルト = OFF 値: ON、OFF

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
使用可能になっていると、可能な場合に、表アクセス・スキャンまたは索引アクセス・スキャンは、データ・レコードが述部評価を満たしたことがわかるまで行ロックを据え置くかまたは回避できます。 この変数を使用可能にすると、非コミット・データで述部評価が行われる場合があります。 これは、カーソル固定分離レベルかまたは読み取り固定分離レベルのいずれかを使用するステートメントに対してのみ適用可能です。索引スキャンの場合、索引はタイプ 2 索引でなければなりません。 さらに、削除された行は表スキャンのアクセス時に無条件でスキップされますが、削除されたキーは、タイプ 2 索引のスキャンでは、レジストリー変数 DB2_SKIPDELETED も設定されていなければスキップされません。 この DB2_EVALUNCOMMITTED レジストリー変数の活動化は **db2start** では有効ですが、据え置きロッキングを適用可能とするかどうかに関する判断は、ステートメントのコンパイル時またはバインド時に行われます。		
DB2_EXTENDED_OPTIMIZATION	すべて	デフォルト = OFF 値: ON または OFF
照会パフォーマンスを向上させるために、照会オプティマイザーが最適化拡張を使用するかどうかを指定します。拡張によって、あらゆる環境で照会パフォーマンスが向上するわけではありません。それぞれの照会パフォーマンスが向上するかどうかを判別するために、テストを行う必要があります。		
DB2_KEEPTABLELOCK	すべて	デフォルト = OFF 値: ON、OFF
使用可能になっていると、DB2 は、非コミット読み取りまたはカーソル固定のいずれかの分離レベルがクローズされるときの表ロックのリリースを回避できます。保持される表ロックは、読み取り固定スキャンまたは反復可能読み取りスキャンの場合にリリースされるように、トランザクションの終了時にリリースされます。 このレジストリー変数はステートメントのコンパイル時またはバインド時にチェックされます。		
DB2_LGPAGE_BP	AIX 5.x 64 ビットのみ Linux	デフォルト = OFF 値: ON または OFF

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		

DB2_LGPAGE_BP レジストリー変数は、AIX 5.x または適切なカーネル・サポートを備えた Linux アーキテクチャー上で実行する場合に、ラージ・ページ・サポートを使用可能にするために使用します。これは、DB2 UDB for AIX、64 ビット版、および DB2 UDB for Linux でのみサポートされます。ラージ・ページの使用は主に、高性能コンピューティング・アプリケーションのパフォーマンスの向上を意図したものです。集中的なメモリー・アクセスを必要とし、大量の仮想メモリーを使用するアプリケーションでは、このラージ・ページの使用によってパフォーマンスを向上できるでしょう。 DB2 でラージ･ページを使用できるようにするには、まずオペレーティング・システムがラージ・ページを使用できるように構成する必要があります。
 	
64 ビット DB2 for AIX では、この変数を使用可能にすると、データベース・メモリーを支持する共用メモリー・セグメントのサイズが必要最小量に減少します (デフォルトでは 64GB セグメントが作成されます。詳細については、database_memory 構成パラメーターを参照してください)。こうして、使用される可能性のある量以上の共用メモリーが RAM 内に滞留するのを防ぐことができます。
 	
この変数セットを使用することによって、全体的なデータベース共用メモリー構成を動的に増やす機能 (たとえばバッファー・プールのサイズを増やす機能) が制限されます。
 	
Linux では、 libcap.so ライブラリーの可用性に関する追加の要件があります。このオプションが有効であるためには、このライブラリーがインストールされていなければなりません。このオプションがオンになっていて、このライブラリーがシステム上にない場合、 DB2 は大容量のカーネル・ページを使用不可にして、以前と同様に機能し続けます。
 	
Linux では、大容量カーネル・ページが使用可能かどうかを検査するために、次のコマンドを発行します。
`cat /proc/meminfo`
 	
使用可能である場合は、次の 3 行が表示されます (マシン上に構成されているメモリーの量によって数値は異なります)。
`HugePages_Total: 200`
`HugePages_Free: 200`
`Hugepagesize: 16384 kB`
 	
これらの行が表示されない場合、または HugePages_Total が `0` である場合は、オペレーティング・システムまたはカーネルの構成が必要です。		

DB2MAXFSCRSEARCH	すべて	デフォルト= 5 値: -1、1 〜 33 554
表にレコードが追加された場合、検索のために、フリー・スペース制御レコードの数を指定します。デフォルトは、フリー・スペース制御レコードを 5 つ検索します。 この値の変更は、スペース再利用で挿入速度の平衡を取れるようにします。スペース再利用の最適化のためには大きな値を使用します。挿入速度の最適化のためには小さな値を使用します。値を -1 に設定するとデータベース・マネージャーはすべてのフリー・スペース制御レコードを強制的に検索します。		
DB2_MAX_NON_TABLE_LOCKS	すべて	デフォルト = YES 値: 以下を参照してください。

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
トランザクションによってすべてが解放される前に持つことのできる、 NON 表ロックの最大数を定義します。 NON 表ロックとは、トランザクションが使用を終えてもハッシュ・テーブルやトランザクション・チェーンに保持されている表ロックのことです。トランザクションが同じ表に何回もアクセスすることはよくあるので、ロックを保持してその状態を NON に変更することでパフォーマンスが改善されます。ロックは再作成する必要がないからです。 最良の結果を得るためにこの変数に推奨されている値は、接続によってアクセスが予想される表の数の最大数です。ユーザー定義の値を指定しない場合、デフォルト値は次のようになります。ロックリスト・サイズが SQLP_THRESHOLD_VAL_OF_LRG_LOCKLIST_SZ_FOR_MAX_NON_LOCKS (現行値 8000) 以上の場合、デフォルト値は SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_LARGE (現行値 150) になります。それ以外の場合のデフォルト値は SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_SMALL (現行値 0) です。		
DB2MEMDISCLAIM	AIX	デフォルト = YES 値: YES または NO
AIX では、DB2 プロセスが使用するメモリーに、関連するページング・スペースを存在させることができます。このページング・スペースは、関連するメモリーが解放された後も予約されたままになる場合があります。これは、AIX システムの (調整可能な) 仮想メモリー管理割り振りポリシーによって異なります。 DB2MEMDISCLAIM レジストリー変数は、解放されたメモリーと予約ページング・スペースとの関連付けを AIX が解除することを DB2 エージェントが明示的に要求するかどうかを制御します。 DB2MEMDISCLAIM を YES に設定すると、ページング・スペースの所要量が少なくなり、ページングのディスク活動も減少します。 DB2MEMDISCLAIM を NO に設定すると、ページング・スペースの所要量は多くなり、ページングのディスク活動も増加します。ページング・スペースが多い場合や、ページングが行われないほど実メモリーが十分にある場合などは、 NO を設定してもパフォーマンスはわずかしか向上しません。		
DB2MEMMAXFREE	すべて	デフォルト= 8 388 608 バイト 値: 0 ～ 2^{32}-1 バイト
未使用メモリーがオペレーティング・システムに戻される前に、DB2 プロセスによって保持される未使用の専用メモリーの最大バイト数を指定します。		
DB2_MMAP_READ	AIX	デフォルト = ON 値: ON または OFF
DB2_MMAP_WRITE と一緒に使用して、入出力の代替方法として DB2 が mmap を使用できるようにします。複数プロセスが同一ファイルの異なるセクションから読み取る場合は、ほとんどの環境において、オペレーティング・システムのロックを防ぐために mmap を使用すべきです。 これらの変数が ON に設定されている場合、 DB2 バッファー・プールとの間で読み書きされるデータは AIX メモリー・キャッシュをバイパスします。使用する DB2 バッファー・プールが比較的小さい場合は、バッファー・プールのサイズを増やすのではなく、 DB2_MMAP_READ と DB2_MMAP_WRITE を OFF に設定することによって AIX メモリー・キャッシングを活用することを考慮する必要があります。		
DB2_MMAP_WRITE	AIX	デフォルト = ON 値: ON または OFF

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2_MMAP_READ と一緒に使用して、入出力の代替方法として DB2 が mmap を使用できるようにします。複数プロセスが同一ファイルの異なるセクションに書き出す場合は、ほとんどの環境において、オペレーティング・システムのロックを防ぐために mmap を使用すべきです。 これらの変数が ON に設定されている場合、 DB2 バッファー・プールとの間で読み書きされるデータは AIX メモリー・キャッシュをバイパスします。使用する DB2 バッファー・プールが比較的小さい場合は、バッファー・プールのサイズを増やすのではなく、 DB2_MMAP_READ と DB2_MMAP_WRITE を OFF に設定することによって AIX メモリー・キャッシングを活用することを考慮する必要があります。		
DB2_NO_FORK_CHECK	UNIX	デフォルト = OFF 値: ON、OFF
この変数を使用可能にすると、DB2 ランタイム・クライアントは、現行プロセスがフォーク呼び出しの結果であるかどうかを判別するチェックを最小化します。これにより、fork() API を使用しない DB2 アプリケーションのパフォーマンスは改善されます。		
DB2_NO_MPFA_FOR_NEW_DB	すべて	デフォルト= 設定なし 値: YES
CREATE DATABASE コマンドまたはそれと同等の API によって作成されるデータベースは、マルチページ・ファイル割り振り (MPFA) を使用可能にしました。一度データベースに対して使用可能になった MPFA は、使用不可にすることができません。 MPFA を使用不可にしたデータベースを作成するには、このレジストリー変数を YES に設定してインスタンスを再始動してから、データベースを作成します。このレジストリー変数を設定すると、作成されるデータベースはすべて MPFA が使用不可になります。 MPFA が使用不可になっているデータベースの MPFA を使用可能にするには、 **db2empfa** コマンドを使用します。		
DB2NTMEMSIZE	Windows NT	デフォルト = (メモリー・セグメントにより異なる)
Windows NT では、プロセス間でアドレスの一致を保証するために DLL 初期設定時にすべての共用メモリー・セグメントを予約する必要があります。必要に応じて DB2NTMEMSIZE により Windows NT の DB2 デフォルトをオーバーライドできます。ほとんどの状態では、デフォルト値で十分なはずです。メモリー・セグメント、デフォルトのサイズ、およびオーバーライド・オプションは以下のとおりです。 1) データベース・カーネル: デフォルトのサイズ 16777216 (16 MB); オーバーライド・オプションは DBMS:<number of bytes> 2) 並列 FCM バッファー: デフォルトのサイズ 22020096 (21 MB); オーバーライド・オプションは FCM:<number of bytes> 3) データベース Admin GUI: デフォルトのサイズ 33554432 (32 MB); オーバーライド・オプションは DBAT:<number of bytes> 4) fenced ストアード・プロシージャー: デフォルトのサイズ 16777216 (16 MB); オーバーライド・オプションは APLD:<number of bytes>。オーバーライド・オプションをセミコロン (;) で区切って、複数のセグメントをオーバーライドできます。たとえば、データベース・カーネルを 256K に制限し、 FCM バッファーを 64 MB に制限するには、以下のようにします。 `db2set DB2NTMEMSIZE=DBMS:256000;FCM:64000000`		
DB2NTNOCACHE	Windows NT	デフォルト = OFF 値: ON または OFF

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2 がデータベース・ファイルを NOCACHE オプションを指定してオープンするかどうかを指定します。DB2NTNOCACHE=ON の場合は、ファイル・システムのキャッシュは除去されます。 DB2NTNOCACHE=OFF の場合、オペレーティング・システムは DB2 ファイルをキャッシュに入れます。これは、long fields または LOBs を含んでいるファイルを除くすべてのデータに適用されます。システム・キャッシュを除去すると、より多くのメモリーがデータベースに利用できるようになるため、バッファー・プールやソート・ヒープの量を増やすことができます。 Windows NT では、デフォルトの動作として、ファイルをオープンするときにキャッシュに入れられます。ファイル内の 1 GB ごとに 1 MB がシステム・プールから予約されます。このレジストリー変数を使用して、キャッシュに関する (文書化されていない) 192 MB 制限をオーバーライドします。キャッシュの限界に達すると、リソース不足を示すエラーが表示されます。		
DB2NTPRICLASS	Windows NT	デフォルト = null 値: R、H、(任意の他の値)
DB2 インスタンスの優先度クラスを設定します (プログラム DB2SYSCS.EXE)。次の 3 つの優先度クラスがあります。 • NORMAL_PRIORITY_CLASS (デフォルトの優先度クラス) • REALTIME_PRIORITY_CLASS (「R」を使って設定) • HIGH_PRIORITY_CLASS (「H」を使って設定) この変数は、個々のスレッド優先順位 (DB2PRIORITIES を使って設定) と一緒に使われ、システム中の別のスレッドに関する DB2 スレッドの絶対優先順位を決定します。 **注:** この変数を使用する際には、注意する必要があります。誤用すると、システム・パフォーマンス全体に悪い影響を及ぼす可能性があります。 詳細は、Win32 資料の *SetPriorityClass()* API を参照してください。		
DB2NTWORKSET	Windows NT	デフォルト = 1,1
DB2 に利用できる最小および最大の実効ページ・セットを変更するために使用されます。デフォルトを使用して、ページングが行われていない場合は、プロセスの実効ページ・セットは必要なだけ大きくすることができます。ただし、ページングが発生しているときは、プロセスが持つことができる最大の実効ページ・セットは約 1 MB です。DB2NTWORKSET を使えば、このデフォルトの動作をオーバーライドできます。 DB2 に対する DB2NTWORKSET の指定は、 DB2NTWORKSET=min,max の構文を使用します。ここで、min と max はメガバイト単位で表されます。		
DB2_OVERRIDE_BPF	すべて	デフォルト = 設定なし 値: 正数のページ数 OR <entry>[;<entry>...] (<entry>=<バッファー・プール ID>,<ページ数>

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
データベース活動化時または初回接続時に作成されるバッファー・プールのサイズをページ数で指定します。これが役立つのは、メモリー制約の結果としてデータベース活動化時または初回接続時に障害が発生する場合です。データベース・マネージャーによって最小のバッファー・プール (16 ページ) も起動しない場合、ユーザーはこの環境変数を使ってさらに小さいページ数を指定した後に再試行することができます。メモリー制約は、稀に実メモリーの不足のために起こることもあれば、データベース・マネージャーがバッファー・プールの構成を間違えて、大きいバッファー・プールを割り当てようとしたために起こることもあります。設定時に、この値は現行バッファー・プールをオーバーライドします。 バッファー・プールのすべてまたはサブセットのサイズを一時的に変更して始動できるように、以下も使用できます。 <entry>[;<entry>...] (<entry>=<バッファー・プール ID>,<ページの数>)		
DB2_PINNED_BP	AIX、HP-UX	デフォルト = NO 値: YES または NO

この変数は、一部の AIX オペレーティング・システムで、データベースに関連するデータベース・グローバル・メモリー (バッファー・プールを含む) をメイン・メモリーに指定するために使用されます。 データベース・グローバル・メモリーをシステム・メイン・メモリーに保持することにより、データベース・パフォーマンスがより一貫性のあるものになります。
 	
たとえば、バッファー・プールがシステム・メイン・メモリーからスワップアウトされる場合、データベース・パフォーマンスは低下します。バッファー・プールをシステム・メモリーに保持することによってディスク入出力が減ると、データベース・パフォーマンスは改善されます。別のアプリケーションがより多くのメイン・メモリーを要求した場合、システム・メイン・メモリー所要量に応じて、データベース・グローバル・メモリーをメイン・メモリーからスワップアウトできます。
 	
64 ビット DB2 for AIX では、この変数を使用可能にすると、データベース・メモリーを支持する共用メモリー・セグメントのサイズが必要最小量に減少します (デフォルトでは 64GB セグメントが作成されます。詳細については、database_memory 構成パラメーターを参照してください)。こうして、使用される可能性のある量以上の共用メモリーが RAM 内に滞留するのを防ぐことができます。
 	
この変数セットを使用することによって、全体的なデータベース共用メモリー構成を動的に増やす機能 (たとえばバッファー・プールのサイズを増やす機能) が制限されます。
 	
64 ビット環境での HP-UX では、このレジストリー環境を変更する他に、 DB2 インスタンス・グループに MLOCK 特権を与えなければなりません。これを行うには、ルート・アクセス権限を持つユーザーが以下の処置を実行します。
1. DB2 インスタンス・グループを /etc/privgroup ファイルに追加します。たとえば、DB2 インスタンス・グループが db2iadm1 グループに属している場合、次の行を /etc/privgroup ファイルに追加します。
`db2iadm1 MLOCK`
2. 次のコマンドを発行します。
`setprivgrp -f /etc/privgroup`		

DB2PRIORITIES	すべて	値の設定はプラットフォームにより異なる
DB2 プロセスとスレッドの優先順位を制御します。		
DB2_SCATTERED_IO	Linux	デフォルト = OFF 値: ON、OFF

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
この変数は readv() を使用してディスクから読み取りを行う、散在データ読み取りをオンにします。ベクトル付けされたロー I/O パフォーマンス改良 Linux カーネル・パッチを含むシステム上で実行中の場合、パフォーマンスを向上させるにはこの変数をオンにする必要があります。 このカーネル・パッチは、IA-32 の場合は現在 UnitedLinux 1.0 SP2 以降にあり、次回の Linux 2.6 カーネルすべてに含まれる予定です。		
DB2_SKIPDELETED	すべて	デフォルト = OFF 値: ON、OFF
使用可能になっていると、カーソル固定分離レベルまたは読み取り固定分離レベルのいずれかを使用するステートメントは、索引アクセス中に削除されたキー、および表アクセス中に削除された行を無条件でスキップすることができます。 DB2_EVALUNCOMMITTED を使用可能にすると、削除された行は自動的にスキップされますが、非コミットの実際には削除されていないタイプ 2 索引のキーは、DB2_SKIPDELETED も使用可能にしていなければスキップされません。 このレジストリー変数は DB2 カタログ表上のカーソルの動作には影響を与えません。 このレジストリー変数の活動化は db2start では有効です。		
DB2_SMS_TRUNC_TMPTABLE_THRESH	Windows	-1、0-n。n = コンテナーごとに保持されるエクステントの数
一時表を表すファイルを SMS 表スペースに保持する際の、最小ファイル・サイズしきい値を指定します。この変数を 0 より大きい値に設定すると、一時表を使用するたびに、ファイルのドロップおよび再作成に関係するシステム・オーバーヘッドが若干減少します。デフォルトでは、一時表が不要になると、その表のためのファイルはコンテナーごとに 1 エクステントに切り捨てられます。ファイルのサイズがすでに 1 エクステントまたはそれより小さい場合は、そのままにされます。この変数の値が 1 より大きい場合は、より大きいファイルが保持されます。 この変数が -1 に設定されている場合は、ファイルが切り捨てられることはなく、ファイルはシステム・リソースによる制限を除いて無制限に増大します。 この変数が 0 に設定されている場合、特殊しきい値処理が実行されることはありません。その代わり、一時表が不要になると、そのファイルは切り捨てられて 0 になります。		
DB2_SORT_AFTER_TQ	すべて	デフォルト = NO 値: YES または NO
受信終了時にデータをソートすることが必要で、受信ノード数が送信ノード数と等しい場合、オプティマイザーがパーティション・データベースの直接表キューを処理する方法を指定します。 DB2_SORT_AFTER_TQ=NO の場合、オプティマイザーは送信終了時には行のソートを、受信終了時には行のマージを行う傾向があります。 DB2_SORT_AFTER_TQ=YES の場合、オプティマイザーはソートをしないで行を送信し、すべての行を受信した後の受信終了時にもマージを行わない傾向があります。		
DB2_SELUDI_COMM_BUFFER	すべて	デフォルト = OFF 値 = ON、OFF

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
SELECT from UPDATE/INSERT/DELETE (UDI) 照会に対するブロッキング・カーソルの処理に適用されます。使用可能になっていると、このレジストリー変数は照会の結果が一時表に保管されないようにします。その代わりに、SELECT from UDI 照会に対するブロッキング・カーソルの OPEN 処理中に、 DB2 は照会の結果全体を通信バッファー・メモリー領域に直接バッファーしようとします。 **注:** 通信バッファー・スペースが照会の結果全体を保持できるほど十分には大きくない場合、 SQLCODE -906 が発行され、トランザクションはロールバックされます。ローカル・アプリケーションおよびリモート・アプリケーションそれぞれの通信バッファー・メモリー領域のサイズの調整については、 *aslheapsz* および *rqrioblk* データベース・マネージャー構成パラメーターを参照してください。 このレジストリー変数は、パーティション・データベース環境で、またはパーティション内並列処理が使用可能である場合にはサポートされていません。		
DB2_TRUSTED_BINDIN	すべて	デフォルト = OFF 値 = OFF、ON、CHECK

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2_TRUSTED_BINDIN が使用可能になっていると、組み込み unfenced ストアード・プロシージャー内にホスト変数を含む SQL ステートメントの実行速度が増します。 この変数が使用可能になっていると、組み込み unfenced ストアード・プロシージャー内に含まれる SQL ステートメントのバインド中に外部 SQLDA フォーマットから内部 DB2 フォーマットへの変換は行われません。これにより、組み込み SQL ステートメントの処理速度が増します。 この変数が使用可能になっている場合、以下のデータ・タイプは組み込み unfenced ストアード・プロシージャーでサポートされません。 • SQL_TYP_DATE • SQL_TYP_TIME • SQL_TYP_STAMP • SQL_TYP_DATALINK • SQL_TYP_CGSTR • SQL_TYP_BLOB • SQL_TYP_CLOB • SQL_TYP_DBCLOB • SQL_TYP_CSTR • SQL_TYP_LSTR • SQL_TYP_BLOB_LOCATOR • SQL_TYP_CLOB_LOCATOR • SQL_TYP_DCLOB_LOCATOR • SQL_TYP_BLOB_FILE • SQL_TYP_CLOB_FILE • SQL_TYP_DCLOB_FILE • SQL_TYP_BLOB_FILE_OBSOLETE • SQL_TYP_CLOB_FILE_OBSOLETE • SQL_TYP_DCLOB_FILE_OBSOLETE これらのデータ・タイプが見つかると、 SQLCODE -804、SQLSTATE 07002 が戻されます。 **注:** 入力ホスト変数のデータ・タイプと長さは、対応するエレメントの内部データ・タイプと長さとに正確に一致する必要があります。ホスト変数の場合、この要件は常に満たされます。しかし、パラメーター・マーカーの場合、データ・タイプが一致しているか確認する必要があります。データ・タイプと長さがすべての入力ホスト変数と一致しているかどうかの確認に CHECK オプションを使用できますが、しかしこのオプションは大抵パフォーマンスを低下させます。		
DB2_USE_ALTERNATE_PAGE_CLEANING	すべて	デフォルト = 設定なし 値: ON、OFF

表 52. パフォーマンス変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2 がデフォルトのページ・クリーニング方式の代わりにページ・クリーニング・アルゴリズムの代替方式を使用するかどうかを指定します。この変数が「ON」に設定されると、DB2 は事前の対策を講じたページ・クリーニングの方式を使用して、変更されたページをディスクに書き込み、LSN_GAP を保持し、積極的にビクティムを検索します。これを行うことにより、ページ・クリーナーは使用可能なディスク I/O 帯域幅をより効果的に使用できます。この変数が「ON」に設定されると、chngpgs_thresh データベース構成パラメーターはページ・クリーナー・アクティビティーを制御しないので、関係がなくなります。		

関連概念:

- 571 ページの『DB2 レジストリー変数と環境変数』

データ・リンク変数

表 53. データ・リンク変数

変数名	オペレーティング・システム	値
説明		
DLFM_BACKUP_DIR_NAME	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: null 値: 任意の有効なパス
DLFM_BACKUP_TARGET が LOCAL に設定されている場合に、アーカイブ・ファイルがバックアップされるディレクトリーのパスを指定します。		
DLFM_BACKUP_TARGET	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: null 値: LOCAL、TSM、XBSA
使用するバックアップ手段を指定します。 この変数が LOCAL に設定されている場合、 DLFM_BACKUP_DIR_NAME 変数を設定する必要があります。 この変数が XBSA に設定されている場合、 DLFM_BACKUP_TARGET_LIBRARY 変数を設定する必要があります。 このレジストリー変数の設定をあるターゲットから別のターゲットに変更しても、アーカイブ・ファイルは移動されません。新しいバックアップだけが新しい位置に置かれます。それまでにアーカイブされたファイルは移動されません。		
DLFM_BACKUP_TARGET_LIBRARY	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: null 値: DLL または共用ライブラリー名への任意の有効なパス
XBSA 準拠アーカイブ・サーバー DLL または共用ライブラリーへの完全修飾パスを指定します。このライブラリーは、 *libdfmxbsa.a* ライブラリーを使用してロードされます。 DLFM_BACKUP_TARGET が XBSA に設定されている場合、この変数が設定されていなければなりません。DLFM_BACKUP_TARGET 変数が別の値に設定されている場合は、適用されません。		

表 53. データ・リンク変数 (続き)

変数名	オペレーティング・システム	値
説明		
DLFM_GC_MODE	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: PASSIVE 値: SLEEP、 PASSIVE、または ACTIVE
データ・リンク・サーバーでのガーベッジ・ファイル・コレクションの制御を指定します。 SLEEP に設定すると、ガーベッジ・コレクションは行われません。 PASSIVE に設定すると、他のトランザクションが実行されていない場合にのみガーベッジ・コレクションが実行されます。 ACTIVE に設定すると、他のトランザクションが実行されている場合でもガーベッジ・コレクションが実行されます。		
DLFM_INSTALL_PATH	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト AIX および Solaris オペレーティング環境の場合: `/home/<instance>/sqllib/bin`。ここで <instance> は Data Links Manager インスタンス ID です。 Windows の場合: `%DB2PATH%¥bin` (%DB2PATH% が設定されている場合) または `c:¥sqllib¥bin` (%DB2PATH% が設定されていない場合) 範囲: 任意の有効なパス
データ・リンク実行可能ファイルがインストールされているパスを指定します。		
DLFM_PORT	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: 50100 値: 任意の有効なポート番号
DB2 Data Links Manager を実行する Data Links サーバーとの通信に使用するポート番号を指定します。		
DLFM_TSM_MGMTCLASS	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: デフォルト TSM 管理クラス 値: 有効 TSM 管理クラス
リンクしたファイルをアーカイブ、検索するために使用する TSM 管理クラスを指定します。この変数の値セットがない場合は、デフォルト TSM 管理クラスが使用されます。		
DLFM_START_ASNCOPYD	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: NO 値: YES、NO
DLFM が開始したら必ず Data Links Manager Replication Daemon (DLFM_ASNCOPYD) を開始するかどうかを指定します。 DB2 Replication を使用して DATALINK ファイルを Data Links Manager サーバーとの間でコピーする場合は、 Data Links Manager Replication Daemon を使用しなければなりません。 この変数が YES に設定されている場合、DLFM_ASNCOPYD_PORT 変数を設定する必要があります。		

表 53. データ・リンク変数 (続き)

変数名	オペレーティング・システム	値
説明		
DLFM_ASNCOPYD_PORT	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: null 値: 任意の有効なポート番号
Data Links Manager Replication Daemon (DLFM_ASNCOPYD) がファイル複製要求のために listen する TCP/IP ポート番号を指定します。 DLFM_START_ASNCOPYD 変数が YES に設定されている場合、この変数を設定する必要があります。		
DLFM_NUM_ARCHIVE_SUBSYSTEMS	AIX、Windows NT、Windows 2000、Solaris オペレーティング環境	デフォルト: 2 値: 1 以上の数
所定の DLFM サーバーで実行する DLFM Copy Daemon プロセスの数を指定します。コピー・プロセスの数が大きくなればなるほど、リンク・ファイルをバックアップする際のスループットも大きくなります。ただし、この値は、リンク・ファイルを指定したアーカイブ領域にコピーするために利用可能な入出力チャネルの数に対応していなければなりません。この値が大きすぎると、消費されるシステム・リソースの量のために入出力並列処理の効果が弱くなってしまう場合があります。		
DLFM_AUTOSTART	AIX、Solaris オペレーティング環境	デフォルト: NO 値: YES、NO
オペレーティング・システムがリブートしたときはいつでも DLFM サーバーに自動開始させるかどうかを指定します。この変数は、ブート処理中に `/etc/inittab` ファイルから呼び出された `dlfsmount` スクリプトによってチェックされます。		

関連概念:

- 571 ページの『DB2 レジストリー変数と環境変数』

その他の変数

表 54. その他の変数

変数名	オペレーティング・システム	値
説明		
DB2ADMINSERVER	Windows および UNIX	デフォルト = null
DB2 Administration Server を指定します。		
DB2CLIINIPATH	すべて	デフォルト = null
DB2 CLI/ODBC 構成ファイル (db2cli.ini) のデフォルト・パスをオーバーライドし、クライアントの異なる位置を指定するために使用されます。ここで指定される値は、クライアント・システム上の有効なパスでなければなりません。		
DB2_COMMIT_ON_EXIT	UNIX	デフォルト = OFF 値: OFF/NO/0 または ON/YES/1

表 54. その他の変数 (続き)

変数名	オペレーティング・システム	値
説明		
UNIX プラットフォームでは、バージョン 8 より前の DB2 が、残った不完了トランザクションを正常アプリケーション出口でコミットしていました。バージョン 8 でこの動作が変更され、不完了トランザクションは出口でロールバックされるようになりました。このレジストリー変数を使用することにより、以前の動作に依存するアプリケーションを使用しているユーザーは、バージョン 8 でもそのアプリケーションを継続して使用できます。 このレジストリー変数はバージョン 10 で使用できなくなり、出口でのコミット動作はサポートされなくなるので注意してください。バージョン 8 より前に開発されたアプリケーションをこの機能に依存したまま使用し続けるか検討し、必要であればアプリケーションに適した明示的 COMMIT ステートメントを追加してください。このレジストリー変数をオンにする場合は、明示的には COMMIT を実行できない新規アプリケーションを出口の前にインプリメントしないように注意してください。 通常は、このレジストリー変数はデフォルト設定のままにしておきます。		
DB2DEFPREP	すべて	デフォルト = NO 値: ALL、YES または NO
DEFERRED_PREPARE プリコンパイル・オプションが利用可能になる前にプリコンパイルされたアプリケーションのために、このオプションの実行時の動作をシミュレートします。たとえば、DB2 V2.1.1 またはそれ以前のアプリケーションを DB2 V2.1.2 以降の環境で実行するときは、 DB2DEFPREP を使用して、望ましい「据え置き準備」動作を指示することができます。		
DB2_DJ_COMM	すべて	デフォルト = null 含めることができる値: libdb2drda.a、libdb2net8.a、libdb2informix.a、db2drda.dll、db2net8.dll、db2informix.a など。
データベース・マネージャーの始動時にロードされるラッパー・ライブラリーを指定します。この変数を指定すると、頻繁に使用されるラッパーをロードするときのランタイム・コストが減少します。他のオペレーティング・システムには別の値がサポートされています (Windows NT オペレーティング・システムには .dll 拡張子、 AIX オペレーティング・システムには .a 拡張子)。ライブラリー名は、プロトコルおよびオペレーティング・システムによって異なります。 この変数は、データベース・マネージャー・パラメーター FEDERATED が YES に設定されていない場合に無視されます。		
DB2_DJ_INI	すべて	デフォルト: • UNIX: db2_instance_directory/cfg/db2dj.ini • Windows: db2_install_directory¥cfg¥db2dj.ini

表 54. その他の変数 (続き)

変数名	オペレーティング・システム	値
説明		

フェデレーション構成ファイルの絶対パス名を指定します。たとえば、db2set DB2_DJ_INI=$HOME/sqllib/cfg/my_db2dj.ini のようにします。このファイルには、データ・ソース環境変数の設定値が含まれています。これらの環境変数は、Informix ラッパー、および DB2 Information Integrator によって提供されるラッパーによって使用されます。
 	
次に示すのは、フェデレーション構成ファイルの例です。
 	
`INFORMIXDIR=/informix/client_sdk`
`INFORMIXSERVER=inf93`
`ORACLE_HOME=/usr/oracle9i`
`SYBASE=/sybase/V12`
`SYBASE_OCS=OCS-12_5`
 	
db2dj.ini ファイルには、以下の制約事項が適用されます。
• *evname=value* の形式で入力しなければなりません。 evname は環境変数の名前で、value はその値です。
• 環境変数名の最大長は 255 バイトです。
• 環境変数値の最大長は 765 バイトです。
 	
この変数は、データベース・マネージャー・パラメーター FEDERATED が YES に設定されていない場合に無視されます。		

DB2DMNBCKCTLR	Windows NT	デフォルト = null 値: ? またはドメイン・ネーム
DB2 サーバーがバックアップ・ドメイン・コントローラーになっている場合、そのドメイン・ネームが分かれば、DB2DMNBCKCTLR=DOMAIN_NAME と設定します。 DOMAIN_NAME は大文字でなければなりません。ローカル・マシンがバックアップ・ドメイン・コントローラーになっているドメインを DB2 が判別するには、DB2DMNBCKCTLR=? と設定します。 DB2DMNBCKCTLR プロファイル変数を設定しなかったりブランクに設定したりすると、 DB2 は 1 次ドメイン・コントローラーで認証を実行します。 **注:** デフォルトでは、DB2 はバックアップ・ドメイン・コントローラーを使用しません。バックアップ・ドメイン・コントローラーは 1 次ドメイン・コントローラーと同期しないことがあり、セキュリティ情報が不足することがあるからです。 1 次ドメイン・コントローラーのセキュリティー・データベースが更新されたが、その変更内容がバックアップ・ドメイン・コントローラーに伝搬していない場合に、同期しなくなることがあります。この事態は、ネットワーク待ち時間が生じた場合やコンピューターのブラウザー・サービスが作動可能でない場合に起こることがあります。		
DB2_DOCHOST	すべて	デフォルト: http://publib.boulder.ibm.com/infocenter/db2help/ http://*hostname* (*hostname* は有効なホスト名または IP アドレス)
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ホスト名を指定します。「DB2 セットアップ」ウィザードで自動構成オプションを選択した場合には、 DB2 インフォメーション・センターのインストール中に、この変数は自動的に設定されます。		
DB2_DOCPORT	すべて	デフォルト: NULL 値: 任意の有効なポート番号
DB2 ヘルプ・システムが DB2 資料を表示するときに使用するポート番号を指定します。「DB2 セットアップ」ウィザードで自動構成オプションを選択した場合には、 DB2 インフォメーション・センターのインストール中に、この変数は自動的に設定されます。		

表 54. その他の変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2_EXTSECURITY	Windows プラットフォーム	デフォルト = ON 値: ON または OFF
DB2 システム・ファイルをロックすることにより、 DB2 への無許可アクセスを防ぎます。問題を未然に防ぐため、このレジストリー変数はオフにしないでください。		
DB2_ENABLE_LDAP	すべて	デフォルト = NO 値: YES または NO
Lightweight Directory Access Protocol (LDAP) を使用するかどうかを指定します。 LDAP は、ディレクトリー・サービスへのアクセス方式の 1 つです。		
DB2_FALLBACK	Windows NT	デフォルト = OFF 値: ON または OFF
この変数を使用することによって、フォールバック処理中に強制的にすべてのデータベース接続を切断できます。これは、Microsoft Cluster Server (MSCS) がある Windows NT 環境で、フェイルオーバー・サポートと一緒に使用されます。 DB2_FALLBACK が未設定、または OFF に設定されていて、フォールバックの間データベースが接続されている場合、 DB2 リソースをオフラインにすることはできません。 つまり、フォールバック処理は失敗します。		
DB2_FMP_COMM_HEAPSZ	Windows、AIX 以外のすべての UNIX	20MB または 10 の fenced ルーチンを実行するのに十分なスペース (大きいほう)
この変数は、ストアード・プロシージャーやユーザー定義関数の呼び出しのような、 fenced ルーチンの呼び出しに使用されるプールのサイズを指定します (4 KB ページ単位)。各 fenced ルーチンが使用するスペースは、 aslheapsz 構成パラメーターの値の 2 倍です。 システムで多くの fenced ルーチンを実行している場合には、この変数の値を増やす必要があるかもしれません。実行している fenced ルーチンがとても少ない場合には、変数の値を減らすことができます。 この値を 0 に設定すると設定なしと見なされ、結果として fenced ルーチンを呼び出すことはできません。このことは、自動データベース保守機能 (自動バックアップ、統計収集、および REORG) が fenced ルーチン・インフラストラクチャーに依存しているため、ヘルス・モニターと自動データベース保守機能が使用不能になるという意味でもあります。		
DB2_GRP_LOOKUP	Windows NT	デフォルト = null 値: LOCAL、DOMAIN
この変数を使うと、DB2 は、ユーザー・アカウントを妥当性検査する場所と、グループ・メンバー検索を実行する場所を判別できます。変数を LOCAL に設定すると、DB2 で常に DB2 サーバー上のグループを列挙し、ユーザー・アカウントを妥当性検査することができます。変数を DOMAIN に設定すると、ユーザー・アカウントが属する Windows NT ドメイン上のグループを DB2 で常に列挙し、ユーザー・アカウントを妥当性検査することができます。		
DB2_HADR_BUF_SIZE	すべて	デフォルト = 2*LOGBUFSZ

表 54. その他の変数 (続き)

変数名	オペレーティング・システム	値
説明		
この変数では、スタンバイ・ログ受信バッファー・サイズをログ・ページの単位で指定します。この変数を設定しないと、 DB2 はプライマリー側の LOGBUFSZ 構成パラメーター値の 2 倍の値をスタンバイ受信バッファー・サイズに使用します。この変数はスタンバイ・インスタンス内で設定します。プライマリー・データベースは、この変数を無視します。 HADR 同期モード (HADR_SYNCMODE データベース構成パラメーター) が ASYNC に設定されている場合、データ転送中に、スタンバイ側の処理が遅いと、プライマリー上での送信操作が停止し、それによってプライマリー上のトランザクション処理が待たされることがあります。デフォルトより大きいログ受信バッファーは、スタンバイ・データベース上に構成することができ、このバッファーにはより多くの未処理ログ・データが入ります。これにより、プライマリーで処理するトランザクションが待たされず、スタンバイがログ・データを取り込むよりも、プライマリーがログ・データを生成する方が速い状態が短時間起こりえます。		
DB2LDAP_BASEDN	すべて	デフォルト = null 値: 任意の有効なベース・ドメイン名。
LDAP ディレクトリーのベース・ドメイン名を指定します。		
DB2LDAPCACHE	すべて	デフォルト = YES 値: YES または NO

LDAP キャッシュを使用可能にするかどうかを指定します。このキャッシュは、ローカル・マシン上のデータベース、ノード、および DCS ディレクトリーのカタログを作成するのに使用します。
 	
確実にキャッシュ内の項目を最新のものにするには、以下を行います。
`REFRESH LDAP DB DIR`
`REFRESH LDAP NODE DIR`
 	
これらのコマンドは、データベース・ディレクトリーおよびノード・ディレクトリーの項目を更新し、正しくない項目は除去します。		

DB2LDAP_CLIENT_PROVIDER	Windows	デフォルト = null (使用可能であれば Microsoft が使用されます。そうでなければ IBM が使用されます。) 値: IBM または Microsoft
Windows 環境で稼働している場合、 DB2 は LDAP ディレクトリーへアクセスするために、 Microsoft LDAP クライアントか IBM LDAP クライアントのいずれかの使用をサポートしています。このレジストリー変数は、 DB2 が使用する LDAP クライアントを明示的に選択するのに使用します。 **注:** このレジストリー変数の現行値を表示するには、以下の `db2set` コマンドを使用します。 `db2set DB2LDAP_CLIENT_PROVIDER`		
DB2LDAPHOST	すべて	デフォルト = null 値: 任意の有効なホスト名
LDAP ディレクトリーの位置のホスト名を指定します。		
DB2LDAP_KEEP_CONNECTION	すべて	デフォルト = YES 値: YES、NO

表 54. その他の変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2 がその内部 LDAP 接続ハンドルをキャッシュするかどうかを指定します。この変数を NO に設定すると、DB2 は LDAP 接続ハンドルをディレクトリー・サーバーにキャッシュしません。この場合、パフォーマンスにマイナスの影響を及ぼす可能性がありますが、同時にアクティブな、ディレクトリー・サーバーへの LDAP クライアント接続の数を最少にすることが必要である場合は、DB2LDAP_KEEP_CONNECTION を NO に設定するとよいでしょう。 最高のパフォーマンスを得るために、この変数はデフォルトで YES に設定されています。 DB2LDAP_KEEP_CONNECTION レジストリー変数は、グローバル・レベル・プロファイル・レジストリー変数として LDAP にインプリメントされているため、次のように **db2set** コマンドに -gl オプションを指定して、この変数を設定しなければなりません。 `db2set -gl DB2LDAP_KEEP_CONNECTION=NO`		
DB2LDAP_SEARCH_SCOPE	すべて	デフォルト = DOMAIN 値: LOCAL、DOMAIN、GLOBAL
Lightweight Directory Access Protocol (LDAP) のパーティションまたはドメインで検出された情報の検索範囲を指定します。「LOCAL」を指定すると、LDAP ディレクトリー内の探索は使用不可になります。「DOMAIN」を指定すると、現行ディレクトリー・パーティションの LDAP 内だけを探索します。「GLOBAL」を指定すると、オブジェクトが見つかるまで全ディレクトリー・パーティション内の LDAP を探索します。		
DB2_LOAD_COPY_NO_OVERRIDE	すべて	デフォルト: NONRECOVERABLE 値: COPY YES、NONRECOVERABLE
この変数は、任意の LOAD COPY NO を、変数の値に応じて、 LOAD COPY YES または NONRECOVERABLE のいずれかに変換します。この変数は HADR プライマリー・データベースと標準 (非 HADR) データベースに適用可能です。 HADR スタンバイ・データベース上では無視されます。 HADR プライマリー・データベース上では、この変数が設定されていないと、 LOAD COPY NO が LOAD NONRECOVERABLE に変換されます。この変数の値は、COPY YES 文節と同じ構文を使用して、リカバリー不能ロードまたはコピー宛先のいずれかを指定します。		
DB2LOADREC	すべて	デフォルト = null
ロールフォワード時にロード・コピーの位置をオーバーライドするために使用されます。ユーザーがロード・コピーの物理的な位置を変更している場合には、ロールフォワードを出す前に DB2LOADREC を設定しておく必要があります。		
DB2LOCK_TO_RB	すべて	デフォルト = null 値: STATEMENT
ロック・タイムアウトの場合にトランザクション全体をロールバックするか、または現行のステートメントだけをロールバックするかを指定します。 DB2LOCK_TO_RB が `STATEMENT` に設定されていると、ロック・タイムアウトによってロールバックされるのは、現行のステートメントだけになります。 その他の設定では、トランザクション全体がロールバックされます。		
DB2NOEXITLIST	すべて	デフォルト = OFF 値: ON または OFF

表 54. その他の変数 (続き)

変数名	オペレーティング・システム	値
説明		
定義される場合、この変数は、 DB2 がアプリケーションに出口リスト・ハンドラーをインストールしないこと、かつ COMMIT を実行しないことを示します。普通、DB2 はアプリケーションに処理出口リスト・ハンドラーをインストールし、アプリケーションが正常に終了するとその出口リスト・ハンドラーが COMMIT 操作を実行します。 アプリケーションが DB2 ライブラリーを動的にロードし、そのライブラリーをアンロードしてからアプリケーションが終了する場合、出口リスト・ハンドラーのルーチンは、既にアプリケーションからアンロードされているので、そのハンドラーの呼び出しは失敗します。この方法でアプリケーションが操作する場合、 DB2NOEXITLIST 変数を設定し、アプリケーションが必ず必須の COMMIT すべてを明示的に呼び出すことを確認します。		
DB2OLDEVMON	すべて	値: イベント・モニター名をコンマで区切って指定します。 *evmon1*, *evmon2*, ...
バージョン 6 以前の形式でデータを書き込むイベント・モニターの名前を指定します。 DB2 バージョン 6 で、データ構造の記述を含むデータ・ストリームがファイルやパイプへのシステム・モニター出力の標準形式になりました。バージョン 6 以前は、システム・モニター・データが固定データ構造で戻されていました。		
DB2REMOTEPREG	Windows NT	デフォルト = null 値: 任意の有効な Windows NT マシン名
DB2 インスタンス・プロファイルおよび DB2 インスタンスの Win32 登録リストが入っているリモート・マシン名を指定します。 DB2REMOTEPREG の値は、 DB2 のインストール後にただ一度だけ設定し、変更すべきではありません。この変数の使用には十分な注意が必要です。		
DB2ROUTINE_DEBUG	AIX および Windows NT	デフォルト = OFF 値: ON、OFF
Java ストアード・プロシージャー用のデバッグ機能を使用可能にするかどうかを指定します。 Java ストアード・プロシージャーをデバッグしない場合は、デフォルト OFF を使用します。デバッグを使用可能にすると、パフォーマンス上の影響があります。		
DB2SATELLITEID	すべて	デフォルト = null 値: サテライト制御データベースで宣言されている有効なサテライト ID
サテライトが同期するときに、サテライト制御サーバーに渡されるサテライト ID を指定します。この変数に値が指定されない場合は、ログオン ID がサテライト ID として使用されます。		
DB2SORT	全、ただしサーバーのみ	デフォルト = null
ロード・ユーティリティーが実行時にロードするライブラリーの位置を指定します。このライブラリーには、索引付きデータのソートに使用される関数の入り口点が入っています。表索引の生成時に LOAD ユーティリティーとともにベンダー提供のソート用製品を利用するときは、 DB2SORT を使用します。提供されるパスは、データベース・サーバーとの関係で表される必要があります。		
DB2SYSTEM	Windows および UNIX	デフォルト = null

表 54. その他の変数 (続き)

変数名	オペレーティング・システム	値
説明		
DB2 サーバー・システムを識別するためにユーザーおよびデータベース管理者が使用する名前を指定します。可能であれば、この名前はご使用のネットワーク内でユニークである必要があります。 この名前は、コントロール・センターのオブジェクト・ツリーのシステム・レベルで表示されるので、管理者がコントロール・センターから管理できるサーバー・システムを識別する場合に役立ちます。 クライアント構成アシスタントの「ネットワークの検索」機能を使用すると、 DB2 ディスカバリーはこの名前を戻し、その結果がオブジェクト・ツリーにシステム・レベルで表示されます。 この名前は、ユーザーがアクセスしたいデータベースが入っているシステムを識別するのに役立ちます。DB2SYSTEM の値は、インストール時に次のように設定されます。 • Windows NT では、セットアップ・プログラムが Windows システムに指定されているコンピューター名と等しい名前を設定します。 • UNIX システムでは、 UNIX システムの TCP/IP ホスト名に等しい名前に設定されます。		
DB2_VENDOR_INI	AIX、HP-UX、Solaris オペレーティング環境、および Windows	デフォルト = null 値: 任意の有効なパスおよびファイル。
すべてのベンダー特定の環境設定を含むファイルを示します。データベース・マネージャーが開始した時に、値が読み取られます。		
DB2_XBSA_LIBRARY	AIX、HP-UX、Solaris オペレーティング環境、および Windows	デフォルト = null 値: 任意の有効なパスおよびファイル。

ベンダーの提供する XBSA ライブラリーを示します。 AIX で、共用オブジェクトが shr.o という名前でない場合は、設定にそのオブジェクトを組み込む必要があります。 HP-UX、Solaris オペレーティング環境、および Windows NT では、共用オブジェクト名は必要ありません。たとえば、Legato's NetWorker Business Suite Module for DB2 を使用するには、レジストリー変数を次のように設定します。
 	
`db2set DB2_XSBA_LIBRARY="/usr/lib/libxdb2.a(bsashr10.o)"`
 	
XBSA インターフェースは、BACKUP DATABASE または RESTORE DATABASE コマンドから呼び出すことができます。例:
 	
`db2 backup db sample use XBSA`
`db2 restore db sample use XBSA`		

関連概念:

• 571 ページの『DB2 レジストリー変数と環境変数』

付録 B. Explain 表

Explain 表

Explain 表は、Explain 機能起動時のアクセス・プランを格納する表です。 Explain 表は、Explain 機能を呼び出す前に作成しておく必要があります。これを作成するには、文書化された表定義を使用します。または、「sqllib」ディレクトリーの「misc」サブディレクトリー内の EXPLAIN.DDL ファイルに入っている、コマンド行プロセッサー (CLP) スクリプトのサンプルを呼び出すことによっても作成できます。このスクリプトを呼び出すには、 Explain 表を必要とするデータベースに接続して、以下のコマンドを発行します。

```
db2 -tf EXPLAIN.DDL
```

Explain 機能によって Explain 表に書き込んでも、トリガー、参照制約、チェック制約が活動化されることはありません。たとえば、挿入トリガーが EXPLAIN_INSTANCE 表に定義されており、かつ適切なステートメントが Explain されているとしても、トリガーは活動化されません。

関連資料:

- 616 ページの『EXPLAIN_ARGUMENT 表』
- 622 ページの『EXPLAIN_OBJECT 表』
- 625 ページの『EXPLAIN_OPERATOR 表』
- 627 ページの『EXPLAIN_PREDICATE 表』
- 632 ページの『EXPLAIN_STREAM 表』
- 634 ページの『ADVISE_INDEX 表』
- 642 ページの『ADVISE_WORKLOAD 表』
- 620 ページの『EXPLAIN_INSTANCE 表』
- 629 ページの『EXPLAIN_STATEMENT 表』
- 637 ページの『ADVISE_INSTANCE 表』
- 638 ページの『ADVISE_MQT 表』
- 640 ページの『ADVISE_PARTITION 表』
- 641 ページの『ADVISE_TABLE 表』

EXPLAIN_ARGUMENT 表

EXPLAIN_ARGUMENT 表は、個々のオペレーターにユニークな特性がある場合、それを示します。

表 55. EXPLAIN_ARGUMENT 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	×	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	×	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	×	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	×	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	×	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	×	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	×	FK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	×	FK	この Explain 情報に関連するパッケージのセクション番号。
OPERATOR_ID	INTEGER	×	×	この照会内のオペレーターのユニークな ID。
ARGUMENT_TYPE	CHAR(8)	No	×	このオペレーターの引き数のタイプ。
ARGUMENT_VALUE	VARCHAR(1024)	Yes	No	このオペレーターの引き数の値。値が LONG_ARGUMENT_VALUE にある場合は NULL。
LONG_ARGUMENT_VALUE	CLOB(2M)	Yes	No	このオペレーターの引き数の値。(テキストが ARGUMENT_VALUE に収まらない場合。) 値が ARGUMENT_VALUE にある場合は NULL。

表 56. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	部分集約標識。
BITFLTR	TRUE FALSE	ハッシュ結合でビット・フィルターを使ってパフォーマンスを向上させる。
BLD_LEVEL	DB2 ビルド ID	ソース・コード・バージョンの内部識別ストリング。
BLKLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT SHARE NONE SHARE UPDATE	ブロック・レベル・ロック意図。
CSERQY	TRUE FALSE	リモート照会が共通副次式。
CSETEMP	TRUE FALSE	共通副次式に一時表を使うかどうかを示すプラグ。
DIRECT	TRUE	直接 FETCH の標識。
DSTSEVER	サーバー名	宛先 (配送元) サーバー。

表 56. *ARGUMENT_TYPE* および *ARGUMENT_VALUE* 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
DUPLWARN	TRUE FALSE	警告複写標識。
EARLYOUT	LEFT RIGHT NONE	EARLYOUT 標識。
ENVVAR	このタイプの各行には以下のものが含まれます。 • 環境変数名 • 環境変数値	オプティマイザーに影響する環境変数
FETCHMAX	IGNORE INTEGER	FETCH オペレーターの MAXPAGES 引き数の値をオーバーライドします。
GREEDY	TRUE	オプティマイザーが貪欲型アルゴリズムを使用してアクセスをプランしたことを示す。
GROUPBYC	TRUE FALSE	Group By 列が与えられているかどうか。
GROUPBYN	Integer	比較列の数。
GROUPBYR	このタイプの各行には以下のものが含まれます。 • group by 文節内の列の順序値 (後に、コロンとスペースが続く) • 列の名前	Group By 要件。
INNERCOL	このタイプの各行には以下のものが含まれます。 • 配列内の列の順序値 (後に、コロンとスペースが続く) • 列の名前 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	内部配列の列。
ISCANMAX	IGNORE INTEGER	ISCAN オペレーターの MAXPAGES 引き数の値をオーバーライドします。
JN_INPUT	INNER OUTER	オペレーターが内部または外部のどちらの結合を送るオペレーターであるかを示す。
LISTENER	TRUE FALSE	Listener 表キューの標識。
MAXPAGES	ALL NONE INTEGER	プリフェッチのための最大ページ数。
MAXRIDS	NONE INTEGER	個々のリスト・プリフェッチ要求に組み込まれる最大行 ID。
NUMROWS	INTEGER	ソートされるべき行の数。
ONEFETCH	TRUE FALSE	1 つの取り出し標識。

表 56. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
OUTERCOL	このタイプの各行には以下のものが含まれます。 • 配列内の列の順序値 (後に、コロンとスペースが続く) • 列の名前 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	外部配列の列。
OUTERJN	LEFT RIGHT FULLLEFT (ANTI) RIGHT (ANTI)	外部結合の標識。
PARTCOLS	列の名前	オペレーターのパーティション列。
PREFETCH	LIST NONE SEQUENTIAL	プリフェッチ有資格属性のタイプ。
RMTQTEXT	照会テキスト	リモート照会テキスト。
RNG_PROD	関数名	拡張索引アクセスのための範囲生成関数。
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	行ロック意図。
ROWWIDTH	INTEGER	ソートされる行の幅。
RSUFFIX	照会テキスト	リモート SQL の接尾部。
SCANDIR	FORWARD REVERSE	スキャンの方向。
SCANGRAN	INTEGER	パーティション内並列処理、パーティション内並列処理のスキャンの細分性。 SCANUNIT の単位で表される。
SCANTYPE	LOCAL PARALLEL	パーティション内並列処理、索引または表のスキャン。
SCANUNIT	ROW PAGE	パーティション内並列処理、スキャンの細分性の単位。
SHARED	TRUE	パーティション内並列処理、共用 TEMP 標識。
SLOWMAT	TRUE FALSE	低速マテリアライズ・フラグ。
SNGLPROD	TRUE FALSE	パーティション内並列処理、単一エージェントによるソートまたは一時作成。
SORTKEY	このタイプの各行には以下のものが含まれます。 • キー内の列の順序値 (後に、コロンとスペースが続く) • 列の名前 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	ソート・キーの列。

表 56. *ARGUMENT_TYPE* および *ARGUMENT_VALUE* 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	パーティション内並列処理、ソート・タイプ。
SRCSEVER	サーバー名	ソース (配送先) サーバー。
STREAM	TRUE FALSE	リモート・ソースがストリーミング。
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	表ロック意図。
TEMPSIZE	INTEGER	一時表のページ・サイズ。
TQDEGREE	INTEGER	パーティション内並列処理、表キューにアクセスするサブエージェントの数。
TQMERGE	TRUE FALSE	マージする (ソート済み) 表キューの標識。
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	表キューの読み取り特性。
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	表キューの送信特性。
TQTYPE	LOCAL	パーティション内並列処理、表キュー。
TRUNCSRT	TRUE	切り捨てソート (作成される行の数を制限する)。
UNIQUE	TRUE FALSE	固有性の標識。
UNIQKEY	このタイプの各行には以下のものが含まれます。 • キー内の列の順序値 (後に、コロンとスペースが続く) • 列の名前	ユニーク・キーの列。
VOLATILE	TRUE	Volatile 表。

EXPLAIN_INSTANCE 表

EXPLAIN_INSTANCE 表は、すべての Explain 情報用の主コントロール表です。Explain 表中のデータの各行は、この表内のあるユニークな 1 行に明示的にリンクされます。 EXPLAIN_INSTANCE 表は、Explain 対象の SQL ステートメントのソースに関する基本情報、および Explain 機能の環境に関する情報を提供します。

表 57. EXPLAIN_INSTANCE 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	PK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	PK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	PK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	PK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	PK	Explain 要求のソースのバージョン。
EXPLAIN_OPTION	CHAR(1)	No	No	この要求に関して要求された Explain 情報の内容を示します。 可能な値は以下のとおりです。 **P** PLAN SELECTION
SNAPSHOT_TAKEN	CHAR(1)	No	No	この要求に関して Explain スナップショットが取られたかどうかを示します。 可能な値は以下のとおりです。 **Y** はい。1 つまたは複数の Explain スナップショットが取られ、 EXPLAIN_STATEMENT 表に保管されました。正規の Explain 情報もキャプチャーされました。 **N** Explain スナップショットは取られませんでした。正規の Explain 情報がキャプチャーされました。 **O** Explain スナップショットだけが取られました。正規の Explain 情報はキャプチャーされませんでした。
DB2_VERSION	CHAR(7)	No	No	この Explain 要求を処理した DB2 Universal Database の製品リリース番号。形式は vv.rr.m です。ただし、 **vv** バージョン番号 **rr** リリース番号 **m** 保守リリース番号
SQL_TYPE	CHAR(1)	No	No	Explain インスタンスが静的と動的のどちらの SQL に関するものであったかどうかを示します。 可能な値は以下のとおりです。 **S** 静的 SQL **D** 動的 SQL
QUERYOPT	INTEGER	No	No	Explain 呼び出しの時点で SQL コンパイラーが使用する照会最適化クラスを示します。値は、Explain 中の SQL ステートメントについて SQL コンパイラーが実行したのが、どのレベルの照会最適化であるかを示します。

表 57. EXPLAIN_INSTANCE 表 (続き)： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
BLOCK	CHAR(1)	No	No	SQL ステートメントのコンパイル時に使用されたカーソルのブロッキング・タイプを示します。詳細については、SYSCAT.PACKAGES の BLOCK 列を参照してください。 可能な値は以下のとおりです。 **N** ブロッキングなし **U** 確定カーソルのブロック **B** すべてのカーソルのブロック
ISOLATION	CHAR(2)	No	No	SQL ステートメントのコンパイル時に使用された分離の種類を示します。詳細については、SYSCAT.PACKAGES の ISOLATION 列を参照してください。 可能な値は以下のとおりです。 **RR** 反復可能読み取り **RS** 読み取り固定 **CS** カーソル固定 **UR** 非コミット読み取り
BUFFPAGE	INTEGER	いいえ	いいえ	Explain の呼び出しの時点で設定された BUFFPAGE データベース構成の値が含まれています。
AVG_APPLS	INTEGER	いいえ	いいえ	Explain 呼び出し時に、 AVG_APPLS データベース構成パラメーターの値が入れられます。
SORTHEAP	INTEGER	いいえ	いいえ	Explain の呼び出しの時点で設定された SORTHEAP データベース構成の値が含まれています。
LOCKLIST	INTEGER	いいえ	いいえ	Explain の呼び出しの時点で設定された LOCKLIST データベース構成の値が含まれています。
MAXLOCKS	SMALLINT	いいえ	いいえ	Explain の呼び出しの時点で設定された MAXLOCKS データベース構成の値が含まれています。
LOCKS_AVAIL	INTEGER	No	No	オプティマイザーによって各ユーザーごとに使用可能と見なされるロックの数が含まれています。 (LOCKLIST および MAXLOCKS から派生したもの。)
CPU_SPEED	DOUBLE	No	いいえ	Explain の呼び出しの時点で設定された CPUSPEED データベース・マネージャー構成設定の値が含まれています。
REMARKS	VARCHAR(254)	Yes	いいえ	ユーザーが入力したコメント。
DBHEAP	INTEGER	いいえ	いいえ	Explain 呼び出し時に DBHEAP データベース構成設定値が入れられます。
COMM_SPEED	DOUBLE	いいえ	いいえ	Explain 呼び出し時に COMM_BANDWIDTH データベース構成設定値が入れられます。
PARALLELISM	CHAR(2)	No	No	可能な値は以下のとおりです。 • N = 並列処理なし • P = パーティション内並列処理 • IP = パーティション間並列処理 • BP = パーティション内並列処理とパーティション間並列処理
DATAJOINER	CHAR(1)	No	No	可能な値は以下のとおりです。 • N = 非フェデレーテッド・システム・プラン • Y = フェデレーテッド・システム・プラン

EXPLAIN_OBJECT 表

EXPLAIN_OBJECT 表は、 SQL ステートメントを満たすために生成されるアクセス・プランが必要とするデータ・オブジェクトを指定します。

表 58. EXPLAIN_OBJECT 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	×	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	×	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	×	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	×	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	×	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	×	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	×	FK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	×	FK	この Explain 情報に関連するパッケージのセクション番号。
OBJECT_SCHEMA	VARCHAR(128)	×	×	このオブジェクトが属しているスキーマ。
OBJECT_NAME	VARCHAR(128)	×	×	オブジェクトの名前。
OBJECT_TYPE	CHAR(2)	×	×	オブジェクトのタイプの記述ラベル。
CREATE_TIME	TIMESTAMP	○	×	オブジェクトの作成時刻。表関数の場合は NULL。
STATISTICS_TIME	TIMESTAMP	Yes	No	このオブジェクトを最後に更新した時刻。このオブジェクトに統計がない場合は NULL 値。
COLUMN_COUNT	SMALLINT	No	No	このオブジェクトの列数。
ROW_COUNT	INTEGER	No	No	このオブジェクトの行数の見積もり。
WIDTH	INTEGER	No	No	オブジェクトの平均幅 (バイト数)。索引の場合は -1 に設定します。
PAGES	INTEGER	No	No	オブジェクトがバッファー・プールで占有するページ数の見積もり。表関数には、-1 に設定します。
DISTINCT	CHAR(1)	No	No	オブジェクト内の行が固有かどうか (つまり、重複があるかどうか) を示します。 可能な値は以下のとおりです。 **Y** Yes **N** No
TABLESPACE_NAME	VARCHAR(128)	Yes	No	このオブジェクトが保管されている表スペースの名前。表スペースが関係していない場合は、NULL 値に設定する。
OVERHEAD	DOUBLE	No	No	指定された表スペースにランダム入出力を 1 回行うためのオーバーヘッドの見積合計 (ミリ秒)。コントローラー・オーバーヘッド、ディスク・シーク、および待ち時間が含まれます。表スペースが関係していない時は -1 に設定します。
TRANSFER_RATE	DOUBLE	No	No	指定の表スペースからデータ・ページを読み取るための時間の見積もり (ミリ秒単位)。表スペースが関係していない時は -1 に設定します。

表 58. *EXPLAIN_OBJECT 表 (続き)*： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
PREFETCHSIZE	INTEGER	No	No	プリフェッチの実行時に読み取るデータ・ページの数。表関数には、-1 に設定します。
EXTENTSIZE	INTEGER	No	No	データ・ページを単位とするエクステント・サイズ。表スペースの中のコンテナーにこの数のページが書き込まれたら、次のコンテナーに切り替わります。表関数には、-1 に設定します。
CLUSTER	DOUBLE	No	No	索引とのデータ・クラスタリングの程度。 >= 1 の場合は CLUSTERRATIO。 0 以上かつ 1 より小さい場合、これは CLUSTERFACTOR。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
NLEAF	INTEGER	No	No	この索引オブジェクトの値が占めるリーフ・ページの数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
NLEVELS	INTEGER	No	No	この索引オブジェクトのツリー内の索引レベルの数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FULLKEYCARD	BIGINT	No	No	この索引オブジェクトに含まれる個別フル・キー値の数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
OVERFLOW	INTEGER	No	No	表内のオーバーフロー・レコードの合計数。索引、表関数について、またはこの統計が使用不可である場合は、-1 に設定します。
FIRSTKEYCARD	BIGINT	No	No	第 1 キーの値の数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FIRST2KEYCARD	BIGINT	No	No	索引の最初の {2,3,4} 列を使用する最初のキー値の種類数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FIRST3KEYCARD	BIGINT	No	No	
FIRST4KEYCARD	BIGINT	No	No	
SEQUENTIAL_PAGES	INTEGER	No	No	索引キーの順序でディスクに存在し、それらの間に大きなギャップがないか、わずかなギャップしかないリーフ・ページの数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
DENSITY	INTEGER	No	No	索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表現される (0〜100 の整数)。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
STATS_SRC	CHAR(1)	No	No	統計のソースを示します。単一のノードからの場合は 1 に設定します。
AVERAGE_SEQUENCE_GAP	DOUBLE	No	No	シーケンス間のギャップ。
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE	No	No	索引を使用してフェッチするときの、シーケンス間のギャップ。
AVERAGE_SEQUENCE_PAGES	DOUBLE	No	No	順次にアクセスできる索引ページの平均数。
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE	No	No	索引を使用してフェッチする際の、順次にアクセスできる表ページの平均数。
AVERAGE_RANDOM_PAGES	DOUBLE	No	No	順次ページ・アクセスの間のランダム索引ページの平均数。
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE	No	No	索引を使用してフェッチする際の、順次ページ・アクセスの間のランダム表ページの平均数。

表 58. EXPLAIN_OBJECT 表 (続き): PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
NUMRIDS	BIGINT	No	No	索引内の行 ID の合計数。
NUMRIDS_DELETED	BIGINT	No	No	索引内の疑似削除された行 ID の合計数。
NUM_EMPTY_LEAFS	BIGINT	No	No	索引内の空白リーフ・ページの合計数。
ACTIVE_BLOCKS	BIGINT	No	No	表内のアクティブなマルチディメンション・クラスタリング (MDC) ブロックの合計数。

表 59. 可能な OBJECT_TYPE 値

値	説明
IX	索引
TA	表
TF	表関数

EXPLAIN_OPERATOR 表

EXPLAIN_OPERATOR 表は、 SQL コンパイラーが SQL ステートメントを満たすために必要とするすべてのオペレーターを格納します。

表 60. EXPLAIN_OPERATOR 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	×	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	×	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	×	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	×	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	×	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	×	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	×	FK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	×	FK	この Explain 情報に関連するパッケージのセクション番号。
OPERATOR_ID	INTEGER	×	×	この照会内のオペレーターのユニークな ID。
OPERATOR_TYPE	CHAR(6)	No	No	オペレーターのタイプの記述ラベル。
TOTAL_COST	DOUBLE	No	No	このオペレーターに至るまで (このオペレーターを含む) の、選択したアクセス・プランの実行にかかる合計コスト (timeron 単位) の累積の見積もり。
IO_COST	DOUBLE	No	No	このオペレーターに至るまで (このオペレーターを含む) の、選択したアクセス・プランの実行にかかる入出力コスト (データ・ページの入出力単位) の累積の見積もり。
CPU_COST	DOUBLE	No	No	選択したアクセス・プランを実行するとき、またこのオペレーターを含めるときにかかる CPU コスト (命令数) の累積の見積もり。
FIRST_ROW_COST	DOUBLE	No	No	アクセス・プランへの 1 行目を取り出すとき、またこのオペレーターを含めるときにかかる累積合計(timeron 数) の見積もり。この値には、必要な初期オーバーヘッドが含まれます。
RE_TOTAL_COST	DOUBLE	No	No	このオペレーターに至るまで (このオペレーターを含む) の、選択したアクセス・プランの次の行の取り出しにかかるコスト (timeron 単位) の累積の見積もり。
RE_IO_COST	DOUBLE	No	No	このオペレーターに至るまで (このオペレーターを含む) の、選択したアクセス・プランの次の行の取り出しにかかる入出力コスト (データ・ページの入出力単位) の累積の見積もり。
RE_CPU_COST	DOUBLE	No	No	このオペレーターに至るまで (このオペレーターを含む) の、選択したアクセス・プランの次の行のフェッチにかかる CPU コスト (命令数) の累積の見積もり。
COMM_COST	DOUBLE	No	No	このオペレーターに至るまで (このオペレーターを含む) の、選択したアクセス・プランの実行にかかる通信コスト (TCP/IP フレーム単位) の累積の見積もり。

表 60. EXPLAIN_OPERATOR 表 (続き)： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
FIRST_COMM_COST	DOUBLE	No	No	このオペレーターに至るまで (このオペレーターを含む) の、選択したアクセス・プランの最初の行の取り出しにかかる通信コスト (TCP/IP フレーム単位) の累積の見積もり。この値には、必要な初期オーバーヘッドが含まれます。
BUFFERS	DOUBLE	No	No	このオペレーターとその入力に必要なバッファーの見積もり。
REMOTE_TOTAL_COST	DOUBLE	No	No	リモート・データベース操作の実行にかかる合計コスト (timeron 単位) の累積の見積もり。
REMOTE_COMM_COST	DOUBLE	No	No	このオペレーターに至るまで (このオペレーターを含む) の、選択したリモート・アクセス・プランの実行にかかる通信コストの累積の見積もり。

表 61. OPERATOR_TYPE の値

値	説明
DELETE	削除
FETCH	取り出し
FILTER	行フィルター
GENROW	行の生成
GRPBY	グループ化
HSJOIN	ハッシュ結合
INSERT	挿入
IXAND	動的ビットマップ索引 Anding
IXSCAN	索引スキャン
MSJOIN	マージ・スキャン結合
NLJOIN	ネスト・ループ結合
RETURN	結果
RIDSCN	行 ID (RID) スキャン
SHIP	リモート・システムへの照会の配送
SORT	ソート
TBSCAN	表スキャン
TEMP	一時表作成
TQ	表キュー
UNION	共用体
UNIQUE	重複の除去
UPDATE	更新

EXPLAIN_PREDICATE 表

EXPLAIN_PREDICATE は、特定のオペレーターによって適用される述部を指定します。

表 62. EXPLAIN_PREDICATE 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	×	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	×	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	×	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	×	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	×	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	×	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	×	FK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	×	FK	この Explain 情報に関連するパッケージのセクション番号。
OPERATOR_ID	INTEGER	×	×	この照会内のオペレーターのユニークな ID。
PREDICATE_ID	INTEGER	×	×	特定のオペレーターのための述部のユニークな ID。
HOW_APPLIED	CHAR(5)	No	No	特定のオペレーターによって述部がどのように使用されているか。
WHEN_EVALUATED	CHAR(3)	No	No	この述部で使用される副照会をいつ評価するかの指示。 可能な値は以下のとおりです。 **ブランク** この述部には副照会が含まれません。 **EAA** この述部で使用される副照会は、適用時に評価されます (EAA)。つまり、指定のオペレーターによって行が処理されるたびに、述部が適用されるので副照会が再評価されます。 **EAO** この述部で使用される副照会は、オープン時に評価されます (EAO)。つまり、指定のオペレーターに対して副照会は 1 回だけ再評価され、結果はそれぞれの行へ述部を適用する際に再利用されます。 **MUL** この述部の副照会のタイプは複数あります。
RELOP_TYPE	CHAR(2)	No	No	この述部で使用される関係オペレーターのタイプ。
SUBQUERY	CHAR(1)	No	No	この述部に対して、副照会からのデータ・ストリームが要求されているか。複数の副照会ストリームが要求されることがあります。 可能な値は以下のとおりです。 **N** 副照会ストリームは要求されていない **Y** 1 つ以上の副照会ストリームが要求されている
FILTER_FACTOR	DOUBLE	No	No	この述部が修飾する小数部の行数の見積もり。

表 62. EXPLAIN_PREDICATE 表 (続き): PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
PREDICATE_TEXT	CLOB(2M)	Yes	No	SQL ステートメントの内部表示から再作成された述部のテキスト。 ステートメントのコンパイル時に使用する場合には、これにホスト変数、特殊レジスター、またはパラメーター・マーカーも含まれます。 この値は、ステートメントが DBADM 権限を持つユーザーによって実行された場合にのみ、EXPLAIN_PREDICATE 表にダンプされます。 DBADM 権限を持たない他のユーザーはこの表に値をダンプできませんが、表にある値はこれらのユーザーも表示することができます。 使用不可の場合は NULL。

表 63. 可能な HOW_APPLIED 値

値	説明
BSARG	各ブロックにつき 1 つの SARGable 述部として評価された。
JOIN	結合に使用した。
RESID	残余述部として評価された。
SARG	索引またはデータ・ページの SARGable 述部として評価された。
START	開始条件として使用された。
STOP	停止条件として使用された。

表 64. 可能な RELOP_TYPE 値

値	説明
ブランク	Not Applicable
EQ	Equals
GE	Greater Than or Equal
GT	Greater Than
IN	In list
LE	Less Than or Equal
LK	Like
LT	Less Than
NE	Not Equal
NL	Is Null
NN	Is Not Null

EXPLAIN_STATEMENT 表

EXPLAIN_STATEMENT 表には、さまざまなレベルの Explain 情報に関する SQL ステートメントのテキストが含まれます。この表には、ユーザーが入力した元の SQL ステートメントと、その SQL ステートメントを満たすアクセス・プランを選択するのに (オプティマイザーで) 使用されるバージョンとが保管されます。後のバージョンは、書き直されているか、 SQL コンパイラーで判別された追加の述部によって拡張されているか、またはその両方であるので、元のバージョンとはあまり類似していないことがあります。

表 65. EXPLAIN_STATEMENT 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	PK、FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	PK、FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	PK、FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	PK、FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	PK	この行に関連する Explain 情報のレベル。 有効な値は次のとおりです。 **O** 元のテキスト (ユーザーが入力したもの) **P** PLAN SELECTION
STMTNO	INTEGER	いいえ	PK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。動的 Explain SQL ステートメントの場合は 1。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS カタログ・ビューで使用されているものと同じです。
SECTNO	INTEGER	いいえ	PK	パッケージ内のセクションのうち、この SQL ステートメントを含むもののセクション番号です。動的 Explain SQL ステートメントの場合、これは、実行時にこのステートメントのセクションを保持するのに使用されるセクション番号です。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS カタログ・ビューで使用されているものと同じです。
QUERYNO	INTEGER	いいえ	いいえ	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントでは STMTNO の値で、動的 SQL ステートメントでは 1 です。

表 65. EXPLAIN_STATEMENT 表 (続き): PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
QUERYTAG	CHAR(20)	いいえ	いいえ	Explain 対象の各 SQL ステートメントの ID タグ。 CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLP」です。 CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLI」です。それ以外の場合、使用されるデフォルト値はブランクです。
STATEMENT_TYPE	CHAR(2)	いいえ	いいえ	Explain 対象の照会のタイプの記述ラベル。 可能な値は以下のとおりです。 **S** 選択 **D** 削除 **DC** カーソルの現在位置の削除 **I** 挿入 **U** 更新 **UC** カーソルの現在位置の更新
UPDATABLE	CHAR(1)	いいえ	いいえ	このステートメントが更新可能であると見なされるかどうかを示します。これは特に、潜在的に更新可能であると見なされる可能性のある SELECT ステートメントに関係しています。 可能な値は以下のとおりです。 ' ' 該当しない (ブランク) **N** いいえ **Y** はい
DELETABLE	CHAR(1)	いいえ	いいえ	このステートメントが削除可能であると見なされるかどうかを示します。これは特に、潜在的に削除可能であると見なされる可能性のある SELECT ステートメントに関係しています。 可能な値は以下のとおりです。 ' ' 該当しない (ブランク) **N** いいえ **Y** はい
TOTAL_COST	DOUBLE	いいえ	いいえ	このステートメントについて選択されたアクセス・プランの実行のための合計コストの見積もり (timeron 単位)。 EXPLAIN_LEVEL が *O* (オリジナル・テキスト) の場合は、この時点で選択されているアクセス・プランがないため、ゼロに設定されます。
STATEMENT_TEXT	CLOB(2M)	いいえ	いいえ	Explain 対象の SQL ステートメントのテキスト、またはその一部。 Explain 機能のプラン選択レベルで表示されるテキストは、内部表記から再構成されたものであり、本質的に SQL ステートメントに類似したものです。再構成されたステートメントが正しい SQL 構文に準拠しているという保証はありません。

表 65. EXPLAIN_STATEMENT 表 (続き): PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
SNAPSHOT	BLOB(10M)	はい	いいえ	示されている Explain_Level での、この SQL ステートメントの内部表記のスナップショット。 この列は、DB2 Visual Explain で使用することを意図したものです。 EXPLAIN_LEVEL が *0* (元のステートメント) の場合は、ステートメントのこの特定バージョンがキャプチャーされた時点でアクセス・プランが選択されていないため、この列は NULL 値に設定されます。
QUERY_DEGREE	INTEGER	いいえ	いいえ	Explain の呼び出し時のパーティション内並列処理の多重度。元のステートメントの場合、ここには指定された多重度のパーティション内並列処理が入ります。 PLAN SELECTION の場合、ここには使用のプランに応じて生成されたパーティション内並列処理の多重度が入ります。
REOPT	CHAR(1)	いいえ	いいえ	ホスト変数、パラメーター・マーカー、または特殊レジスターの実際の値を使用して、SQL ステートメントを最適化し直すかどうかを指定します。 可能な値は以下のとおりです。 **N** デフォルトの変数推定値を使用してアクセス・パスを作成する。 **Y** ホスト変数、パラメーター・マーカー、または特殊レジスターの使用可能な値を使用して、アクセス・パスを最適化し直す。

EXPLAIN_STREAM 表

EXPLAIN_STREAM 表は、個々のオペレーターとデータ・オブジェクトの間の、入出力データ・ストリームを表します。データ・オブジェクト自体は、EXPLAIN_OBJECT 表に示されています。データ・ストリームに関連するオペレーターは、EXPLAIN_OPERATOR 表にあります。

表 66. EXPLAIN_STREAM 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	×	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	×	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	×	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	×	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	×	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	×	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	×	FK	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	×	FK	この Explain 情報に関連するパッケージのセクション番号。
STREAM_ID	INTEGER	×	×	このデータ・ストリームに対する特定のオペレーターのユニーク ID。
SOURCE_TYPE	CHAR(1)	No	No	このデータ・ストリームのソースを示します。 **O** オペレーター **D** データ・オブジェクト
SOURCE_ID	SMALLINT	No	No	このデータ・ストリームのソースである照会内の、オペレーターに対するユニーク ID。 SOURCE_TYPE が「D」の場合は -1 に設定されます。
TARGET_TYPE	CHAR(1)	No	No	このデータ・ストリームのターゲットを示します。 **O** オペレーター **D** データ・オブジェクト
TARGET_ID	SMALLINT	No	No	このデータ・ストリームのターゲットである照会内の、オペレーターに対するユニーク ID。 TARGET_TYPE が「D」の場合は -1 に設定されます。
OBJECT_SCHEMA	VARCHAR(128)	Yes	No	影響を受けるデータ・オブジェクトが属するスキーマ。 SOURCE_TYPE および TARGET_TYPE が共に「O」である場合、NULL に設定します。
OBJECT_NAME	VARCHAR(128)	Yes	No	データ・ストリームのサブジェクトであるオブジェクトの名前。 SOURCE_TYPE および TARGET_TYPE が共に「O」である場合、NULL に設定します。
STREAM_COUNT	DOUBLE	×	×	データ・ストリームのカーディナリティー推定値。
COLUMN_COUNT	SMALLINT	No	No	データ・ストリーム内の列数。
PREDICATE_ID	INTEGER	No	No	ストリームが述部に対する副照会の一部である場合、述部 ID が反映される。それ以外は列は -1 に設定される。

表 66. EXPLAIN_STREAM 表 (続き)： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
COLUMN_NAMES	CLOB(2M)	Yes	No	この列には、このストリームに関連した列の名前や配列情報が含まれています。 名前は以下の形式に従います。 `NAME1(A)+NAME2(D)+NAME3+NAME4` ここで、*(A)* は昇順の列、 *(D)* は降順の列を示し、配列情報がないものは、列が配列されていないか、配列が関係ないかのいずれかを示します。
PMID	SMALLINT	No	No	パーティション・マップの ID。
SINGLE_NODE	CHAR(5)	Yes	No	このデータ・ストリームが単一または複数のパーティションにあるかどうかを示します。 **MULT** 複数のパーティションにある **COOR** コーディネーター・ノードにある **HASH** ハッシュを使用して指定される **RID** 行 ID を使用して指定される **FUNC** 関数を使用して指定される (HASHEDVALUE() または DBPARTITIONNUM()) **CORR** 相関値を使用して指定される **Numberic** 事前に決められた単一ノードに指定される
PARTITION_COLUMNS	CLOB(2M)	Yes	No	このデータ・ストリームがパーティション分割される列のリスト。

ADVISE_INDEX 表

ADVISE_INDEX 表は、推奨索引を示しています。

表 67. ADVISE_INDEX 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	×	×	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	×	×	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	×	×	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	×	×	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	×	×	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	×	×	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	×	×	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	×	×	この Explain 情報に関連するパッケージのセクション番号。
QUERYNO	INTEGER	No	No	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントでは STMTNO の値で、動的 SQL ステートメントでは 1 です。
QUERYTAG	CHAR(20)	No	×	Explain 対象の各 SQL ステートメントの ID タグ。 CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLP」です。 CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLI」です。それ以外の場合、使用されるデフォルト値はブランクです。
NAME	VARCHAR(128)	×	×	索引の名前。
CREATOR	VARCHAR(128)	×	×	索引名の修飾子。
TBNAME	VARCHAR(128)	×	×	索引が定義されている表またはニックネームの名前。
TBCREATOR	VARCHAR(128)	×	×	表名の修飾子。
COLNAMES	CLOB(2M)	×	×	列名のリスト。
UNIQUERULE	CHAR(1)	No	No	ユニーク値に関する規則。 D = 重複可 P = 1 次索引 U = ユニークな項目のみ可
COLCOUNT	SMALLINT	×	×	キー内の列数と組み込み列 (もしあれば) の数の合計。
IID	SMALLINT	×	×	索引の内部 ID。
NLEAF	INTEGER	×	×	リーフ・ページの数。統計が収集されていない場合は -1。
NLEVELS	SMALLINT	×	×	索引レベルの数。統計が収集されていない場合は -1。
FIRSTKEYCARD	BIGINT	×	×	第 1 キーの値の数。統計が収集されていない場合は -1。
FULLKEYCARD	BIGINT	×	×	個別の全キー値の数。統計が収集されていない場合は -1。

表 67. ADVISE_INDEX 表 (続き)： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
CLUSTERRATIO	SMALLINT	×	×	索引によるデータ・クラスタリングの程度。統計が収集されていない場合、または詳細な索引統計が収集されている場合は -1 (それらの場合は CLUSTERFACTOR の方が使用されます)。
CLUSTERFACTOR	DOUBLE	×	×	より高い計算精度のクラスタリング。詳細索引統計を収集していない場合、あるいはニックネームに索引が定義されていない場合は -1。
USERDEFINED	SMALLINT	×	×	ユーザーによる定義。
SYSTEM_REQUIRED	SMALLINT	No	No	次の条件のいずれかを満たす場合は 1。 – この索引が主キー制約またはユニーク・キー制約で必要です。またはこの索引がマルチディメンション・クラスタリング (MDC) 表のディメンション・ブロック索引または複合ブロック索引です。 – これはタイプ表のオブジェクト ID (OID) 列に対する索引です。 次の条件の両方を満たす場合は 2。 – この索引は主キー制約またはユニーク・キー制約に必要です。または、この索引は MDC 表に対するディメンション・ブロック索引です。 – これはタイプ表のオブジェクト ID (OID) 列に対する索引です。 それ以外の場合は 0。
CREATE_TIME	TIMESTAMP	No	×	索引の作成された時刻。
STATS_TIME	TIMESTAMP	Yes	No	この索引について記録されている統計値が最後に変更された時刻。統計が利用できない場合は、NULL。
PAGE_FETCH_PAIRS	VARCHAR(254)	No	No	文字形式で表された、整数の対のリスト。それぞれの対は、仮のバッファー内のページ数と、その仮のバッファーを使用した表のスキャンに必要なページ取り出しの回数を表しています。 (データが利用できない場合は、長さ 0 のストリング。)
REMARKS	VARCHAR(254)	Yes	×	ユーザー提供のコメントまたは NULL 値。
DEFINER	VARCHAR(128)	×	×	索引を作成したユーザー。
CONVERTED	CHAR(1)	×	×	将来の使用のために予約済み。
SEQUENTIAL_PAGES	INTEGER	No	No	索引キーの順序でディスクに存在し、それらの間に大きなギャップがないか、わずかなギャップしかないリーフ・ページの数。 (統計が入手できない場合は -1。)
DENSITY	INTEGER	×	×	索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表されます (0 ～ 100 の整数。統計が入手できない場合は -1)。
FIRST2KEYCARD	BIGINT	×	×	索引の最初の 2 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。
FIRST3KEYCARD	BIGINT	×	×	索引の最初の 3 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。
FIRST4KEYCARD	BIGINT	×	×	索引の最初の 4 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。

表 67. ADVISE_INDEX 表 (続き): PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
PCTFREE	SMALLINT	No	No	索引を最初に作成する際に予約する索引リーフ・ページのパーセンテージ。このスペースは、索引の作成後に行う挿入用に使用可能です。
UNIQUE_COLCOUNT	SMALLINT	No	×	ユニーク・キーに必要な列の数。常に <=COLCOUNT。列を含む場合のみ、< COLCOUNT。索引にユニーク・キーがない場合は -1 (重複可能)。
MINPCTUSED	SMALLINT	×	×	ゼロでない場合は、オンライン索引デフラグが使用可能になり、その値は、ページをマージをする前に使用される最小スペースのしきい値です。
REVERSE_SCANS	CHAR(1)	No	No	Y = 索引は逆スキャンをサポートする N = 索引は逆スキャンをサポートしない
USE_INDEX	CHAR(1)	Yes	No	Y = 推奨または評価された索引 N = 推奨されない索引
CREATION_TEXT	CLOB(2M)	No	×	索引の作成に使用された SQL ステートメント。
PACKED_DESC	BLOB(1M)	Yes	No	表の内部記述。

ADVISE_INSTANCE 表

ADVISE_INSTANCE 表には、**db2advis** の実行に関する情報が入ります。これには開始時刻に関する情報も含まれます。 **db2advis** の実行ごとに 1 行が入ります。他の ADVISE 表には、ADVISE_INSTANCE 表の START_TIME 列 (設計アドバイザーの同一の実行に関するもの) とリンクした外部キー (RUN_ID) があります。

表 68. ADVISE_INSTANCE 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
START_TIME	TIMESTAMP	No	PK	**db2advis** の実行が開始した時刻。
END_TIME	TIMESTAMP	No	No	**db2advis** の実行が終了した時刻。
MODE	VARCHAR(4)	No	No	設計アドバイザーの -m オプションに指定された値。たとえば、「MC」は MQT と MDC を指定します。
WKLD_COMPRESSION	CHAR(4)	No	No	設計アドバイザーを実行したときのワークロードの圧縮。
STATUS	CHAR(9)	No	No	設計アドバイザーの実行の状況。状況は「STARTED」、「COMPLETED」(正常な場合)、またはエラー番号になります。エラー番号の接頭部は、内部エラーの場合は「EI」、外部エラーの場合は「EX」で、後者の場合、エラー番号は SQLCODE を表します。

ADVISE_MQT 表

ADVISE_MQT 表には、設計アドバイザーが推奨するマテリアライズ照会表 (MQT) に関する情報が入ります。

表 69. ADVISE_MQT 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	No	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	No	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	No	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	No	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	No	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	No	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	No	No	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
NAME	VARCHAR(128)	No	No	MQT 名。
CREATOR	VARCHAR(128)	No	No	MQT 作成者名。
IID	SMALLINT	No	No	内部 ID。
CREATE_TIME	TIMESTAMP	No	No	MQT が作成された時刻。
STATS_TIME	TIMESTAMP	Yes	No	統計が取られた時刻。
NUMROWS	DOUBLE	No	No	MQT の推定行数。
NUMCOLS	SMALLINT	No	No	MQT に定義されている列数。
ROWSIZE	DOUBLE	No	No	MQT の行の平均の長さ (バイト単位)。
BENEFIT	FLOAT	No	No	将来の使用のために予約済み。
USE_MQT	CHAR(1)	Yes	No	MQT の使用が推奨される場合は「Y」に設定。
MQT_SOURCE	CHAR(1)	Yes	No	MQT 候補がどこに生成されたかを示します。 MQT 候補が refresh-immediate MQT の場合は「I」、フル refresh-deferred MQT として作成することしかできない場合は「D」に設定されます。
QUERY_TEXT	CLOB(2M)	No	No	MQT を定義する照会。
CREATION_TEXT	CLOB(2M)	No	No	MQT の CREATE TABLE DDL。
SAMPLE_TEXT	CLOB(2M)	No	No	MQT の詳細な統計を得るために使用される、サンプリング照会。設計アドバイザーが詳細な統計を必要とする場合にのみ使用されます。結果として得られた、サンプリング済みの統計がこの表に示されます。 NULL の場合、この MQT にはサンプリング照会が作成されませんでした。
COLSTATS	CLOB(2M)	No	No	MQT の列統計 (NULL でない場合)。これらの統計は XML フォーマットであり、列名と列のカーディナリティー、およびオプションで HIGH2KEY 値と LOW2KEY 値が含まれます。
EXTRA_INFO	BLOB(2M)	No	No	各種の出力のために予約済み。
TBSPACE	VARCHAR(128)	No	No	MQT のために推奨される表スペース。

表 69. ADVISE_MQT 表 (続き): PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
RUN_ID	TIMESTAMP	Yes	FK	ADVISE_INSTANCE 表の 1 つの行の START_TIME に対応する値。設計アドバイザーの同一の実行にリンクします。
REFRESH_TYPE	CHAR(1)	No	No	即時の場合は「I」、据え置きの場合は「D」に設定。
EXISTS	CHAR(1)	No	No	MQT がデータベース・カタログに存在する場合は「Y」に設定。

ADVISE_PARTITION 表

ADVISE_PARTITION 表には、設計アドバイザーが推奨するデータベース・パーティションに関する情報が入ります。この表にデータを追加できるのはパーティション・データベース環境の場合に限られます。

表 70. ADVISE_PARTITION 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	No	No	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	No	No	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	No	No	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	No	No	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	No	No	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	No	No	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	No	No	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
SECTNO	INTEGER	No	No	パッケージ内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
QUERYNO	INTEGER	No	No	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントでは STMTNO の値で、動的 SQL ステートメントでは 1 です。
QUERYTAG	CHAR(20)	No	No	Explain 対象の各 SQL ステートメントの ID タグ。 CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLP」です。 CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLI」です。それ以外の場合、使用されるデフォルト値はブランクです。
TBNAME	VARCHAR(128)	Yes	No	表名を指定します。
TBCREATOR	VARCHAR(128)	Yes	No	表の作成者名を指定します。
PMID	SMALLINT	Yes	No	パーティション・マップ ID を指定します。
TBSPACE	VARCHAR(128)	Yes	No	表が存在する表スペースを指定します。
COLNAMES	CLOB(2M)	Yes	No	パーティション列名をコンマ区切りで指定します。
COLCOUNT	SMALLINT	Yes	No	パーティション列の数を指定します。
REPLICATE	CHAR(1)	Yes	No	パーティションが複製されているかどうかを指定します。
COST	DOUBLE	Yes	No	パーティションの使用のコストを指定します。
USEIT	CHAR(1)	Yes	No	パーティションが EVALUATE PARTITION モードで使用中かどうかを指定します。 USEIT が「Y」または「y」に設定されている場合、パーティションは使用中です。
RUN_ID	TIMESTAMP	Yes	FK	ADVISE_INSTANCE 表の 1 つの行の START_TIME に対応する値。設計アドバイザーの同一の実行にリンクします。

ADVISE_TABLE 表

ADVISE_TABLE 表には、マテリアライズ照会表 (MQT)、マルチディメンション・クラスタリング表 (MDC)、およびパーティションに関する設計アドバイザーの最終的な勧告を使った、テーブル作成用のデータ定義言語 (DDL) が格納されます。

表 71. ADVISE_TABLE 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
RUN_ID	TIMESTAMP	Yes	FK	ADVISE_INSTANCE 表の 1 つの行の START_TIME に対応する値。設計アドバイザーの同一の実行にリンクします。
TABLE_NAME	VARCHAR(128)	No	No	表の名前。
TABLE_SCHEMA	VARCHAR(128)	No	No	表作成者の名前。
TABLESPACE	VARCHAR(128)	No	No	表を作成する表スペース。
SELECTION_FLAG	VARCHAR(4)	No	No	推奨のタイプを示します。有効な値は、「M」(MQT の場合)、「P」(パーティション化の場合)、および「C」(MDC の場合) です。このフィールドには上記の値のあらゆるサブセットが入れます。たとえば、「MC」は表が MQT および MDC 表として推奨されることを示します。
TABLE_EXISTS	CHAR(1)	No	No	表がデータベース・カタログに存在する場合は「Y」に設定。
USE_TABLE	CHAR(1)	No	No	表に設計アドバイザーからの勧告がある場合は「Y」に設定。
GEN_COLUMNS	CLOB(2M)	No	No	表作成 DDL によって生成された列を必要とする MDC 推奨がこの行に含まれている場合、生成列のストリング。
ORGANIZE_BY	CLOB(2M)	No	No	MDC 推奨の場合、表作成 DDL の ORGANIZE BY 文節。
CREATION_TEXT	CLOB(2M)	No	No	表作成 DDL。
ALTER_COMMAND	CLOB(2M)	No	No	表の ALTER TABLE ステートメント。

ADVISE_WORKLOAD 表

ADVISE_WORKLOAD 表は、ワークロードを構成するステートメントを示しています。

表 72. ADVISE_WORKLOAD 表： PK は、その列が主キーの一部であることを示します。 FK は、その列が外部キーの一部であることを示します。

列名	データ・タイプ	NULL 可能?	キー?	説明
WORKLOAD_NAME	CHAR(128)	×	×	このステートメントが属している SQL ステートメント (ワークロード) の集合の名前。
STATEMENT_NO	INTEGER	×	×	ワークロード内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
STATEMENT_TEXT	CLOB(1M)	×	×	SQL ステートメントの内容。
STATEMENT_TAG	VARCHAR(256)	×	×	Explain 対象の各 SQL ステートメントの ID タグ。
FREQUENCY	INTEGER	×	×	ワークロード内でこのステートメントが使用される回数。
IMPORTANCE	DOUBLE	×	×	ステートメントの重要性。
WEIGHT	DOUBLE	No	×	ステートメントの優先度。
COST_BEFORE	DOUBLE	○	×	推奨された索引が作成されない場合の照会にかかるコスト (timerons 単位)。
COST_AFTER	DOUBLE	Yes	No	推奨された索引が作成される場合の照会にかかるコスト (timerons 単位)。
COMPILABLE	CHAR(17)	Yes	No	ステートメントを準備しようとしている間に生じた照会コンパイル・エラーを示します。この列が NULL であるか、または SQLCA で始まっていない場合、 SQL 照会は db2advis でコンパイルできます。コンパイル・エラーが db2advis または設計アドバイザーによって検出される場合、 COMPILABLE 列値は、8 文字の SQLCA.sqlcaid フィールド、コロン (:)、および 8 文字の SQLCA.sqlstate フィールドで構成されますが、この値は SQL ステートメントの戻りコードです。

付録 C. SQL Explain ツール

db2expln および dynexpln ツールを使用すると、特定の SQL ステートメント用に選択されたアクセス・プランについて理解することができます。また、コントロール・センターに統合された Explain 機能である Visual Explain と組み合わせて使用し、特定の SQL ステートメント用に選択されたアクセス・プランについて理解することもできます。 Explain 機能を用いることにより、動的および静的 SQL ステートメントの両方を Explain することができます。 Explain ツールとの相違点の 1 つは、 Visual Explain では、Explain 情報がグラフィック形式で表示されることです。それ以外は、2 つの方式で提示される明細のレベルは同じです。

db2expln および dynexpln の出力を十分に活用するためには、以下の事項について理解しておく必要があります。

- サポートされる各種 SQL ステートメントと、それらのステートメントに関連した用語 (SELECT ステートメント内の述部など)
- パッケージの目的 (アクセス・プラン)
- システム・カタログ表の目的および内容
- 一般的なアプリケーション・チューニングの概念

このセクションのトピックで、 db2expln および dynexpln について説明します。

SQL Explain ツール

db2expln ツールは、 SQL ステートメント用に選択されたアクセス・プランを記述します。これを使用して、 Explain データがキャプチャーされなかったときに選択されたアクセス・プランに関する早見 Explain を入手することができます。静的 SQL では、 db2expln を使用してシステム・カタログ表に保管されたパッケージを調べます。動的 SQL では、 db2expln を使用して SQL キャッシュ内のセクションを調べます。

dynexpln ツールを使用して、動的ステートメント用に選択されたアクセス・プランを記述することもできます。このツールは、ステートメント用に静的パッケージを作成し、 db2expln ツールを使用してステートメントを記述します。しかし、動的 SQL は db2expln を使用して調べることができるので、このユーティリティーが残っているのは以前のバージョンとの互換性のためだけです。

Explain ツール (db2expln および dynexpln) は、ご使用のインスタンスの sqllib ディレクトリーの bin サブディレクトリーの中にあります。 db2expln および dynexpln が現行ディレクトリーにない場合は、該当の PATH 環境変数に示されているディレクトリーにあるはずです。

db2expln プログラムは、データベースが最初にアクセスされるときに接続され、db2expln.bnd、db2exsrv.bnd、および db2exdyn.bnd ファイルを用いてデータベースにバインドされます。

db2expln を実行するには、システム・カタログ・ビューに対する SELECT 特権と、 db2expln、db2exsrv、および db2exdyn のパッケージの EXECUTE 特権を持っている必要があります。 dynexpln を実行するには、データベースへの BINDADD 権限を所有し、データベースに接続する際に使用するスキーマが存在しているか、もしくはデータベースへの IMPLICIT_SCHEMA 権限を所有している必要があります。 db2expln または dynexpln を使用して動的 SQL を Explain するには、 Explain されている SQL ステートメントに必要な特権も必要です。 (SYSADM 権限または DBADM 権限を持っている場合は、自動的にこれらの権限レベルをすべて持つことになります。)

関連概念:

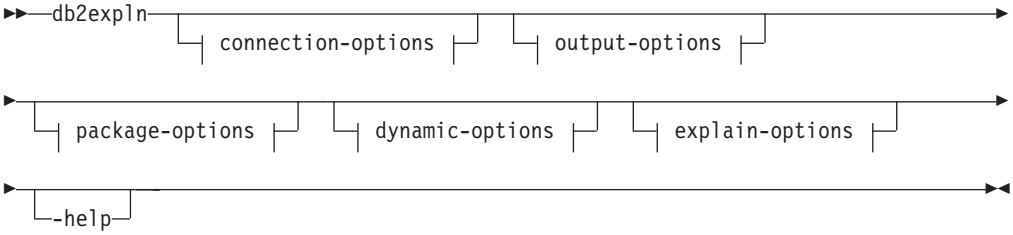
- 213 ページの『SQL Explain 機能』
- 223 ページの『Explain 情報のキャプチャーのガイドライン』
- 226 ページの『Explain 情報の分析のガイドライン』

db2expln

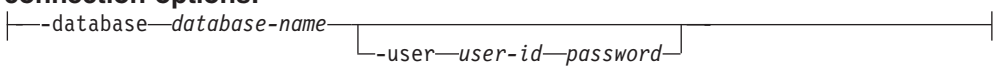
次のセクションで、 *db2expln* の構文とパラメーターについて説明し、使用上の注意を示します。

db2expln - SQL Explain

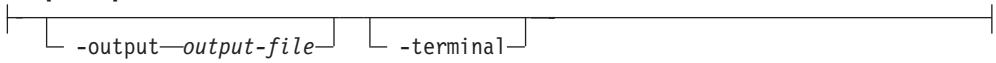
コマンド構文:



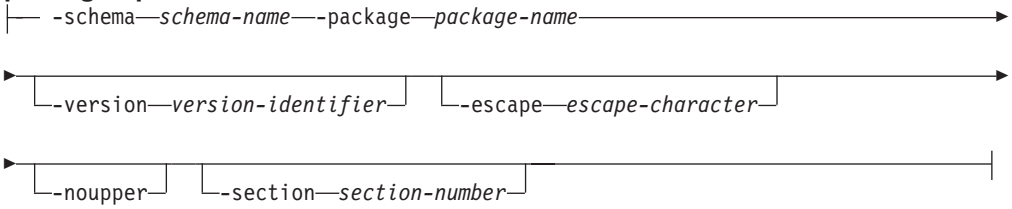
connection-options:



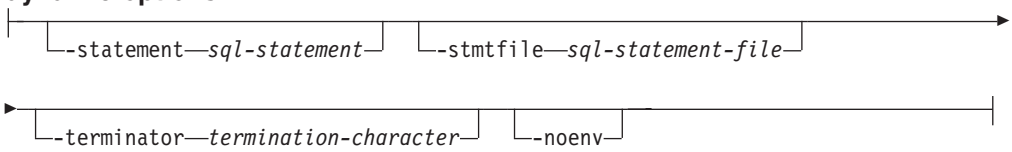
output-options:



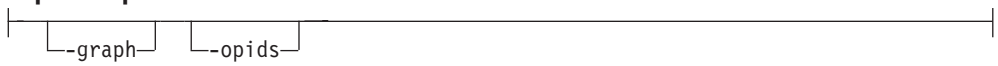
package-options:



dynamic-options:



explain-options:



コマンド・パラメーター:
オプションは任意の順序で指定できます。

connection-options:

これらのオプションは、接続するデータベースおよび接続に必要なオプションを指定します。接続オプションは、**-help** オプションが指定されている場合を除いて必要となります。

-database *database-name*
Explain されるパッケージが入っているデータベースの名前。

以前のバージョンとの互換性のために、 **-d** を **-database** の代わりに使用できます。

-user *user-id password*
データベース接続を確立するときに使用する許可 ID およびパスワード。 *user-id* および *password* はどちらも、 DB2® 命名規則に従った有効なもので、さらにデータベースが認識できるものでなければなりません。

以前のバージョンとの互換性のために、 **-u** を **-user** の代わりに使用できます。

output-options:

これらのオプションは、db2expln の出力先の誘導する場所を指定します。 **-help** オプションが指定されている場合を除いて、少なくとも 1 つの出力オプションを指定する必要があります。両方のオプションを指定した場合、出力はファイルおよび端末に送られます。

-output *output-file*
db2expln の出力は指定したファイルに書き込まれます。

以前のバージョンとの互換性のために、 **-o** を **-output** の代わりに使用できます。

-terminal
db2expln 出力は端末に送信されます。

以前のバージョンとの互換性のために、 **-t** を **-terminal** の代わりに使用できます。

package-options:

これらのオプションは Explain される 1 つ以上のパッケージおよびセクションを指定します。パッケージおよびセクション内の静的 SQL だけが Explain されます。

注: LIKE 述部のように、パーセント記号 (%) および下線 (_) のパターン・マッチング文字を使用して、*schema-name*、*package-name*、および *version-identifier* を指定できます。

-schema *schema-name*

Explain されるパッケージ (1 つまたは複数) のスキーマ。

以前のバージョンとの互換性のために、 **-s** を **-schema** の代わりに使用できます。

-package *package-name*

Explain されるパッケージ (1 つまたは複数) の名前。

以前のバージョンとの互換性のために、 **-p** を **-package** の代わりに使用できます。

-version *version-identifier*

Explain されるパッケージ (1 つまたは複数) のバージョン ID。デフォルトのバージョンは空ストリングです。

-escape *escape-character*

schema-name、*package-name*、および *version-identifier* でパターン・マッチングのエスケープ文字として使用される *escape-character* 文字。

たとえば、パッケージ TESTID.CALC% を Explain するための db2expln コマンドは、次のとおりです。

```
db2expln -schema TESTID -package CALC% ....
```

しかし、このコマンドは、 CALC で始まるものであればその他のプランも Explain することになります。 TESTID.CALC% パッケージだけを Explain したい場合は、エスケープ文字を使用しなければなりません。感嘆符 (!) をエスケープ文字に指定した場合、コマンドを以下のように変更できます。 `db2expln -schema TESTID -escape ! -package CALC!% ...` . その結果、! 文字はエスケープ文字として使用されるので、 !% は「すべてとマッチする」パターンとしてではなく % 文字として解釈されます。デフォルトのエスケープ文字はありません。

以前のバージョンとの互換性のために、 **-e** を **-escape** の代わりに使用できます。

注: 問題を避けるため、オペレーティング・システムのエスケープ文字を db2expln エスケープ文字として指定しないでください。

-noupper

schema-name、 *package-name*、および *version-identifier* を、マッチするパッケージの検索前に大文字に変換しないことを指定します。

デフォルトでは、これらの変数はパッケージの検索前に大文字に変換されます。このオプションは、これらの値を入力されたとおりに使用することを指定します。

後方の互換性のために、 **-l** (小文字の L であり数字の 1 ではない) を **-noupper** の代わりに使用できます。

-section *section-number*
パッケージ (1 つまたは複数) 内にある Explain するセクションの番号。

各パッケージ内のすべてのセクションを Explain するには、数字のゼロ (0) を使用します。これがデフォルトの動作です。このオプションを指定しない場合、または *schema-name*、 *package-name*、または *version-identifier* にパターン・マッチング文字が含まれる場合は、すべてのセクションが表示されます。

セクション番号を検索するには、システム・カタログ・ビュー SYSCAT.STATEMENTS を照会します。システム・カタログ・ビューについての説明は、 *SQL リファレンス*を参照してください。

以前のバージョンとの互換性のために、 **-s** を **-section** の代わりに使用できます。

dynamic-options:

これらのオプションは Explain される 1 つ以上の動的 SQL ステートメントを指定します。

-statement *sql-statement*
動的に準備して Explain する SQL ステートメント。複数のステートメントを Explain するには、**-stmtfile** オプションを使用して Explain する SQL ステートメントを含むファイルを供給するか、または **-terminator** オプションを使用して **-statement** オプション内のステートメントを分離するために使用できる終止符を定義します。

`dynexpln` との互換性のために、 **-q** を **-statement** の代わりに使用できます。

-stmtfile *sql-statement-file*
動的に準備して Explain する 1 つ以上の SQL ステートメントを含むファイル。デフォルトでは、ファイルの各行は別個の SQL ステートメントであると見なされます。ステートメントが複数の行に及ぶ場合には、**-terminator** オプションを指定して SQL ステートメントの終了を印付ける文字を指定します。

`dynexpln` との互換性のために、 **-f** を **-stmtfile** の代わりに使用できます。

-terminator *termination-character*
動的 SQL ステートメントの終了を示す文字。デフォルトでは、**-statement** オプションは単一の SQL ステートメントを提供して、 **-stmtfile** 内のファイルの各行は別個の SQL ステートメントとして扱われます。指定した終了文字は、**-statement** で複数の SQL ステートメントを供給したり、**-stmtfile** ファイルでステートメントが複数の行に及ぶようにするために使用できます。

`dynexpln` との互換性のために、 **-z** を **-terminator** の代わりに使用できます。

-noenv

コンパイル環境を変更する動的ステートメントが Explain された後に実行されないように指定します。

デフォルトでは、db2expln は以下のステートメントを Explain した後に実行します。

```
SET CURRENT DEFAULT TRANSFORM GROUP
SET CURRENT DEGREE
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
SET CURRENT QUERY OPTIMIZATION
SET CURRENT REFRESH AGE
SET PATH
SET SCHEMA
```

これらのステートメントによって、db2expln が生成する後続の動的 SQL ステートメントのために選択されたプランを変更することができます。

-noenv を指定すると、これらのステートメントは Explain されますが、実行されません。

動的 SQL を Explain するには、 **-statement** または **-stmtfile** を指定することが必要です。両方のオプションを db2expln の 1 回の呼び出しで指定することもできます。

explain-options:

これらのオプションは、Explain されたプランで追加される情報を判別します。

-graph

オプティマイザーのプランをグラフ表示します。個々のセクションが検査され、 Visual Explain で生成されたオリジナルのオプティマイザーのプランのグラフが構成されます。生成されるグラフは、Visual Explain のグラフと正確には一致しない場合もあることに注意してください。

以前のバージョンとの互換性のために、 **-g** を **-graph** の代わりに指定できます。

-opids Explain されるプランの中のオペレーター ID 番号を表示します。

オペレーター ID 番号を使用すると、 db2expln からの出力を、Explain 機能からの出力と突き合わせることができます。すべてのオペレーターに ID 番号があるのではなく、 Explain 機能出力に存在するいくつかの ID 番号は db2expln 出力には存在しないことに注意してください。

以前のバージョンとの互換性のために、 **-i** を **-opids** の代わりに指定できます。

-help db2expln のヘルプ・テキストを表示します。このオプションを指定した場合、どのパッケージも Explain されません。

ほとんどのコマンド行は *db2exsrv* ストアード・プロシージャーで処理されます。使用可能なすべてのオプションについてのヘルプを表示するには、 **connection-options** を **-help** と共に指定する必要があります。たとえば、以下を使用します。

```
db2expln -help -database SAMPLE
```

以前のバージョンとの互換性のために、**-h** または **-?** を指定できます。

使用上の注意:

-help オプションを指定していなければ、 package-options または dynamic-options のいずれかを指定しなければなりません。パッケージおよび動的 SQL の両方を、単一の db2expln 呼び出しで Explain できます。

上記のオプション・フラグの中には、ご使用のオペレーティング・システムにとって特別な意味をもつものもあり、その結果、db2expln コマンド行で正しく解釈されないこともあります。しかし、これらの文字の前にオペレーティング・システムのエスケープ文字を付けて入力することも可能です。詳細については、ご使用のオペレーティング・システムのマニュアルを参照してください。うっかりとオペレーティング・システムのエスケープ文字を db2expln エスケープ文字として指定しないようにしてください。

db2expln によって作成されるヘルプおよび初期状況メッセージは、標準出力に書き出されます。 Explain ツールにより作成されるすべてのプロンプトおよびその他の状況メッセージは、標準エラーに書き出されます。 Explain テキストは、選択した出力オプションに基づいて、標準出力またはファイルに書き出されます。

例:

db2expln の 1 回の呼び出しで複数のプランを Explain するには、 **-package**、**-schema**、および **-version** オプションを使用して LIKE パターンでパッケージと作成者に関するストリング定数を指定します。つまり、下線 (_) を使用して単一文字を表し、パーセント記号 (%) を用いてゼロ個以上の文字があることを表すことができます。

SAMPLE という名前のデータベース内のすべてのパッケージに関するすべてのセクションの Explain を、ファイル **my.exp** に書き込まれるようにするには、次のように入力します。

```
db2expln -database SAMPLE -schema % -package %  -output my.exp
```

別の例として、ユーザーが「statements.db2」と呼ばれる CLP スクリプト・ファイルを持ち、ファイル内のステートメントを Explain したいと想定します。ファイルには以下のステートメントが含まれます。

```
SET PATH=SYSIBM, SYSFUN, DEPT01, DEPT93@
SELECT EMPNO, TITLE(JOBID) FROM EMPLOYEE@
```

これらのステートメントを Explain するには、以下のコマンドを入力します。

```
db2expln -database DEPTDATA -stmtfile statements.db2 -terminator @ -terminal
```

関連概念:

- 643 ページの『SQL Explain ツール』
- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

db2expln の使用上の注意

db2expln で表示される共通のメッセージを以下に示します。

- データベース・パッケージ・パターン: バージョン "<version>" の "<creator>".<package> に一致するパッケージが見つかりません。 (No packages found for database package pattern: "<creator>".<package> with version "<version>")

  指定されたパターンに一致するパッケージがデータベース内に見つからなかった場合に、このメッセージが出力に表示されます。
- バインド・メッセージは db2expln.msg にあります。(Bind messages can be found in db2expln.msg.)

  db2expln.bnd のバインドがうまく行われなかった場合に、このメッセージが出力に表示されます。検出された問題に関するさらに詳しい情報は、現行ディレクトリー内のファイル db2expln.msg にあります。
- 潜在的な重複パッケージについてセクション番号が 0 にオーバーライドされました (全セクション)。(Section number overridden to 0 (all sections) for potential multiple packages.)

  db2expln が複数のパッケージを検出する可能性がある場合に、このメッセージが出力に表示されます。いずれかのパターン照合文字がパッケージまたは作成者入力引き数に使用されている場合に、この処置が取られます。
- <bind file> のバインド・メッセージが <message file> にあります。(Bind messages for <bind file> can be found in <message file>)

  所定のバインド・ファイルのバインドがうまく行われなかった場合に、このメッセージが表示されます。検出された問題に関するさらに詳しい情報は、データベース・サーバー上の所定のメッセージ・ファイルにあります。
- パッケージに適した静的セクションがありません。(No static sections qualify from package.)

  指定されたパッケージに動的 SQL ステートメントだけが含まれている (すなわち、静的セクションがないことを意味する) 場合に、このメッセージが出力に表示されます。
- パッケージ "<creator>"."<package>", "<version>" は有効ではありません。パッケージを再バインドしてから、db2expln を再実行してください。 (Package "<creator>"."<package>", "<version>", is not valid. Rebind the package and then rerun db2expln.)

  指定されたパッケージが現在は有効ではない場合に、このメッセージが出力に表示されます。指示どおりに、プランに対して BIND または REBIND コマンドを再発行し、データベース内に有効なパッケージを再作成して、さらに db2expln を再実行してください。

除外される SQL ステートメント: 以下のステートメントについては、Explain されません。

- BEGIN/END DECLARE SECTION
- BEGIN/END COMPOUND
- INCLUDE
- WHENEVER

- COMMIT および ROLLBACK
- CONNECT
- OPEN カーソル
- FETCH
- CLOSE カーソル
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- DESCRIBE
- 動的 DECLARE CURSOR
- SQL 制御ステートメント

コンパウンド SQL ステートメント内の個々のサブステートメントには、独自のセクションが存在していることがあり、そのセクションを **db2expln** で Explain することができます。

dynexpln

dynexpln ツールは、以前のバージョンとの互換性のために今でも使用可能です。しかし、db2expln の **dynamic-options** を使用して、 dynexpln のすべての機能を実行できます。

db2expln の dynamic-options を使用するとき、ステートメントは真の動的 SQL として準備され、生成されるプランは SQL キャッシュから Explain されます。explain の出力方法は、ステートメントを静的 SQL として準備する dynexpln よりも正確なアクセス・プランを提供します。これにより、パラメーター・マーカーなど動的 SQL だけで使用できる機能も使用可能になります。

関連概念:

- 643 ページの『SQL Explain ツール』
- 671 ページの『db2expln および dynexpln 出力の例』

Explain 出力情報

次のセクションで、 *db2expln* および *dynexpln* が提供できる情報の種類について説明します。

db2expln および dynexpln 出力の説明

出力では、各パッケージの Explain 情報が次の 2 つの部分に分かれて表示されます。

- バインド日付や関係する BIND オプションなどのパッケージ情報。
- セクション番号 (前) および Explain される SQL ステートメント (後) などの、セクション情報。セクション情報の下には、表示された SQL ステートメント用に選択されたアクセス・プランの Explain 出力が表示されます。

アクセス・プラン (つまりセクション) のステップは、データベース・マネージャーが実行する順序で示されます。それぞれの主なステップは、左寄せの見出しで示され、そのステップに関する情報はその下に字下げして示されます。字下げの棒線は、アクセス・プランの Explain 出力の左マージンに表示されます。これらの棒線は、操作の有効範囲を示すものともなります。ずっと右側にある、同じ操作の中でもより下位の字下げレベルの操作は、すぐ上位にある字下げレベルに戻る前に処理されます。

出力に表示されている、選択されたアクセス・プランが元の SQL ステートメントの増補されたものに基づいていることを思い出してください。たとえば、元のステートメントがトリガーや制約を活動化するものである場合があります。さらに、SQL コンパイラーの照会再書き込みコンポーネントは、 SQL ステートメントを同等ではあるもののより効率的な形式に書き直すことがあります。オプティマイザーがこのステートメントを満足させるのに最も効率的なプランを決める際に、オプティマイザーはこれらのすべての要素を含む情報を使用します。したがって、Explain 出力に示されるアクセス・プランが、元の SQL ステートメントについて予期していたアクセス・プランとは本質的に異なってしまうという場合もあります。 Explain 表、SET CURRENT EXPLAIN モード、および Visual Explain を含む SQL Explain 機能は、最適化に使用される実際の SQL ステートメントを、照会の内部表記を逆変換して作成し、 SQL のようなステートメントの形式で示します。

db2expln または dynexpln からの出力を、 Explain 機能の出力と比較するときに、オペレーター ID オプション (**-opids**) は非常に便利です。 db2expln または dynexpln が、 Explain 機能から新しいオペレーターの処理を開始するたびに、オペレーター ID 番号が、Explain されるプランの左側に印刷されます。オペレーター ID は、異なる表示のアクセス・プランの中で、ステップを突き合わせるために使用することができます。 Explain 機能の出力の中のオペレーターと、 db2expln および dynexpln によって示される操作の間が、つねに 1 対 1 で対応しているわけではないことに注意してください。

関連概念:

- 651 ページの『dynexpln』
- 653 ページの『表アクセス情報』
- 658 ページの『一時表の情報』
- 660 ページの『結合の情報』
- 662 ページの『データ・ストリーム情報』
- 663 ページの『挿入、更新、および削除の情報』
- 664 ページの『ブロックと行 ID の準備の情報』
- 665 ページの『集約の情報』
- 665 ページの『並列処理の情報』
- 668 ページの『フェデレーテッド照会の情報』
- 669 ページの『その他の情報』

関連資料:

- 644 ページの『db2expln - SQL Explain』

表アクセス情報

このステートメントは、アクセスする表の名前とタイプを指定します。使用できる形式には、次の 2 つがあります。

1. レギュラー表には、以下の 3 つのタイプがあります。
   - アクセスした表名:

     ```
     Access Table Name = schema.name  ID = ts,n
     ```

     説明:
     - *schema.name* は、アクセスする表の完全修飾名です。
     - *ID* は、 SYSCAT.TABLES カタログ内での表に対応する TABLESPACEID と TABLEID です。
   - アクセスした階層表名:

     ```
     Access Hierarchy Table Name = schema.name  ID = ts,n
     ```

     説明:
     - *schema.name* は、アクセスする表の完全修飾名です。
     - *ID* は、 SYSCAT.TABLES カタログ内での表に対応する TABLESPACEID と TABLEID です。
   - アクセスしたマテリアライズ照会表名:

     ```
     Access Materialized Query Table Name = schema.name  ID = ts,n
     ```

     説明:
     - *schema.name* は、アクセスする表の完全修飾名です。
     - *ID* は、 SYSCAT.TABLES カタログ内での表に対応する TABLESPACEID と TABLEID です。
2. 一時表には、以下の 2 つのタイプがあります。
   - アクセスした一時表 ID:

     ```
     Access Temp Table  ID = tn
     ```

     説明:
     - *ID* は、db2expln から割り当てられた対応の ID です。
   - アクセスした宣言済みグローバル一時表 ID:

     ```
     Access Global Temp Table  ID = ts,tn
     ```

     説明:
     - *ID* は、 SYSCAT.TABLES カタログ内での表に対応する TABLESPACEID (ts)、および db2expln から割り当てられた、対応する ID (tn) です。

表アクセス・ステートメントに続いて、アクセスを詳しく記述するためにさらにステートメントが提供されています。これらのステートメントは、表アクセス・ステートメントの下に字下げされます。可能なステートメントは、以下のとおりです。

- 列の数
- ブロック・アクセス
- 並列スキャン

- スキャンの方向
- 行アクセス方式
- ロックの意図
- 述部
- その他のステートメント

列の数

次のステートメントは、表の各行から使用する列の数を示します。

```
#Columns = n
```

ブロック・アクセス

以下のステートメントは、表に 1 つ以上のディメンション・ブロック索引が定義されていることを示しています。

```
Clustered by Dimension for Block Index Access
```

このテキストが表示されていない場合は、表は DIMENSION 文節を指定されないで作成されました。

並列スキャン

次のステートメントは、データベース・マネージャーがサブエージェントをいくつか使用して、表から並列に読み取りをすることを示します。

```
Parallel Scan
```

このテキストが示されない場合は、表からの読み取りは 1 つのエージェント (またはサブエージェント) によってのみ行われます。

スキャンの方向

次のステートメントは、データベース・マネージャーが行を逆順に読み取ることを示します。

```
Scan Direction = Reverse
```

このテキストが示されていない場合は、スキャン方向は順方向で、これがデフォルトです。

行アクセス方式

次のステートメントのうちの 1 つが表示されている場合、表内の修飾行がアクセスされる方法を示します。

- Relation Scan ステートメントは、修飾行を検索するために表を順次スキャンすることを示します。
  - 次のステートメントは、データのプリフェッチが行われないことを示しています。

    ```
    Relation Scan
    |  Prefetch: None
    ```

  - 次のステートメントは、プリフェッチされるページ数をオプティマイザーが事前に判断したことを示します。

```
Relation Scan
|  Prefetch: n Pages
```

– 次のステートメントは、データをプリフェッチすることを示します。

```
Relation Scan
|  Prefetch: Eligible
```

– 次のステートメントは、修飾行が索引によって識別およびアクセスされることを示します。

```
Index Scan:  Name = schema.name  ID = xx
| Index type
| Index Columns:
```

説明:

- *schema.name* は、スキャンする索引の完全修飾名です。
- *ID* は、 SYSCAT.INDEXES カタログ表の中の対応する ID 列です。
- 索引のタイプは、以下の 1 つです。

```
Regular Index (Not Clustered)
Regular Index (Clustered)
Dimension Block Index
Composite Dimension Block Index
```

その後には、索引の列ごとに 1 つの行が続きます。索引の各列は、以下のいずれかの形式でリストされます。

```
n: column_name (Ascending)
n: column_name (Descending)
n: column_name (Include Column)
```

次のステートメントは、索引スキャンのタイプを明確にするために提供されています。

- 索引の範囲区切り述部は、以下のようにして示されます。

```
#Key Columns = n
|  Start Key: xxxxx
|  Stop Key: xxxxx
```

ここで、xxxxx は以下のいずれかに該当します。

- Start of Index
- End of Index
- Inclusive Value: or Exclusive Value:

索引スキャンには、inclusive キー値が含まれます。スキャンに exclusive キー値は含まれません。キーの値は、キーの各部を構成する以下の行のいずれかによって指定されます。

```
n: 'string'
n: nnn
n: yyyy-mm-dd
n: hh:mm:ss
n: yyyy-mm-dd hh:mm:ss.uuuuuu
n: NULL
n: ?
```

リテラル・ストリングが示される場合は、最初の 20 文字だけが表示されます。ストリングの長さが 20 文字を超える場合、ストリングの終わりに

は ... が示されます。キーによっては、セクションが実行されるまで判別できないものがあります。その場合は、値として ? が示されます。

- Index-Only Access

  必要なすべての列が索引キーから入手できる場合、このステートメントが表示され、表データにはアクセスしません。

- 次のステートメントは、索引ページのプリフェッチが行われないことを示しています。

  ```
  Index Prefetch: None
  ```

- 次のステートメントは、索引ページをプリフェッチすることを示します。

  ```
  Index Prefetch: Eligible
  ```

- 次のステートメントは、データ・ページのプリフェッチが行われないことを示しています。

  ```
  Data Prefetch: None
  ```

- 次のステートメントは、データ・ページをプリフェッチすることを示します。

  ```
  Data Prefetch: Eligible
  ```

- 索引ガバナーに渡されて索引項目を修飾するのに役立つ述部があれば、述部の数を示すために次のステートメントを使用します。

  ```
  Sargable Index Predicate(s)
     |  #Predicates = n
  ```

– アクセス・プランですでに作成されている行 ID (RID) を使用して、修飾行にアクセスしている場合、それは次のステートメントによって示されます。

```
Fetch Direct Using Row IDs
```

表に 1 つ以上のブロック索引が定義されている場合、ブロックまたは行 ID を使用して行にアクセスすることができます。このことは、次に示します。

```
Fetch Direct Using Block or Row IOs
```

ロックの意図

表アクセスのたびに、表および行レベルで獲得されるロックのタイプを、次のステートメントを用いて表示することができます。

```
Lock Intents
|  Table: xxxx
|  Row  : xxxx
```

表ロックに可能な値は、以下のとおりです。

- Exclusive
- Intent Exclusive
- Intent None
- Intent Share
- Share
- Share Intent Exclusive
- Super Exclusive
- Update

行ロックに可能な値は、以下のとおりです。

- Exclusive
- Next Key Exclusive (db2expln 出力に表示されません)
- None
- Share
- Next Key Share
- Update
- Next Key Weak Exclusive
- Weak Exclusive

述部

アクセス・プランに用いられる述部に関する情報を提供するステートメントは、2 つあります。

1. 次のステートメントは、ブロック化索引から検索されたデータのブロックごとに、述部の数が評価されることを示します。

   ```
   Block Predicates(s)
      |  #Predicates = n
   ```

2. 次のステートメントは、データ・アクセス中に、述部の数が評価されることを示します。述部のカウントには、集約やソートなどのプッシュダウン操作は含まれていません。

   ```
   Sargable Predicate(s)
      |  #Predicates = n
   ```

3. 次のステートメントは、一度データが戻されたときに、述部の数が評価されることを示します。

   ```
   Residual Predicate(s)
      |  #Predicates = n
   ```

このステートメントに表示された述部の数は、 SQL ステートメント中の述部の数を反映していないことがあります。なぜなら、述部は次のような場合があるからです。

- 同じ照会内で複数回適用されます。
- 照会最適化処理時に暗黙述部が追加され、変形と拡張が行われます。
- 照会最適化処理時に変形と圧縮が行われ、述部が少なくなります。

その他の表ステートメント

- 次のステートメントは、1 行だけにアクセスできることを示します。

  ```
  Single Record
  ```

- 次のステートメントは、この表アクセスに使用されている分離レベルが、ステートメントとは異なる分離レベルを使用しているときに表示されます。

  ```
  Isolation Level: xxxx
  ```

  さまざまな理由により、異なる分離レベルを使用することも可能です。その理由としては、以下のことが挙げられます。

  - パッケージが反復可能読み取りにバインドされており、参照保全制約に影響を与える。参照保全制約を調べるために親表にアクセスすると、この表で不要なロックを保持しないように、カーソル固定の分離レベルにダウングレードします。
  - 非コミット読み取りにバインドされているパッケージが DELETE または UPDATE ステートメントを出しています。実際に削除するために表アクセスをすると、カーソル固定にアップグレードします。
- 次のステートメントは、使用可能なメモリーが十分ある場合に、一時表から読み取られた行の一部またはすべてがバッファー・プール以外にキャッシュされることを示します。

  ```
  Keep Rows In Private Memory
  ```
- 表に volatile のカーディナリティー属性セットがある場合、そのことが以下のように示されます。

  ```
  Volatile Cardinality
  ```

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

一時表の情報

一時表は、アクセス・プランを実行中にデータを一時的な作業表に保管するために使用されます。この表は、アクセス・プラン実行中にのみ存在します。通常は、アクセス・プランの初期段階で副照会の評価が必要になったとき、または中間結果が利用可能なメモリーに納まらないときに、一時ファイルが使用されます。

一時表を作成することが必要になると、 2 つのステートメントのうちのどちらかが表示されることがあります。これらのステートメントは、一時表を作成し、その表に行が挿入されるように指示します。 ID は、一時表を参照するときの便宜上 db2expln によって割り当てられる ID です。この ID は、接頭部として文字「t」が付き、その表が一時表であることを示します。

- 次のステートメントは、一般的な一時表が作成されることを示します。

  ```
  Insert Into Temp Table  ID = tn
  ```
- 次のステートメントは、通常の一時表が複数のサブエージェントによって並列に作成されることを示します。

  ```
  Insert Into Shared Temp Table  ID = tn
  ```
- 次のステートメントは、ソート済みの一時表が作成されることを示します。

  ```
  Insert Into Sorted Temp Table  ID = tn
  ```
- 次のステートメントは、ソート済みの一時表が複数のサブエージェントによって並列に作成されることを示します。

  ```
  Insert Into Sorted Shared Temp Table  ID = tn
  ```
- 次のステートメントは、宣言済みグローバル一時表が作成されることを示します。

  ```
  Insert Into Global Temp Table  ID = ts,tn
  ```
- 次のステートメントは、宣言済みグローバル一時表が複数のサブエージェントによって並列に作成されることを示します。

```
Insert Into Shared Global Temp Table  ID = ts,tn
```

- 次のステートメントは、ソートされた宣言済みグローバル一時表が作成されることを示します。

```
Insert Into Sorted Global Temp Table  ID = ts,tn
```

- 次のステートメントは、ソートされた宣言済みグローバル一時表が複数のサブエージェントによって並列に作成されることを示します。

```
Insert Into Sorted Shared Global Temp Table  ID = ts,tn
```

上記のステートメントの後には、いずれも次の 1 行を付加することができます。

```
#Columns = n
```

これは、一時表に挿入している各行を何列にするかを示します。

ソート済みの一時表

次のような操作から、ソート済みの一時表を作成することができます。

- ORDER BY
- DISTINCT
- GROUP BY
- Merge Join
- 「= ANY」副照会
- 「<> ALL」副照会
- INTERSECT または EXCEPT
- UNION (ALL キーワードの指定なし)

ソート済みの一時表を作成するための元のステートメントの後に、いくつかの追加ステートメントを付加することもできます。

- 次のステートメントは、ソートに使用するキー列の数を示します。

```
#Sort Key Columns = n
```

  ソート・キー中の列ごとに、次の行のうち 1 つが表示されます。

```
Key n: column_name (Ascending)
Key n: column_name (Descending)
Key n: (Ascending)
Key n: (Descending)
```

- 次のステートメントは、ランタイムに最適なソート・ヒープを割り当てることができるように、行数および行サイズの見積もりを提供します。

```
Sortheap Allocation Parameters:
|  #Rows     = n
|  Row Width = n
```

- ソート結果の先頭行だけが必要な場合は、次のように表示されます。

```
Sort Limited To Estimated Row Count
```

- 対称マルチプロセッサー (SMP) 環境でのソートの場合は、実行されるソートのタイプは次のようなステートメントのいずれかによって指示されます。

```
Use Partitioned Sort
Use Shared Sort
Use Replicated Sort
Use Round-Robin Sort
```

- 次のステートメントは、ソートの結果をソート・ヒープに残すかどうかを指定します。

  ```
  Piped
  ```

  および

  ```
  Not Piped
  ```

  パイプ・ソートが指示されている場合は、データベース・マネージャーは、ソートの出力をメモリーに保管して、ソート結果を別の一時表には入れません。
- 次のステートメントは、ソート中に重複値を除去することを示しています。

  ```
  Duplicate Elimination
  ```
- ソートの中で集約が行われている場合は、下記のステートメントのいずれかによって指示されます。

  ```
  Partial Aggregation
  Intermediate Aggregation
  Buffered Partial Aggregation
  Buffered Intermediate Aggregation
  ```

一時表の完了: 一時表を作成するためのプッシュダウン操作を含む表アクセス (つまり、表アクセスの有効範囲内で生じる一時表の作成) の後には、「完了 (completion)」ステートメントがあります。このステートメントは、一時表がそれ以後の一時表アクセスで行を提供できるようにしておくことによって、ファイルの終わりを処理します。次の行のうち 1 つが表示されます。

```
Temp Table Completion  ID = tn
Shared Temp Table Completion  ID = tn
Sorted Temp Table Completion  ID = tn
Sorted Shared Temp Table Completion  ID = tn
```

表関数

表関数とは、データを表の形式でステートメントに戻すユーザー定義関数 (UDF) のことです。表関数は以下によって示されます。

```
Access User Defined Table Function
|  Name = schema.funcname
|  Specific Name = specificname
|  SQL Access Level = accesslevel
|  Language = lang
|  Parameter Style = parmstyle
|  Fenced                      Not Deterministic
|  Called on NULL Input        Disallow Parallel
|  Not Federated               Not Threadsafe
```

特定の名前は起動される表関数を一意的に識別します。残りの行は、機能の属性を詳細に示しています。

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

結合の情報

次の 3 つのタイプの結合があります。

- ハッシュ結合

- マージ結合
- ネスト・ループ結合

結合セクションが実行される時になると、以下のステートメントのうち 1 つが表示されます。

```
Hash Join
Merge Join
Nested Loop Join
```

左方外部結合を行うこともできます。左方外部結合は、下記のステートメントのいずれかで指示されます。

```
Left Outer Hash Join
Left Outer Merge Join
Left Outer Nested Loop Join
```

マージ結合とネスト・ループ結合の場合、結合の外部表は、出力に表示されている直前のアクセス・ステートメント内で参照される表です。結合の内部表は、結合ステートメントの有効範囲内にあるアクセス・ステートメントで参照される表です。ハッシュ結合の場合、逆に結合の有効範囲内のアクセス・ステートメントによって参照される表が外部表になり、結合の前に表示されるアクセス・ステートメントによって参照される表が内部表になります。

ハッシュ結合とマージ結合の場合、次のような追加のステートメントが表示されます。

- ある種の環境では、結合は、内部表の中の任意の行が外部表の現在行と一致しているかだけを判別する必要があります。これは、次のステートメントで指示されます。

  ```
  Early Out: Single Match Per Outer Row
  ```

- 結合が完了したあとで、述部を適用することもできます。適用される述部の数は、次のように指示されます。

  ```
  Residual Predicate(s)
  |  #Predicates = n
  ```

ハッシュ結合の場合、次のような追加のステートメントが表示されます。

- ハッシュ表は内部表から作成されます。作成されたハッシュ表が、内部表のアクセスに関する述部にプッシュダウンされた場合、そのことが内部表のアクセスに関するステートメント中に次のように指示されます。

  ```
  Process Hash Table For Join
  ```

- 外部表にアクセスする際には、プローブ表が作成されて結合のパフォーマンスが向上します。プローブ表が作成された場合、そのことは外部表のアクセスに関する次のステートメントで指示されます。

  ```
  Process Probe Table For Hash Join
  ```

- ハッシュ表を作成するのに必要なバイト数の見積もりは次のように表されます。

  ```
  Estimated Build Size: n
  ```

- プローブ表に必要なバイト数の見積もりは次のように表されます。

  ```
  Estimated Probe Size: n
  ```

ネストしたループ結合の場合、次の追加ステートメントが結合ステートメントの直後に表示されます。

```
Piped Inner
```

このステートメントは、結合の内部表が別の一連の操作の結果であることを示します。これを、*composite inner* ともいいます。

結合が 3 つ以上の表に関係する場合、Explain ステップは上から下に読みます。たとえば、Explain 出力に次のような流れがあるとします。

```
Access ..... W
Join
|  Access ..... X
Join
|  Access ..... Y
Join
|  Access ..... Z
```

この場合、実行ステップは次のようになります。

1. W から修飾行を取り出す。
2. W からの行を、X からの行 (次の行) と結合し、結果 P1 (部分結合の結果番号 1) を呼び出す。
3. P1 を Y からの行 (次の行) と結合して、 P2 を作成します。
4. P2 を Z からの行 (次の行) と結合し、 1 つの完全結果行を入手します。
5. さらに行が Z にあれば、ステップ 4 に戻る。
6. さらに行が Y にあれば、ステップ 3 に戻る。
7. さらに行が X にあれば、ステップ 2 に戻る。
8. さらに行が W にあれば、ステップ 1 に戻る。

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

データ・ストリーム情報

アクセス・プラン内では、ある一連の操作から別の一連の操作へとデータの作成と流れを制御することがしばしば必要になります。データ・ストリームという概念を用いると、あるアクセス・プラン内での一群の操作を 1 つの単位として制御することが可能になります。データ・ストリームの先頭は、次のステートメントで示します。

```
Data Stream n
```

ここで、n は、参照を容易にするために db2expln によって割り当てられたユニークな ID です。データ・ストリームの末尾は、次のステートメントで示します。

```
End of Data Stream n
```

この 2 つのステートメントの間のすべての操作が、同一のデータ・ストリームの一部と見なされます。

データ・ストリームにはいくつもの特性があり、最初のデータ・ストリーム・ステートメントの後には 1 つまたは複数のステートメントに続けて、それらの特性を記述することができます。

- データ・ストリームの操作がアクセス・プランで以前に生成された値によって決まる場合、そのデータ・ストリームには、以下のマークが付けられます。

  ```
  Correlated
  ```

- ソート済みの一時表と同様、以下のステートメントは、データ・ストリームの結果をメモリーに保持するかどうかを示します。

  ```
  Piped
  ```

  および

  ```
  Not Piped
  ```

  一時表の場合にそうであったように、パイプ・データ・ストリームも、実行時にメモリーが十分になければディスクに書き込むことができます。アクセス・プランでは、いずれの場合にも対応できるようになっています。

- 次のステートメントは、このデータ・ストリームから要求されているのが 1 つのレコードだけであることを示します。

  ```
  Single Record
  ```

データ・ストリームがアクセスされると、次のステートメントが出力に表示されます。

```
Access Data Stream n
```

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

挿入、更新、および削除の情報

これらの SQL ステートメントの Explain テキストは、解説するまでもありません。これらの SQL 操作に使用可能なステートメント・テキストは、次のとおりです。

```
Insert: Table Name = schema.name ID = ts,n
Update: Table Name = schema.name ID = ts,n
Delete: Table Name = schema.name ID = ts,n
Insert: Hierarchy Table Name = schema.name ID = ts,n
Update: Hierarchy Table Name = schema.name ID = ts,n
Delete: Hierarchy Table Name = schema.name ID = ts,n
Insert: Materialized Query Table = schema.name ID = ts,n
Update: Materialized Query Table = schema.name ID = ts,n
Delete: Materialized Query Table = schema.name ID = ts,n
Insert: Global Temporary Table ID = ts, tn
Update: Global Temporary Table ID = ts, tn
Delete: Global Temporary Table ID = ts, tn
```

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

ブロックと行 ID の準備の情報

一部のアクセス・プランでは、実際の表アクセスが実行される前に、修飾行およびブロックの ID (ID) をソートしておき、重複を削除しておくか (index ORing の場合)、またはアクセスされているすべての索引に出てくる ID を識別するための手法を使用すると (index ANDing の場合)、効率がよくなります。 Explain ステートメントによって示される ID 作成には、主に 3 つの使用法があります。

- 次のステートメントはいずれも、Index ORing を使用して修飾 ID のリストを作成します。

  ```
  Index ORing Preparation
  Block Index ORing Preparation
  ```

  Index ORing は、複数の索引アクセスを作成し、その結果を結合して、アクセスされるいずれの索引にも出てくる別個の ID を組み込む技法を指します。 OR キーワードにより述部が接続される場合や、IN 述部がある場合に、オプティマイザーは索引の OR を考慮します。索引アクセスは、同一の索引または別々の索引に作成できます。
- ID 作成のもう 1 つの使用法は、以下のいずれかに示すように、リスト・プリフェッチ中に使用される入力データを作成することです。

  ```
  List Prefetch Preparation
  Block List Prefetch RID Preparation
  ```
- *Index ANDing* とは、複数の索引アクセスを作成し、その結果を結合して、アクセスされるすべての索引に出てくる ID を組み込む技法を指します。 Index ANDing 処理は、次のステートメントのいずれか開始されます。

  ```
  Index ANDing
  Block Index ANDing
  ```

  オプティマイザーが結果のセットのサイズを算定した場合は、下記のステートメントによって算定結果が示されます。

  ```
  Optimizer Estimate of Set Size: n
  ```

  Index ANDing フィルター操作は、ID を処理し、ビット・フィルター操作を使用して、アクセスされるすべての索引に出てくる ID を判別します。下記のステートメントは、index ANDing 用に ID が処理されていることを示します。

  ```
  Index ANDing Bitmap Build Using Row IDs
  Index ANDing Bitmap Probe Using Row IDs
  Index ANDing Bitmap Build and Probe Using Row IDs
  Block Index ANDing Bitmap Build Using Block IDs
  Block Index ANDing Bitmap Build and Probe Using Block IDs
  Block Index ANDing Bitmap Build and Probe Using Row IDs
  Block Index ANDing Bitmap Probe Using Block IDs and Build Using Row IDs
  Block Index ANDing Bitmap Probe Using Block IDs
  Block Index ANDing Bitmap Probe Using Row IDs
  ```

  オプティマイザーがビットマップ用に結果のセットのサイズを算定した場合は、次のステートメントによって算定結果が示されます。

  ```
  Optimizer Estimate of Set Size: n
  ```

どのタイプの ID 作成の場合でも、リスト・プリフェッチを実行できる場合は、次のステートメントを用いてそれが表示されます。

```
Prefetch: Enabled
```

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

集約の情報

集約は、SQL ステートメント述部によって指定される基準があれば、その基準に合致する行で実行されます。何らかの集約関数が実行されると、以下のステートメントのいずれかが表示されます。

```
Aggregation
Predicate Aggregation
Partial Aggregation
Partial Predicate Aggregation
Intermediate Aggregation
Intermediate Predicate Aggregation
Final Aggregation
Final Predicate Aggregation
```

述部集約とは、データに実際にアクセスするときに、集約操作がプッシュダウン式に述部として処理されることを表します。

上述の集約ステートメントのいずれの場合もその下に、実行される統計関数のタイプの指示があります。

```
Group By
Column Function(s)
Single Record
```

特定の列関数は、元の SQL ステートメントから引き出すことができます。単一のレコードは、MIN または MAX 演算の条件を満たす索引から取り出されます。

述部集約を使用すると、集約が示される表アクセス・ステートメントに続いて、集約「完了」になります。これは、各グループの完了またはファイルの終わりの際に必要とされる処理をすべて実行します。次の行のうちのいずれかが表示されます。

```
Aggregation Completion
Partial Aggregation Completion
Intermediate Aggregation Completion
Final Aggregation Completion
```

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

並列処理の情報

SQL ステートメントを並列で実行する場合 (パーティション内並列処理またはパーティション間並列処理のいずれかを使用して) は、特別な操作が必要になります。並列プランの操作について、以下で説明します。

- パーティション内並列プランを実行するときは、サブエージェントをいくつか使用してプランの部分が同時に実行されます。サブエージェントの作成は、次のステートメントによって指示されます。

```
Process Using n Subagents
```

- パーティション間並列プランを実行しているときは、セクションはいくつかのサブセクションに分けられます。各サブセクションは、1 つまたはいくつかのデータベース・パーティションに送られて、実行されます。重要なサブセクションは、コーディネーター・サブセクション です。コーディネーター・サブセクションは、あらゆるプランの中で最初のサブセクションです。これは、最初に制御を得るもので、他のサブセクションを分配したり、呼び出し側のアプリケーションに結果を戻したりする責任があります。

  サブセクションの分散は、次のステートメントによって指示されます。

  ```
  Distribute Subsection #n
  ```

  サブセクションを受け取るデータベース・パーティションは、以下の 8 つの方法のいずれかで判別することができます。

  – 以下のステートメントは、列の値に基づいて、データベース・パーティション・グループ内にあるデータベース・パーティションにサブセクションが送られることを示します。

    ```
    Directed by Hash
    |  #Columns = n
    |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
    ```

  – 以下のステートメントは、事前に決められたデータベース・パーティションにサブセクションが送られることを示します。 (これは、ステートメントが NODENUMBER() 関数を使用しているときに、よく見られます。)

    ```
    Directed by Node Number
    ```

  – 以下のステートメントは、指定のデータベース・パーティション・グループで事前に決められたパーティション番号に対応するデータベース・パーティションにサブセクションが送られることを示します。 (これは、ステートメントが PARTITION () 関数を使用しているときに、よく見られます。)

    ```
    Directed by Partition Number
    |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
    ```

  – 以下のステートメントは、アプリケーションのカーソル用の現在行を提示したデータベース・パーティションに、サブセクションが送られることを示します。

    ```
    Directed by Position
    ```

  – 以下のステートメントは、ステートメントがコンパイルされたときに判別された、ある 1 つのデータベース・パーティションだけがサブセクションを受け取ることを示します。

    ```
    Directed to Single Node
    |  Node Number = n
    ```

  – 以下のステートメントのいずれかは、コーディネーター・ノードに対してサブセクションが実行されることを示します。

    ```
    Directed to Application Coordinator Node
    Directed to Local Coordinator Node
    ```

  – 以下のステートメントは、リストされたすべてのデータベース・パーティションにサブセクションが送られることを示します。

    ```
    Broadcast to Node List
    |  Nodes = n1, n2, n3, ...
    ```

- 以下のステートメントは、ステートメントが実行されるときに判別された、ある 1 つのデータベース・パーティションだけがサブセクションを受け取ることを示します。

  ```
  Directed to Any Node
  ```

- パーティション・データベース環境内のサブセクション同士の間、または対称マルチプロセッサー (SMP) 環境にあるサブエージェント同士の間でデータを移動するために、表キューが使用されます。表キューは、次のように記述されます。
  - 下記のステートメントは、データが表キューに挿入されることを示します。

    ```
    Insert Into Synchronous Table Queue  ID = qn
    Insert Into Asynchronous Table Queue  ID = qn
    Insert Into Synchronous Local Table Queue  ID = qn
    Insert Into Asynchronous Local Table Queue  ID = qn
    ```

  - データベース・パーティション表キューの場合は、表キューに挿入された行の宛先は、以下のいずれかで記述されます。

    すべての行がコーディネーター・ノードに送られます。

    ```
    Broadcast to Coordinator Node
    ```

    指定のサブセクションが実行されているすべてのデータベース・パーティションに、すべての行が送られます。

    ```
    Broadcast to All Nodes of Subsection n
    ```

    行にある値に基づいて、各行がデータベース・パーティションに送られます。

    ```
    Hash to Specific Node
    ```

    各行は、ステートメントが実行されている間に決定されたデータベース・パーティションに送られます。

    ```
    Send to Specific Node
    ```

    各行は、ランダムに決定されたデータベース・パーティションに送られます。

    ```
    Send to Random Node
    ```

  - ある種の状況では、データベース・パーティション表キューは、一部の行を一時的に一時表にオーバーフローさせる必要があります。このような可能性は、次のステートメントによって識別します。

    ```
    Rows Can Overflow to Temporary Table
    ```

  - 表アクセスの際にプッシュダウン操作により行を表キューに挿入した場合、即時送信できなかった行について示した「完了」ステートメントがその後に表示されます。次の行のうちのいずれかが表示されます。

    ```
    Insert Into Synchronous Table Queue Completion  ID = qn
    Insert Into Asynchronous Table Queue Completion  ID = qn
    Insert Into Synchronous Local Table Queue Completion  ID = qn
    Insert Into Asynchronous Local Table Queue Completion  ID = qn
    ```

  - 下記のステートメントは、データが表キューから検索されることを示します。

    ```
    Access Table Queue  ID = qn
    Access Local Table Queue  ID = qn
    ```

    これらのメッセージの後には常に、検索される列の数の表示が付いています。

    ```
    #Columns = n
    ```

– 表キューが受信端で行をソートする場合は、表キュー・アクセスには、下記のメッセージのいずれかが出されます。

```
Output Sorted
Output Sorted and Unique
```

これらのメッセージの後には、ソート操作に使用されるキーの数の表示が付いています。

```
#Key Columns = n
```

ソート・キー中の列ごとに、次のうち 1 つが表示されます。

```
Key n: (Ascending)
Key n: (Descending)
```

– 表キューの受信端によって述部が行に適用される場合は、次のメッセージが表示されます。

```
Residual Predicate(s)
|  #Predicates = n
```

- パーティション・データベース環境にある一部のサブセクションは、次のステートメントを用いて、サブセクションの先頭まで、明示的にループバックします。

```
Jump Back to Start of Subsection
```

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

フェデレーテッド照会の情報

フェデレーテッド・データベースで SQL ステートメントを実行する場合、ほかのデータ・ソースに対してステートメントの部分を実行できなければなりません。

読み取られるデータ・ソースは以下のように指示されます。

```
Ship Distributed Subquery #n
|   #Columns = n
```

分散副照会から戻されるデータに述部に適用することができます。適用される述部の数は、次のように指示されます。

```
Residual Predicate(s)
|   #Predicates = n
```

データ・ソースで生じる挿入、更新、または削除操作は、適切なメッセージによって示されます。

```
Ship Distributed Insert #n
Ship Distributed Update #n
Ship Distributed Delete #n
```

表がデータ・ソースで明示的にロックされている場合、これは次のステートメントによって示されます。

```
Ship Distributed Lock Table #n
```

データ・ソースに対する DDL ステートメントは、 2 つの部分に分割されます。データ・ソースで呼び出される部分は、以下によって示されます。

```
Ship Distributed DDL Statement #n
```

フェデレーテッド・サーバーがパーティション・データベースである場合、 DDL ステートメントの一部はカタログ・ノードで実行する必要があります。このことは、次に示します。

```
Distributed DDL Statement #n Completion
```

各分散サブステートメントの詳細は、個別に指定されます。分散ステートメントのオプションは、以下のように記述されます。

- 副照会のデータ・ソースは、以下のいずれかで示されます。

  ```
  Server: server_name (type, version)
  Server: server_name (type)
  Server: server_name
  ```

- データ・ソースがリレーショナルである場合、サブステートメントの SQL は以下のように表示されます。

  ```
  SQL Statement:
       statement
  ```

  非リレーショナルのデータ・ソースは、以下によって示されます。

  ```
  Non-Relational Data Source
  ```

- サブステートメントで参照されるニックネームは、以下のようにリストされます。

  ```
  Nicknames Referenced:
     schema.nickname ID = n
  ```

  データ・ソースがリレーショナルである場合、ニックネームの基本表は以下のように示されます。

  ```
  Base = baseschema.basetable
  ```

  データ・ソースが非リレーショナルである場合、ニックネームのソース・ファイルは以下のように示されます。

  ```
  Source File = filename
  ```

- サブステートメントを実行する前にフェデレーテッド・サーバーからデータ・ソースに値が渡される場合、値の数は以下のように示されます。

  ```
  #Input Columns: n
  ```

- サブステートメントを実行した後でデータ・ソースからフェデレーテッド・サーバーに値が渡される場合、値の数は以下のように示されます。

  ```
  #Output Columns: n
  ```

関連概念:

- 204 ページの『フェデレーテッド照会を評価する場合の分析のガイドライン』
- 651 ページの『db2expln および dynexpln 出力の説明』

その他の情報

- データ定義言語ステートメントのセクションは、出力に次のように示されます。

  ```
  DDL Statement
  ```

  DDL ステートメントには、そのほかに Explain 出力はありません。

- 更新可能な特殊レジスター (**CURRENT EXPLAIN SNAPSHOT** など) 用の SET ステートメントのセクションは、出力に次のように示されます。

```
SET Statement
```

SET ステートメントには、そのほかに Explain 出力はありません。

- SQL ステートメントに DISTINCT 文節が含まれる場合、次のテキストが出力に表示されます。

  ```
  Distinct Filter  #Columns = n
  ```

  ここで、n は、入手している個別行に含まれる列の数です。個別行の値を検索するには、重複値をスキップできるように行を順序付ける必要があります。データベース・マネージャーが明示的に重複を除去する必要がなければ、このステートメントは表示されません。以下のような場合があります。

  – ユニーク索引が存在しており、索引キー内のすべての列が DISTINCT 操作の一部になっています。
  – ソート中に重複を除去することができます。

- 以下のステートメントは、次の操作が特定のレコード ID に依存している場合に表示されます。

  ```
  Positioned Operation
  ```

  位置操作がフェデレーテッド・データ・ソースに対するものである場合、ステートメントは以下のようになります。

  ```
  Distributed Positioned Operation
  ```

  このステートメントは、 WHERE CURRENT OF 構文を使用する SQL ステートメントに使われます。

- 次のステートメントは、結果には適用しなければならないが、別の操作の一部として適用することはできない述部がある場合に表示されます。

  ```
  Residual Predicate Application
  |  #Predicates = n
  ```

- 次のステートメントは、SQL ステートメントに UNION オペレーターがある場合に表示されます。

  ```
  UNION
  ```

- 次のステートメントは、後続の操作に使用される行の値を作成することだけを目的とした操作がアクセス・プラン内にある場合に表示されます。

  ```
  Table Constructor
  |  n-Row(s)
  ```

  表構成プログラムを使用して、1 つの集合として存在している値を一連の行に変形し、後続の操作に渡すことができます。表構成プログラムを次の行に入力するよう要求されると、次のステートメントが表示されます。

  ```
  Access Table Constructor
  ```

- 次のステートメントは、特定の条件の下でのみ処理される操作があるときに表示されます。

  ```
  Conditional Evaluation
  |  Condition #n:
  |  #Predicates = n
  |  Action #n:
  ```

条件付き評価は、SQL CASE ステートメントなどの活動や、参照保全制約やトリガーなどの内部機構を実行するときに使用します。処置に何も示されていない場合は、条件が真であるときにのみデータ操作命令が処理されます。

- ALL、ANY、または EXISTS 副照会がアクセス・プラン内で処理中である場合には、以下のステートメントのうちのいずれかが表示されます。
  - ANY/ALL Subquery
  - EXISTS Subquery
  - EXISTS SINGLE Subquery
- 特定の UPDATE 操作と DELETE 操作の前に、表中の特定の行の位置を確立する必要があります。このことは、次のステートメントで示します。

  ```
  Establish Row Position
  ```

- 次のステートメントは、アプリケーションに戻っている行がある場合に表示されます。

  ```
  Return Data to Application
  |   #Columns = n
  ```

  操作が表アクセスにプッシュダウンされる場合、完了フェーズが必要になります。このフェーズは、次のように表示されます。

  ```
  Return Data Completion
  ```

- ストアード・プロシージャーが呼び出されている場合、以下の情報が表示されます。

  ```
  Call Stored Procedure
  |   Name = schema.funcname
  |   Specific Name = specificname
  |   SQL Access Level = accesslevel
  |   Language = lang
  |   Parameter Style = parmstyle
  |   Expected Result Sets = n
  |   Fenced                      Not Deterministic
  |   Called on NULL Input        Disallow Parallel
  |   Not Federated               Not Threadsafe
  ```

- 1 つ以上の LOB ロケーターが解放されている場合、以下の情報が表示されます。

  ```
  Free LOB Locators
  ```

関連概念:

- 651 ページの『db2expln および dynexpln 出力の説明』
- 671 ページの『db2expln および dynexpln 出力の例』

db2expln および dynexpln 出力の例

次の出力例は、特定の環境での特定の照会について収集された Explain 情報を示しています。

db2expln および dynexpln 出力の例

ここに示されている 5 つの例は、 db2expln および dynexpln からの出力のレイアウトと形式を理解するために役立ちます。これらの例は、DB2® で提供される

SAMPLE データベースに対して実行されたものです。それぞれの例について、簡単な説明が添えられています。 1 つの例と次の例との重要な相違点は、**太字**で示してあります。

関連概念:

- 651 ページの『dynexpln』
- 672 ページの『例 1: 非並列』
- 674 ページの『例 2: パーティション内並列処理による単一パーティションのプラン』
- 675 ページの『例 3: パーティション間並列処理による複数パーティションのプラン』
- 678 ページの『例 4: パーティション間並列処理とパーティション内並列処理による複数パーティションのプラン』
- 680 ページの『例 5 : フェデレーテッド・データベースのプラン』

関連資料:

- 644 ページの『db2expln - SQL Explain』

例 1: 非並列

この例は、全従業員の名前、職種、部門名とその場所、および現在携わっているプロジェクト名のリストを要求するだけのものです。このアクセス・プランの特徴は、指定したそれぞれの表から関係するデータを結合するのにハッシュ結合を使用するという点です。索引を使用することができないので、アクセス・プランは各表が結合される際にリレーション・スキャンを行います。

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:05:00

 Bind Timestamp = 2002-01-04-14.05.00.415403

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 120.518692
```

```
Estimated Cardinality = 221.535980

(    6) Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  #Columns = 3
        |  Relation Scan
        |  |  Prefetch: Eligible
        |  Lock Intents
        |  |  Table: Intent Share
        |  |  Row  : Next Key Share
(    6) |  Process Build Table for Hash Join
(    2) Hash Join
        |  Estimated Build Size: 7111
        |  Estimated Probe Size: 9457
(    5) |  Access Table Name = DOOLE.PROJECT  ID = 2,7
        |  |  #Columns = 2
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    5) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 5737
        |  |  Estimated Probe Size: 6421
(    4) |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  #Columns = 3
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    4) |  |  |  Process Probe Table for Hash Join
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
              HSJOIN
              (   2)
              /     \
        HSJOIN       TBSCAN
        (   3)       (   6)
        /     \         |
   TBSCAN    TBSCAN   Table:
   (   4)    (   5)   DOOLE
     |         |      EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

アクセス・プランの最初の部分では、DEPARTMENT および PROJECT 表にアクセスし、ハッシュ結合を使ってそれらの表を結合します。 この結合の結果は EMPLOYEE 表に結合されます。結果行はアプリケーションに戻されます。

例 2: パーティション内並列処理による単一パーティションのプラン

この例は、最初の例と同じ SQL ステートメントを示していますが、この照会は、4-way の SMP マシン用にコンパイルされたものです。

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:12:38

 Bind Timestamp = 2002-01-04-14.12.38.732627

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 133.934692
Estimated Cardinality = 221.535980

(    2) Process Using 4 Subagents
(    7) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Parallel Scan
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    7)    |  Process Build Table for Hash Join
(    3)    Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 9457
(    6)    |  Access Table Name = DOOLE.PROJECT  ID = 2,7
        |  |  |  #Columns = 2
        |  |  |  Parallel Scan
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    6)    |  |  Process Build Table for Hash Join
(    4)    |  Hash Join
        |  |  |  Estimated Build Size: 5737
        |  |  |  Estimated Probe Size: 6421
```

```
(    5) |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  |  #Columns = 3
        |  |  |  |  Parallel Scan
        |  |  |  |  Relation Scan
        |  |  |  |  |  Prefetch: Eligible
        |  |  |  |  Lock Intents
        |  |  |  |  |  Table: Intent Share
        |  |  |  |  |  Row  : Next Key Share
(    5) |  |  |  |  Process Probe Table for Hash Join
(    2) |  Insert Into Asynchronous Local Table Queue  ID = q1
(    2) Access Local Table Queue  ID = q1  #Columns = 5
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

               RETURN
               (   1)
                 |
                LTQ
               (   2)
                 |
               HSJOIN
               (   3)
              /      \
         HSJOIN         TBSCAN
         (   4)         (   7)
        /      \          |
    TBSCAN    TBSCAN   Table:
    (   5)    (   6)   DOOLE
      |         |      EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

このプランは、最初の例のプランとほとんど同じです。主な相違は、プランが最初に開始されるときに 4 つのサブエージェントを作成すること、および、アプリケーションに戻す前におのおののサブエージェントの作業の結果を収集するために、プランの終了時に表キューを作成することです。

例 3: パーティション間並列処理による複数パーティションのプラン

この例は、最初の例と同じ SQL ステートメントを示していますが、この照会は、3 つのデータベース・パーティションからなるパーティション・データベースでコンパイルされたものです。

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:54:57

 Bind Timestamp = 2002-01-04-14.54.57.033666

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5
```

```
 Partition Parallel       = Yes
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 118.483406
Estimated Cardinality = 474.720032

        Coordinator Subsection:
(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

        Subsection #1:
(    8)    Access Table Queue  ID = q2  #Columns = 2
(    3)    Hash Join
           |  Estimated Build Size: 5737
           |  Estimated Probe Size: 8015
(    6)    |  Access Table Queue  ID = q3  #Columns = 3
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5333
           |  |  Estimated Probe Size: 6421
(    5)    |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
           |  |  |  #Columns = 3
           |  |  |  Relation Scan
           |  |  |  |  Prefetch: Eligible
           |  |  |  Lock Intents
           |  |  |  |  Table: Intent Share
           |  |  |  |  Row  : Next Key Share
(    5)    |  |  |  Process Probe Table for Hash Join
(    2)    Insert Into Asynchronous Table Queue  ID = q1
           |  Broadcast to Coordinator Node
           |  Rows Can Overflow to Temporary Table

        Subsection #2:
(    9)    Access Table Name = DOOLE.PROJECT  ID = 2,7
           |  #Columns = 2
           |  Relation Scan
           |  |  Prefetch: Eligible
           |  Lock Intents
           |  |  Table: Intent Share
           |  |  Row  : Next Key Share
(    9)    |  Insert Into Asynchronous Table Queue  ID = q2
           |  |  Hash to Specific Node
           |  |  Rows Can Overflow to Temporary Tables
(    8)    Insert Into Asynchronous Table Queue Completion  ID = q2
```

```
        Subsection #3:
(    7)    Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
           |  #Columns = 3
           |  Relation Scan
           |  |  Prefetch: Eligible
           |  Lock Intents
           |  |  Table: Intent Share
           |  |  Row  : Next Key Share
(    7)    |  Insert Into Asynchronous Table Queue  ID = q3
           |  |  Hash to Specific Node
           |  |  Rows Can Overflow to Temporary Tables
(    6)    Insert Into Asynchronous Table Queue Completion  ID = q3

End of section


Optimizer Plan:

                RETURN
                (   1)
                  |
                 BTQ
                (   2)
                  |
                HSJOIN
                (   3)
               /      \
          HSJOIN        DTQ
          (   4)       (   8)
         /      \        |
    TBSCAN      DTQ     TBSCAN
    (   5)     (   6)   (   9)
      |          |        |
 Table:        TBSCAN   Table:
 DOOLE         (   7)   DOOLE
 DEPARTMENT      |      PROJECT
             Table:
             DOOLE
             EMPLOYEE
```

このプランは、最初の例のプランと全く同じ内容ですが、セクションが 4 つのサブセクションに分けられています。サブセクションは、次のようなタスクを行います。

- **コーディネーター・サブセクション**。このサブセクションは、他のサブセクションを調整するものです。このプランでは、他のサブセクションを分散させ、アプリケーションに戻される結果を集めるために表キューを使用します。
- **サブセクション #1**。このサブセクションは表キュー q2 をスキャンし、ハッシュ結合を使って表キュー q3 からのデータと結合します。 2 番目のハッシュ結合は、DEPARTMENT 表のデータへの追加を行います。結合された行は次に、表キュー q1 を使ってコーディネーター・サブセクションに送られます。
- **サブセクション #2**。このサブセクションは、PROJECT 表をスキャンして結果を使用して特定のノードへハッシュします。これらの結果は、サブセクション #1 が読み取ります。
- **サブセクション #3**。このサブセクションは、EMPLOYEE 表をスキャンして結果を使用して特定のノードへハッシュします。これらの結果は、サブセクション #1 が読み取ります。

例 4: パーティション間並列処理とパーティション内並列処理による複数パーティションのプラン

この例は、最初の例と同じ SQL ステートメントを示していますが、この照会は、3 つのデータベース・パーティション (4-way SMP マシン上にある) から成るパーティション・データベースでコンパイルされたものです。

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:58:35

 Bind Timestamp = 2002-01-04-14.58.35.169555

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 145.198898
Estimated Cardinality = 474.720032

        Coordinator Subsection:
(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

        Subsection #1:
(    3)    Process Using 4 Subagents
(   10)    |  Access Table Queue  ID = q3  #Columns = 2
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5737
           |  |  Estimated Probe Size: 8015
(    7)    |  |  Access Table Queue  ID = q5  #Columns = 3
(    5)    |  |  Hash Join
           |  |  |  Estimated Build Size: 5333
           |  |  |  Estimated Probe Size: 6421
```

```
(    6)     |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
            |  |  |  |  #Columns = 3
            |  |  |  |  Parallel Scan
            |  |  |  |  Relation Scan
            |  |  |  |  |  Prefetch: Eligible
            |  |  |  |  Lock Intents
            |  |  |  |  |  Table: Intent Share
            |  |  |  |  |  Row  : Next Key Share
(    6)     |  |  |  |  Process Probe Table for Hash Join
(    3)     |  Insert Into Asynchronous Local Table Queue  ID = q2
(    3)     Access Local Table Queue  ID = q2  #Columns = 5
(    2)     Insert Into Asynchronous Table Queue  ID = q1
            |  Broadcast to Coordinator Node
            |  Rows Can Overflow to Temporary Table

        Subsection #2:
(   11)     Process Using 4 Subagents
(   12)     |  Access Table Name = DOOLE.PROJECT  ID = 2,7
            |  |  #Columns = 2
            |  |  Parallel Scan
            |  |  Relation Scan
            |  |  |  Prefetch: Eligible
            |  |  Lock Intents
            |  |  |  Table: Intent Share
            |  |  |  Row  : Next Key Share
(   11)     |  Insert Into Asynchronous Local Table Queue  ID = q4
(   11)     Access Local Table Queue  ID = q4  #Columns = 2
(   10)     Insert Into Asynchronous Table Queue  ID = q3
            |  Hash to Specific Node
            |  Rows Can Overflow to Temporary Tables

        Subsection #3:
(    8)     Process Using 4 Subagents
(    9)     |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
            |  |  #Columns = 3
            |  |  Parallel Scan
            |  |  Relation Scan
            |  |  |  Prefetch: Eligible
            |  |  Lock Intents
            |  |  |  Table: Intent Share
            |  |  |  Row  : Next Key Share
(    8)     |  Insert Into Asynchronous Local Table Queue  ID = q6
(    8)     Access Local Table Queue  ID = q6  #Columns = 3
(    7)     Insert Into Asynchronous Table Queue  ID = q5
            |  Hash to Specific Node
            |  Rows Can Overflow to Temporary Tables

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
              BTQ
             (   2)
                |
              LTQ
             (   3)
                |
              HSJOIN
             (   4)
            /      \
       HSJOIN       DTQ
       (   5)      (  10)
      /      \       |
```

```
  TBSCAN     DTQ      LTQ
  (   6)    (   7)   (  11)
    |         |        |
Table:       LTQ     TBSCAN
DOOLE       (   8)   (  12)
DEPARTMENT    |        |
            TBSCAN   Table:
            (   9)   DOOLE
              |      PROJECT
           Table:
           DOOLE
           EMPLOYEE
```

このプランは、3 番目の例にあるプランと似ていますが、複数のサブエージェントが各サブセクションを実行する点が異なります。また、各サブセクションの最後に、ローカル表キューが、すべてのサブエージェントの結果を収集してから、修飾行が 2 番目の表キューに挿入され、特定のノードでハッシュされる点も異なります。

例 5 : フェデレーテッド・データベースのプラン

この例は、最初の例と同じ SQL ステートメントを示していますが、この照会は、フェデレーテッド・データベースでコンパイルされたものです。この例では、表 DEPARTMENT および PROJECT はデータ・ソースに、表 EMPLOYEE はフェデレーテッド・サーバーにあります。

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/11
 Prep Time = 13:52:48

 Bind Timestamp = 2002-01-11-13.52.48.325413

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 1804.625000
Estimated Cardinality = 112000.000000

(    7) Ship Distributed Subquery #2
        |  #Columns = 2
(    2) Hash Join
        |  Estimated Build Size: 48444
```

```
        |  Estimated Probe Size: 232571
(    6) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    6) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 64606
(    4) |  |  Ship Distributed Subquery #1
        |  |  |  #Columns = 3
(    1)    Return Data to Application
           |  #Columns = 5

Distributed Substatement #1:
(    4) Server: REMOTE  (DB2/UDB 8.1)
       SQL Statement:

            SELECT A0."DEPTNO", A0."DEPTNAME", A0."LOCATION"
            FROM "DOOLE"."DEPARTMENT" A0

       Nicknames Referenced:
          DOOLE.DEPARTMENT  ID = 32768
             Base = DOOLE.DEPARTMENT
       #Output Columns = 3

Distributed Substatement #2:
(    7) Server: REMOTE  (DB2/UDB 8.1)
       SQL Statement:

            SELECT A0."DEPTNO", A0."PROJNAME"
            FROM "DOOLE"."PROJECT" A0

       Nicknames Referenced:
          DOOLE.PROJECT  ID = 32769
             Base = DOOLE.PROJECT
       #Output Columns = 2

End of section

Optimizer Plan:

               RETURN
               (   1)
                 |
               HSJOIN
               (   2)
              /      \
       HSJOIN          SHIP
       (   3)         (   7)
      /      \          |
   SHIP      TBSCAN   Nickname:
  (   4)     (   6)   DOOLE
    |          |      PROJECT
Nickname:   Table:
DOOLE       DOOLE
DEPARTMENT  EMPLOYEE
```

このプランは、最初の例のプランと全く同じ内容ですが、 2 つの表のデータはデータ・ソースから取られます。この 2 つの表には、分散副照会を使用してアクセスし

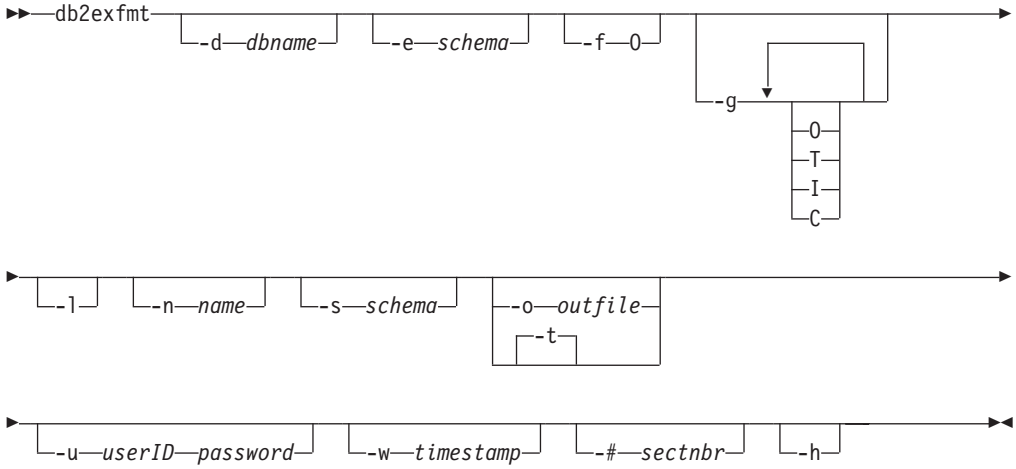
ます。この例では、単純にそれらの表のすべての行を選択しています。データがフェデレーテッド・サーバーに戻されると、そのデータはローカル表から取られたデータと結合させられます。

付録 D. db2exfmt - Explain 表形式

db2exfmt ツールを使用して、Explain 表の内容をフォーマットして出力することができます。

このツールを使用するには、フォーマットする Explain 表に対する読み取りアクセスが必要です。

コマンド構文:



コマンド・パラメーター:

-d *dbname*
パッケージを含むデータベースの名前。

-e *schema*
Explain 表のスキーマ。

-f フォーマット・フラグ。このリリースでは、O (オペレーターのサマリー) だけがサポートされています。

-g グラフのプラン。 -g だけを指定すると、グラフの後に、すべての表に関するフォーマットされた情報が生成されます。他にも指定する場合は、以下の有効値の組み合わせを指定できます。

- **O** グラフだけを生成します。表の内容はフォーマットされません。
- **T** グラフ中で、各オペレーターの下に合計コストが示されます。
- **I** グラフ中で、各オペレーターの下に入出力コストが示されます。
- **C** グラフ中で、各オペレーターの予期出力のカーディナリティー (タプルの数) が示されます。

-l パッケージ名の処理時に大文字小文字を考慮します。

-n *name*
Explain 要求のソースの名前 (SOURCE_NAME)。

-s *schema*
Explain 要求のソースのスキーマまたは修飾子 (SOURCE_SCHEMA)。

-o *outfile*
出力ファイル名。

-t 出力の宛先を端末にします。

-u *userID password*
データベースに接続するときは、与えられたユーザー ID とパスワードを使用してください。

ユーザー ID とパスワードはどちらも、命名規則に従った有効なもので、さらにデータベースが認識できるものでなければなりません。

-w *timestamp*
タイム・スタンプを Explain します。最新の Explain 要求を取得するには、-1 を指定します。

-# *sectnbr*
ソース中のセクション番号。すべてのセクションを要求するには、ゼロを指定します。

-h ヘルプ情報を表示します。このオプションを指定すると、他のオプションはすべて無視され、ヘルプ情報が表示されます。

使用上の注意:

-h オプションと -l オプションの場合を除いて、パラメーター値を指定していなかったり指定が誤っていたりするとそのことを指摘されます。

Explain 表のスキーマを指定しないと、環境変数 **USER** の値がデフォルトとして使用されます。この変数がない場合は、 Explain 表スキーマを指定するよう指示されます。

ソース名、ソースのスキーマ、および Explain タイム・スタンプは LIKE 述部の形式で指定できます。その際、パターン照合文字としてパーセント記号 (%) と下線 (_) を使用して、 1 回の呼び出しで複数のソースを選択できます。最新の Explain ステートメントの場合、 Explain 時を -1 と指定できます。

ファイル名を指定せずに -o を指定し、しかも -t を指定しないと、ファイル名 (デフォルトの名前は db2exfmt.out) を指定するよう指示されます。 -o と -t を両方とも指定しないと、ファイル名 (デフォルト・オプションは端末出力) を指定するよう指示されます。 -o と -t を両方とも指定すると、出力は端末に送信されます。

関連概念:

- 214 ページの『Explain ツール』
- 216 ページの『Explain 情報の使用のガイドライン』
- 223 ページの『Explain 情報のキャプチャーのガイドライン』

付録 E. db2adutl コマンドと logarchopt1 および vendoropt データベース構成パラメーターを使ったノード間リカバリー

以下の例は、db2adutl コマンドと、*logarchopt1* および *vendoropt* データベース構成パラメーターを使用して、ノード間リカバリーを実行する方法の例です。

以下の例で、コンピューター 1 は bar という名前で、AIX を実行しています。このマシンの所有者は roecken です。 bar 上のデータベースは zample という名前です。コンピューター 2 は dps という名前です。このマシンも AIX を実行しており、所有者は regress9 です。

PASSWORDACCESS = generate:

コンピューター 1:

1. ログ・アーカイブ用のデータベースを TSM にセットアップする。 zample データベースのデータベース構成パラメーター *logarchmeth1* を更新します。

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   次の情報が戻されます。

   ```
   DB20000I  UPDATE DATABASE CONFIGURATION コマンドは正常に完了しました。
   ```

   注: データベース構成を更新する前に、データベースのオフライン・バックアップを取っておく必要があるかもしれません。

2. データベースのオンライン・バックアップを取る。

   ```
   db2 backup db zample online use tsm
   ```

   次の情報が戻されます。

   ```
   バックアップに成功しました。
   このバックアップ・イメージのタイム・スタンプ: 20040216151025
   ```

3. zample データベースに接続し、そこに表を作成する。
4. 新しい表にデータをロードする。この例では、表を a とし、区切り文字で区切られている ASCII ファイル mr からデータをロードするものとします。COPY YES オプションを指定することにより、ロードするデータのコピーを作成します。また、USE TSM オプションにより、データのコピーを Tivoli Storage Manager に保管するよう指定します。

   注: COPY YES オプションを指定できるのは、データベースがロールフォワード・リカバリーを使用可能な場合だけです。つまり、データベースに対して *logretain* または *userexit* データベース構成パラメーター (あるいはその両方) が使用可能でなければなりません。

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader replace
   into a copy yes use tsm
   ```

   ユーティリティーは、進行状況を示すために一連のメッセージを戻します。

   ```
   SQL3109N  ユーティリティーはファイル "/home/roecken/mr" からのデータのロードを開始します。
   ```

```
SQL3500W  ユーティリティーは "LOAD" フェーズを "02/16/2004
15:12:13.392633" に開始します。

SQL3519W  ロード整合点を開始します。入力レコード・カウント = "0"。

SQL3520W  ロード整合点は成功しました。

SQL3110N  ユーティリティーは処理を完了しました。 "1" 行が入力ファイルから
読み取られました。

SQL3519W  ロード整合点を開始します。入力レコード・カウント = "1"。

SQL3520W  ロード整合点は成功しました。

SQL3515W  ユーティリティーは "LOAD" フェーズを "02/16/2004
15:12:13.445718" に終了しました。


読み取った行の数              = 1
スキップした行の数            = 0
ロードした行の数              = 1
リジェクトした行の数          = 0
削除した行の数                = 0
コミットした行の数            = 1
```

この時点で、TSM には 1 つのバックアップ・イメージ、1 つのロード・コピー、および 1 つのログ・ファイルがあるはずです。 zample データベースに対する照会は、次のようにして実行できます。

```
bar:/home/roecken> db2adutl query db zample
```

次の情報が戻されます。

```
データベース全体のバックアップ情報を検索します。
   1 時刻: 20040216151025  最も古いログ: S0000000.LOG  DB パーティション番号: 0
   セッション: 1
増分データベース・バックアップ情報を検索します。
ZAMPLE の増分データベース・バックアップ・イメージが見つかりません。


差分データベース・バックアップ情報を検索します。
ZAMPLE の差分データベース・バックアップ・イメージが見つかりません。


表スペース・バックアップ情報を検索します。
ZAMPLE の表スペース・バックアップ・イメージが見つかりません。


増分表スペース・バックアップ情報を検索します。
ZAMPLE の増分表スペース・バックアップ・イメージが見つかりません。


差分表スペース・バックアップ情報を検索します。
ZAMPLE の差分表スペース・バックアップ・イメージが見つかりません。


コピーのロード情報を検索します。
1 時刻: 20040216151213


ログ・アーカイブ情報を検索します。
  ログ・ファイル: S0000000.LOG、チェーン番号: 0、DB パーティション番号: 0、
  実行時: 2004-02-16-15.10.38
```

5. ノード間リカバリーを使用可能にするには、もう一方のノードとアカウントに bar コンピューターのオブジェクトへのアクセスを与える必要がある。この例では、ノード dps とユーザー regress9 にアクセスを与えます。

```
bar:/home/roecken> db2adutl grant user regress9 on nodename dps for db zample
```

次の情報が戻されます。

```
regress9 がノード dps 上の ZAMPLE にアクセスするための許可が正常に追加されました。
```

db2adutl GRANT 操作の結果を照会するには、次のコマンドを発行します。

```
bar:/home/roecken> db2adutl queryaccess
```

次の情報が戻されます。

```
ノード              ユーザー名       データベース名       タイプ
------------------------------------------------------------
DPS                 regress9            ZAMPLE          A
------------------------------------------------------------
 アクセス・タイプ:  B - バックアップ・イメージ    L - ログ    A - 両方
```

PASSWORDACCESS = generate 環境:

コンピューター 2:

コンピューター 2 (dps) はまだセットアップされていません。 dps の zample データベースに **db2adutl** 照会を実行すると、次の結果が戻されます。

```
dps:/home/regress9> db2adutl query db zample
--- データベース・ディレクトリーは空です ---
警告: DB2 により ADSM サーバー上に作成されたファイル・スペースはありません。
警告: 別名用の DB2 バックアップ・イメージが ADSM に見つかりません。


dps:/home/regress9> db2adutl query db zample nodename bar owner roecken
--- データベース・ディレクトリーは空です ---

データベース ZAMPLE の照会


データベース全体のバックアップ情報を検索します。
   1 時刻: 20040216151025  最も古いログ: S0000000.LOG  DB パーティション番号: 0
   セッション: 1
増分データベース・バックアップ情報を検索します。
ZAMPLE の増分データベース・バックアップ・イメージが見つかりません。


差分データベース・バックアップ情報を検索します。
ZAMPLE の差分データベース・バックアップ・イメージが見つかりません。


表スペース・バックアップ情報を検索します。
ZAMPLE の表スペース・バックアップ・イメージが見つかりません。


増分表スペース・バックアップ情報を検索します。
ZAMPLE の増分表スペース・バックアップ・イメージが見つかりません。


差分表スペース・バックアップ情報を検索します。
ZAMPLE の差分表スペース・バックアップ・イメージが見つかりません。


コピーのロード情報を検索します。
1 時刻: 20040216151213
```

```
ログ・アーカイブ情報を検索します。
  ログ・ファイル: S0000000.LOG、チェーン番号: 0、DB パーティション番号: 0、
  実行時: 2004-02-16-15.10.38
```

dps コンピューターにはまだ zample データベースが存在していません。

1. zample データベースを dps コンピューターにリストアする。

```
dps:/home/regress9> db2 restore db zample use tsm options
"'-fromnode=bar -fromowner=roecken'" without prompting
```

次の情報が戻されます。

```
DB20000I  RESTORE DATABASE コマンドは正常に完了しました。
```

注: dps に zample データベースがすでに存在している場合は、 OPTIONS パラメーターを省略し、データベース構成パラメーター *vendoropt* を使用することになります。この構成パラメーターは、バックアップまたはリストア操作の OPTIONS パラメーターをオーバーライドします。

zample データベースに対してロールフォワード操作を実行すると、失敗することになります。ロールフォワード・ユーティリティーがログ・ファイルを見つけられないためです。ロールフォワード操作の一例は次のとおりです。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次のエラーが戻されます。

```
SQL4970N  データベース "ZAMPLE" のロールフォワード・リカバリーは、
ノード "0" 上にログ・ファイルがないため、指定した停止点 (ログの終わり
またはポイント・イン・タイム) に到達できません。
```

2. 強制的にロールフォワード・ユーティリティーに別のマシンでログ・ファイルを探させるため、適切な *logarchopt* 値を構成する必要がある。この状況では、*logarchopt1* データベース構成パラメーターになります。

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
"'-fromnode=bar -fromowner=roecken'"
```

3. ロールフォワード・ユーティリティーがロード・コピー・イメージを使用できるようにするため、 *vendoropt* データベース構成パラメーターも設定する必要がある。

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

4. ここで zample データベースがロールフォワード可能になる。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次の情報が戻されます。

```
                                  ロールフォワード状況

 入力データベース別名                 = zample
 状況を返したノードの数               = 1

 ノード番号                           = 0
 ロールフォワード状況                 = 未ペンディング
 次に読み込むログ・ファイル           =
 処理したログ・ファイル               = S0000000.LOG - S0000000.LOG
 最後にコミットしたトランザクション   = 2004-02-16-20.10.38.000000

DB20000I  ROLLFORWARD コマンドは正常に完了しました。
```

PASSWORDACCESS = prompt 環境:

PROMPT 環境では、余分の情報が必要です。特に、オブジェクトを作成したマシンの TSM ノード名とパスワードです。

db2adutl に関して、 dsm.sys ファイル (Windows ベースのプラットフォームでは *dsm.opt* ファイル) を更新し、NODENAME bar をサーバー文節に追加します (bar はソース・コンピューターの名前であるため)。

```
dps:/home/regress9> db2adutl query db zample nodename bar owner roecken password *******
```

次の情報が戻されます。

```
データベース ZAMPLE の照会


データベース全体のバックアップ情報を検索します。
   1 時刻: 20040216151025  最も古いログ: S0000000.LOG  DB パーティション番号: 0
   セッション: 1
増分データベース・バックアップ情報を検索します。
ZAMPLE の増分データベース・バックアップ・イメージが見つかりません。


差分データベース・バックアップ情報を検索します。
ZAMPLE の差分データベース・バックアップ・イメージが見つかりません。


表スペース・バックアップ情報を検索します。
ZAMPLE の表スペース・バックアップ・イメージが見つかりません。


増分表スペース・バックアップ情報を検索します。
ZAMPLE の増分表スペース・バックアップ・イメージが見つかりません。


差分表スペース・バックアップ情報を検索します。
ZAMPLE の差分表スペース・バックアップ・イメージが見つかりません。


コピーのロード情報を検索します。
1 時刻: 20040216151213


ログ・アーカイブ情報を検索します。
  ログ・ファイル: S0000000.LOG、チェーン番号: 0、DB パーティション番号: 0、
  実行時: 2004-02-16-15.10.38
```

1. データベースが存在しない場合は、空の zample データベースが作成される。zample データベースがすでに存在している場合は、このステップと、データベース構成を更新するための次の 2 つのステップをスキップできます。

   ```
   dps:/home/regress9> db2 create db zample
   ```

2. zample データベースのデータベース構成パラメーター *tsm_nodename* を更新する。

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

3. zample データベースのデータベース構成パラメーター *tsm_password* を更新する。

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_password ********
   ```

4. zample データベースをリストアする。

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

リストア操作は正常に完了するものの、警告が出されます。

```
SQL2540W  リストアは成功しましたが、非割り込みモードでデータベースを
リストア中に、警告 "2523" が出されました。
```

ここでも、ロールフォワード・ユーティリティーは正しいログ・ファイルを見つけることができません。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次のエラー・メッセージが戻されます。

```
SQL1268N  ノード "0" 上のデータベース "ZAMPLE" のログ・ファイル
"S0000000.LOG" の検索中に、エラー "-2112880618" のために
ロールフォワード・リカバリーが停止されました。
```

5. データベースのリストア操作でデータベース構成ファイルが置き換えられるため、 TSM データベース構成値を正しい値に設定する必要がある。最初に *tsm_nodename* 構成パラメーターをリセットする必要があります。

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

6. *tsm_password* データベース構成パラメーターをリセットする必要がある。

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
   ```

7. ロールフォワード・ユーティリティーが正しいログ・ファイルを見つけられるよう、 *logarchopt1* データベース構成パラメーターをリセットする必要がある。

   ```
   dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
   ```

8. ロード・リカバリー・ファイルも使用できるよう、 *vendoropt* データベース構成パラメーターもリセットする必要がある。

   ```
   dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
   ```

9. データベース構成パラメーターをセットすると、データベースをロールフォワードできるようになる。

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

zample データベースに対して ROLLFORWARD QUERY STATUS コマンドを実行すると、次のように表示されます。

```
                                  ロールフォワード状況

 入力データベース別名                 = zample
 状況を返したノードの数               = 1

 ノード番号                           = 0
 ロールフォワード状況                 = 未ペンディング
 次に読み込むログ・ファイル           =
 処理したログ・ファイル               = S0000000.LOG - S0000000.LOG
 最後にコミットしたトランザクション   = 2004-02-16-20.10.38.000000

DB20000I  ROLLFORWARD コマンドは正常に完了しました。
```

関連資料:

- 「コマンド・リファレンス」の『db2adutl - TSM 内の DB2 オブジェクトの管理コマンド』
- 466 ページの『logarchopt1 - 1 次ログ・アーカイブ・オプション』
- 474 ページの『vendoropt - ベンダー・オプション』

付録 F. DB2 Universal Database の技術情報

DB2 資料とヘルプ

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
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- ダウンロード可能な PDF ファイル、CD 上の PDF ファイル、および印刷された資料
  - ガイド
  - リファレンス・マニュアル
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ
  - SQL 状態ヘルプ
- インストール済みソース・コード
  - サンプル・プログラム

ibm.com® にある技術資料、白書、 Redbooks™ その他の DB2 Universal Database™ 技術情報にオンライン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 (www.ibm.com/software/data/pubs/) にアクセスしてください。

DB2 資料の更新

IBM® は、 DB2 インフォメーション・センターの資料のフィックスパックやその他の資料更新を定期的に発行しています。 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) にアクセスすれば、常に最新の情報が掲載されます。 DB2 インフォメーション・センターをローカル・インストールしている場合、更新記事を表示するには、まず手動で更新をインストールしてください。新しい情報が発表されたときに資料を更新することにより、 *DB2 インフォメーション・センター CD* からインストールした情報を更新することができます。

インフォメーション・センターの方が、PDF 資料やハードコピー資料よりも頻繁に更新されます。 DB2 の最新の技術情報を入手するには、資料更新が発行されたときにそれをインストールするか、または www.ibm.com サイトの DB2 インフォメーション・センターにアクセスしてください。

関連概念:

- 「*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*」の『CLI サンプル・プログラム』

- 「*アプリケーション開発ガイド アプリケーションの構築および実行*」の『Java サンプル・プログラム』
- 692 ページの『DB2 インフォメーション・センター』

関連タスク:

- 713 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 703 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 714 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 714 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 715 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 705 ページの『DB2 PDF 資料および印刷された資料』

DB2 インフォメーション・センター

DB2® インフォメーション・センターを使用すると、 DB2 Universal Database™、 DB2 Connect™、 DB2 Information Integrator および DB2 Query Patroller™ などの DB2 ファミリー製品を最大限に活用するのに必要なすべての情報にアクセスできます。また、DB2 インフォメーション・センターは、 DB2 の主な機能とコンポーネントに関する情報を提供します (レプリケーション、データウェアハウジング、および DB2 の種々の Extender など)。

Mozilla 1.0 以上または Microsoft® Internet Explorer 5.5 以上で表示する場合、 DB2 インフォメーション・センターには以下の機能があります。以下のいくつかの機能では、 JavaScript™ のサポートを使用可能にする必要があります:

柔軟なインストール・オプション
以下の中から、 ご使用の環境に最も適したオプションを使って DB2 資料を表示できます。

- 最新の資料を常に自動的に利用できるようにするには、 IBM® の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターからすべての資料に直接アクセスします。
- 更新処理を最小化し、イントラネット内のネットワーク・トラフィックだけに制限するには、イントラネット上の 1 つのサーバーに DB2 資料をインストールします。
- 柔軟性を改善し、ネットワーク接続への依存を軽減するには、個々のコンピューターに DB2 資料をインストールします。

検索 「**検索**」テキスト・フィールドに検索語を入力することにより、 DB2 インフォメーション・センターのすべてのトピックを検索できます。複数の語句を引用符で囲めば、完全一致を検索できます。また、ワイルドカード演算子 (*、?) とブール演算子 (AND、 NOT、OR) を使用して検索を絞り込むことができます。

タスク指向の目次
単一の目次の中から、DB2 資料のトピックを見付けることができます。目

次は、主に実行するタスクの種類に従って編成されていますが、そのほかに製品概要、特定のゴール (目的) の情報、参照情報、索引、および用語集も含まれます。

- 製品概要では、DB2 ファミリーで使用可能な製品間の関係、そうした各製品で提供される機能、および各製品の最新リリース情報について説明されています。
- インストール、管理および開発などのゴール・カテゴリーには、タスクを迅速に完了し、そのための背景情報をよく理解できるようにするトピックが含まれています。
- 「参照」トピックでは、その対象に関する詳細な情報 (ステートメントとコマンドの構文、メッセージ・ヘルプ、構成パラメーターなど) が説明されています。

現在のトピックを目次に表示する

現在のトピックが目次のどの部分に該当するかを表示するには、目次フレーム内の**「リフレッシュ/現在のトピックの表示 (Refresh/Show Current Topic)」**ボタンをクリックするか、コンテンツ・フレーム内の**「目次に表示 (Show in Table of Contents)」**ボタンをクリックします。幾つかのファイルで関連トピックへの複数のリンクをたどった場合、または検索結果からトピックにアクセスした場合には、この機能が役立ちます。

索引 索引から、すべての資料にアクセスすることができます。索引では、用語が 50 音順に編成されています。

用語集 用語集を見れば、DB2 資料で使われているさまざまな用語の定義を調べることができます。用語集では、用語が 50 音順に編成されています。

組み込まれているローカライズ情報

DB2 インフォメーション・センターは、ブラウザーで設定された言語でトピックを表示します。設定された言語のトピックが利用できない場合、DB2 インフォメーション・センターにはそのトピックの英語版が表示されます。

iSeries™ 技術情報については、 IBM eServer™ iSeries Information Center (www.ibm.com/eserver/iseries/infocenter/) を参照してください。

関連概念:

- 694 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 703 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 704 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 702 ページの『DB2 インフォメーション・センターの呼び出し』
- 696 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 699 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 インフォメーション・センターのインストール・シナリオ

さまざまに異なる業務環境のもとでは、 DB2® 情報にどのようにアクセスするかの要件もそれぞれ異なります。 DB2 インフォメーション・センターにアクセスするには、 IBM® の Web サイト、サーバーまたは組織のネットワーク、あるいはコンピューターへのインストールという 3 つの方法が可能です。この 3 つのケースのいずれも、資料は DB2 インフォメーション・センター内に置かれます。インフォメーション・センターは、ブラウザーを使って表示できるように設計されたトピック・ベースの情報の Web サイトです。デフォルトでは、DB2 製品から、 IBM Web サイト上の DB2 インフォメーション・センターにアクセスします。これに対して、 イントラネット・サーバーまたはご自分のコンピューターから DB2 インフォメーション・センターにアクセスしたい場合、製品メディア・パック内にある DB2 インフォメーション・センター CD から DB2 インフォメーション・センターをインストールする必要があります。以下では、DB2 資料へのアクセス・オプションの要約、および 3 つのインストール・シナリオを示します。これを参考にして、お客様の業務環境で DB2 インフォメーション・センターにアクセスするにはどの方法が最適か、どのようなインストール上の問題に配慮する必要があるかを判別してください。

DB2 資料にアクセスするオプションの要約:

以下の表は、お客様の実際の業務環境で、 DB2 インフォメーション・センターの DB2 製品情報にアクセスする方法としてどんなオプションが推奨されるかを示します。

インターネット・アクセス	イントラネット・アクセス	推奨されるアクション
はい	はい	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス、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・センターへのアクセス
はい	いいえ	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス
いいえ	はい	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・センターへのアクセス
いいえ	いいえ	ローカル・コンピューター上の DB2 インフォメーション・センターへのアクセス

シナリオ: コンピューター上の DB2 インフォメーション・センターへのアクセス:

Tsu-Chen 氏は小さな町で工場を経営していますが、その町には、インターネット・アクセスを提供する地元のインターネット・サービス・プロバイダーがありません。彼は、在庫、製品オーダー、銀行口座情報、および営業経費を管理するために DB2 Universal Database™ を購入しました。 Tsu-Chen 氏は以前に DB2 製品を利用したことがないので、 DB2 の使用方法を習得するために、DB2 製品資料を参照する必要があります。

Tsu-Chen 氏は 標準インストール・オプションを使って DB2 Universal Database を自分のコンピューターにインストールした後、 DB2 資料にアクセスしようとします。しかし、開こうとしているページが見つからないというエラー・メッセージがブラウザーから通知されました。 Tsu-Chen 氏は DB2 製品のインストール・マニュアルを調べた結果、 DB2 資料を自分のコンピューター上で利用するには、 DB2 インフォメーション・センターをインストールしなければならないことに気付きます。そしてメディア・パックの中にあった *DB2 インフォメーション・センター CD* を見つけ出して、インストールしました。

これで、Tsu-Chen 氏はオペレーティング・システムのアプリケーション・ランチャーから DB2 インフォメーション・センターにアクセスできるようになり、より良い業務成果をあげるために DB2 製品を利用する方法を習得できます。

シナリオ: IBM Web サイト上の DB2 インフォメーション・センターへのアクセス:

Colin は、あるセミナー企業に所属する情報技術コンサルタントです。彼の専門はデータベース・テクノロジーおよび SQL で、 DB2 Universal Database を使って北米一帯の企業を対象にこれらの科目のセミナーを開催しています。 Colin のセミナーでは、教材として DB2 資料も使用されます。たとえば、SQL の講習コースでは、データベース照会の基本構文と拡張構文を教えるために SQL に関する DB2 資料が使用されます。

Colin が教えている企業の大半はインターネット・アクセスを配備しています。このような状況から判断して、Colin は、最新バージョンの DB2 Universal Database を自分のモバイル・コンピューターにインストールしたとき、 IBM Web サイト上の DB2 インフォメーション・センターにアクセスするよう構成しました。この構成によって、Colin はセミナーで教えるときに最新の DB2 資料にオンライン・アクセスすることができます。

しかし、時折、Colin は移動中にインターネット・アクセスを利用できないことがあります。これは問題となります。担任するセミナーの準備のために DB2 資料にアクセスする必要のある場合には、とくにそうです。このような事態が起きないようにするために、 Colin は自分のモバイル・コンピューターに DB2 インフォメーション・センターのコピーをインストールしました。

こうして、Colin は常に DB2 資料のコピーを自在に活用できるようになりました。**db2set** コマンドを使って自分のモバイル・コンピューターのレジストリー変数を簡単に構成し、どこにいるかに応じて、IBM Web サイトまたは自分のモバイル・コンピューターから DB2 インフォメーション・センターにアクセスできます。

シナリオ: イントラネット・サーバー上の DB2 インフォメーション・センターへのアクセス:

Eva は、生命保険会社のデータベース上級管理者です。彼女は管理業務の一環として、会社の UNIX® データベース・サーバーに最新バージョンの DB2 Universal Database をインストールおよび構成します。彼女の会社は最近、セキュリティー上の理由から、インターネット・アクセスをもはや業務で利用できないようにすると社員に通知しました。同社はネットワーク環境を装備しているため、 Eva は DB2 インフォメーション・センターのコピーをイントラネット・サーバー上にインスト

ールして、社内のデータウェアハウスを定期的に利用するすべての社員 (営業担当者、営業部長、および業務分析担当者) から DB2 資料へのアクセスを可能にすることにしました。

Eva は、応答ファイルを使って全社員のコンピューター上に最新バージョンの DB2 Universal Database をインストールするようデータベース・チームに指示します。その際、イントラネット・サーバーのホスト名とポート番号を使って DB2 インフォメーション・センターにアクセスできるよう、確実に各コンピューターを構成します。

しかし、Eva のチームの下級データベース管理者である Migual の誤解によって、数人の社員のコンピューター上で、イントラネット・サーバーの DB2 インフォメーション・センターにアクセスするよう DB2 Universal Database を構成する代わりに、 DB2 インフォメーション・センターのコピーをそれらのコンピューターにインストールしてしまいました。これを訂正するために、Eva は、 **db2set** コマンドを使ってこれらのコンピューター上の DB2 インフォメーション・センターのレジストリー変数 (ホスト名は DB2_DOCHOST、ポート番号は DB2_DOCPORT) を変更するよう Migual に指示しました。これで、ネットワーク上の適切なすべてのコンピューターが DB2 インフォメーション・センターにアクセスできるようになり、社員は DB2 に関する質問の答えを DB2 資料から見つけることができます。

関連概念:

- 692 ページの『DB2 インフォメーション・センター』

関連タスク:

- 703 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 696 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 699 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』

関連資料:

- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)

DB2 製品資料にアクセスする方法として、 IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、 *DB2 インフォメーション・センター CD* から資料をインストールする必要があります。 DB2 セットアップ・ウィザードを使用すれば、インストール設

定を定義し、 UNIX オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、UNIX コンピューターに DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

- **ハードウェア要件**

  以下のいずれかのプロセッサーが必要です。

  - PowerPC (AIX)
  - HP 9000 (HP-UX)
  - Intel 32 ビット (Linux)
  - Solaris UltraSPARC コンピューター (Solaris オペレーティング環境)

- **オペレーティング・システム要件**

  以下のいずれかのオペレーティング・システムが必要です。

  - IBM AIX 5.1 (PowerPC 上)
  - HP-UX 11i (HP 9000 上)
  - Red Hat Linux 8.0 (Intel 32 ビット上)
  - SuSE Linux 8.1 (Intel 32 ビット上)
  - Sun Solaris バージョン 8 (Solaris オペレーティング環境の UltraSPARC コンピューター上)

  注: DB2 インフォメーション・センターは、 DB2 クライアントをサポートする UNIX オペレーティング・システム上で稼動します。このため、IBM Web サイトから DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

- **ソフトウェア要件**
  - 以下のブラウザーがサポートされています。
    - Mozilla バージョン 1.0 以上
- DB2 セットアップ・ウィザードは、グラフィック・インストーラーです。ご使用のマシンで DB2 セットアップ・ウィザードのグラフィカル・ユーザー・インターフェースを表示可能にする X Window システム・ソフトウェアをインプリメントする必要があります。 DB2 セットアップ・ウィザードを実行する前に、ディスプレイを正しくエクスポートしたことを確認してください。たとえば、コマンド・プロンプトで

  ```
  export DISPLAY=9.26.163.144:0.
  ```

  というコマンドを入力します。

- **通信要件**
  - TCP/IP

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. システムにログオンします。
2. DB2 インフォメーション・センター製品 CD を挿入してシステムにマウントします。
3. 次のコマンドを入力して、CD がマウントされているディレクトリーに移動します。

   ```
   cd /cd
   ```

   /cd は、CD のマウント・ポイントを表します。
4. **./db2setup** コマンドを入力して、 DB2 セットアップ・ウィザードを開始します。
5. IBM DB2 セットアップ・ランチパッドが開きます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、**「ヘルプ」**をクリックします。**「キャンセル」**をクリックすれば、いつでもインストールを終了できます。
6. **「インストールしたい製品を選択します」**ページでは、**「次へ」**をクリックします。
7. **「DB2 セットアップ・ウィザードにようこそ (Welcome to the DB2 Setup wizard)」**ページで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
8. インストールを続行するには、使用許諾条件に同意する必要があります。**「ご使用条件」**ページで、**「ご使用条件に同意します (I accept the terms in the license agreement)」** をクリックして、**「次へ」**をクリックします。
9. **「インストール・アクションの選択」**で、**「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」**を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、**「設定を応答ファイルに保管する」**を選択します。**「次へ」**をクリックします。
10. **「インストールする言語の選択」**ページでは、 DB2 インフォメーション・センターをインストールする言語を選択します。**「次へ」**をクリックします。
11. **「DB2 インフォメーション・センター・ポートの指定」**ページでは、 DB2 インフォメーション・センターへの着信通信を構成します。**「次へ」**をクリックしてインストールを続けます。
12. **「ファイルのコピーの開始」**ページでは、インストールの選択項目を確認します。設定を変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

このほか、応答ファイルを使って DB2 インフォメーション・センターをインストールすることもできます。

インストール・ログ db2setup.his、 db2setup.log、および db2setup.err は、デフォルトでは /tmp ディレクトリーに置かれます。

db2setup.log ファイルは、エラーも含めた DB2 製品のインストール情報をすべてキャプチャーします。 db2setup.his ファイルは、コンピューター上の DB2 製品インストール内容をすべて記録します。 DB2 は、db2setup.log ファイルを db2setup.his に付加します。 db2setup.err ファイルは、 Java から戻されるすべてのエラー出力 (例外やトラップの情報など) をキャプチャーします。

インストールが完了したら、ご使用の UNIX オペレーティング・システムに応じて、 DB2 は以下のいずれかのディレクトリーにインストールされます。

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris オペレーティング環境: /opt/IBM/db2/V8.1

関連概念:

- 692 ページの『DB2 インフォメーション・センター』
- 694 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 のインストール (UNIX)』
- 703 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 704 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 702 ページの『DB2 インフォメーション・センターの呼び出し』
- 699 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)

DB2 製品資料にアクセスする方法として、 IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、 *DB2 インフォメーション・センター CD* から DB2 資料をインストールする必要があります。 DB2 セットアップ・ウィザードを使用すれば、インストール設定を定義し、 Windows オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、Windows に DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

- **ハードウェア要件**

  以下のいずれかのプロセッサーが必要です。

  – 32 ビット・コンピューター: Pentium または Pentium 互換の CPU

- **オペレーティング・システム要件**

  以下のいずれかのオペレーティング・システムが必要です。

  – Windows 2000

  – Windows XP

  注: DB2 インフォメーション・センターは、 DB2 クライアントをサポートする Windows オペレーティング・システム上で稼動します。このため、IBM Web サイトの DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

- **ソフトウェア要件**

  – 以下のブラウザーがサポートされています。

    - Mozilla 1.0 以上

    - Internet Explorer バージョン 5.5 または 6.0 (Windows XP の場合はバージョン 6.0)

- **通信要件**

  – TCP/IP

制約事項:

- DB2 インフォメーション・センターをインストールするには、管理権限をもつアカウントが必要です。

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. DB2 インフォメーション・センターのインストールで定義したアカウントで、システムにログオンします。
2. CD をドライブに挿入します。自動実行機能が使用可能になっていれば、IBM DB2 セットアップ・ランチパッドが起動します。
3. DB2 セットアップ・ウィザードは、システム言語を判別して、その言語用のセットアップ・プログラムを立ち上げます。英語以外の言語でセットアップ・プログラムを実行したい場合、またはセットアップ・プログラムの自動始動が失敗した場合には、 DB2 セットアップ・ウィザードを手動で開始できます。

   次のようにして、DB2 セットアップ・ウィザードを手動で開始します。

   a. **「スタート」**をクリックし、**「ファイル名を指定して実行」**を選択します。

   b. **「開く」**フィールドで、以下のコマンドを入力します。

      ```
      x:¥setup.exe /i 2-letter language identifier
      ```

ここで、*x:* は CD ドライブ、 *2-letter language identifier* (2 文字の言語識別子) はセットアップ・プログラムを実行する言語を表します。

c. **「OK」**をクリックします。

4. IBM DB2 セットアップ・ランチパッドが開きます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、**「ヘルプ」**をクリックします。**「キャンセル」**をクリックすれば、いつでもインストールを終了できます。
5. **「インストールしたい製品を選択します」**ページでは、**「次へ」**をクリックします。
6. **「DB2 セットアップ・ウィザードにようこそ (Welcome to the DB2 Setup wizard)」**ページで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
7. インストールを続行するには、使用許諾条件に同意する必要があります。**「ご使用条件」**ページで、**「ご使用条件に同意します (I accept the terms in the license agreement)」** をクリックして、**「次へ」**をクリックします。
8. **「インストール・アクションの選択」**で、**「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」**を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、**「設定を応答ファイルに保管する」**を選択します。**「次へ」**をクリックします。
9. **「インストールする言語の選択」**ページでは、 DB2 インフォメーション・センターをインストールする言語を選択します。**「次へ」**をクリックします。
10. **「DB2 インフォメーション・センター・ポートの指定」**ページでは、 DB2 インフォメーション・センターへの着信通信を構成します。**「次へ」**をクリックしてインストールを続けます。
11. **「ファイルのコピーの開始」**ページでは、インストールの選択項目を確認します。設定を変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

応答ファイルを使って DB2 インフォメーション・センターをインストールすることができます。また、**db2rspgn** コマンドを使って、既存のインストール内容に基づく応答ファイルを生成することもできます。

インストール時に検出されるエラーの詳細については、 「マイ ドキュメント」¥DB2LOG¥ ディレクトリー内の db2.log ファイルと db2wi.log ファイルを参照してください。 「マイ ドキュメント」 ディレクトリーの場所は、ご使用のコンピューターの設定によって異なります。

db2wi.log ファイルは、 DB2 の最新のインストール情報をキャプチャーします。db2.log は、DB2 製品のインストールの履歴をキャプチャーします。

関連概念:

• 692 ページの『DB2 インフォメーション・センター』

- 694 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 製品のインストール (Windows)』
- 703 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 704 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 702 ページの『DB2 インフォメーション・センターの呼び出し』
- 696 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』

関連資料:

- 「コマンド・リファレンス」の『db2rspgn - 応答ファイル生成プログラム・コマンド』

DB2 インフォメーション・センターの呼び出し

DB2 インフォメーション・センターは、Linux、UNIX、および Windows オペレーティング・システム用の DB2 製品 (DB2 Universal Database、 DB2 Connect、DB2 Information Integrator、 DB2 Query Patroller など) を使用するために必要なすべての情報を提供します。

DB2 インフォメーション・センターは、以下の場所から呼び出すことができます。

- DB2 UDB クライアントまたはサーバーがインストールされているコンピューター
- 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イントラネット・サーバーまたはローカル・コンピューター
- IBM の Web サイト

前提条件:

DB2 インフォメーション・センターを呼び出すための要件は、以下のとおりです。

- *オプショナル*: 希望する言語でトピックを表示するようブラウザーを構成する
- *オプショナル*: 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・センターを使用するよう DB2 クライアントを構成する

手順:

DB2 UDB クライアントまたはサーバーがインストールされているコンピューターから DB2 インフォメーション・センターを呼び出すには、以下のようにします。

- (Windows オペレーティング・システムの)「スタート」メニューから: **「スタート」→「プログラム」→「IBM DB2」→「情報」→「インフォメーション・センター」**をクリックします。
- コマンド行プロンプトから:
  - Linux および UNIX オペレーティング・システムの場合、 **db2icdocs** コマンドを発行します。

– Windows オペレーティング・システムの場合、 **db2icdocs.exe** コマンドを発行します。

イントラネット・サーバーまたはローカル・コンピュータ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 Web ブラウザーで開くには、以下のようにします。

- Web ページ http://<host-name>:<port-number>/ を開きます (<host-name> はホスト名、 <port-number> は DB2 インフォメーション・センターを利用可能なポート番号)。

IBM Web サイトにある DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ publib.boulder.ibm.com/infocenter/db2help/ を開きます。

関連概念:

- 692 ページの『DB2 インフォメーション・センター』
- 694 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 704 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 713 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 703 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 714 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』

関連資料:

- 「*コマンド・リファレンス*」の『HELP コマンド』

コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール

http://publib.boulder.ibm.com/infocenter/db2help/ から利用できる DB2 インフォメーション・センターは、資料の新規追加または変更によって定期的に更新されます。さらに、更新された DB2 インフォメーション・センターをコンピューターまたはイントラネット・サーバーにダウンロードしてインストールできる場合もあります。DB2 インフォメーション・センターを更新しても、 DB2 クライアント製品またはサーバー製品は更新されません。

前提条件:

インターネットに接続されたコンピューターへのアクセスが必要です。

手順:

DB2 インフォメーション・センターの更新を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するには、以下のようにします。

1. IBM の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターを開きます。
2. 「DB2 インフォメーション・センターにようこそ」ページの見出し「サービスおよびサポート」の「ダウンロード」セクションで、**「DB2 資料」**リンクをクリックします。
3. 最新のドキュメンテーション・イメージのレベルと、インストール済みのドキュメンテーション・レベルを比較して、 DB2 インフォメーション・センターを更新する必要があるかどうかを確認します。「DB2 インフォメーション・センターにようこそ」ページに、インストール済みのドキュメンテーションのレベルがリストされます。
4. より新しいバージョンの DB2 インフォメーション・センターが存在する場合、ご使用のオペレーティング・システムに対応する最新の *DB2 インフォメーション・センター・イメージをダウンロードします。*
5. 最新の *DB2 インフォメーション・センター・イメージをインストールするには、* Web ページの指示に従ってください。

関連概念:

- 694 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 702 ページの『DB2 インフォメーション・センターの呼び出し』
- 696 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 699 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 インフォメーション・センターにおける特定の言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、 DB2 インフォメーション・センターでは英語でトピックが表示されます。

手順:

Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。

1. Internet Explorer の**「ツール」**—> **「インターネット オプション」**—> **「言語...」**ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
   - リストに新しい言語を追加するには、**「追加...」**ボタンをクリックします。

     注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。
   - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで**「上へ」**ボタンをクリックします。

3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

Mozilla Web ブラウザーの場合に、使いたい言語でトピックを表示するには、以下のようにします。

1. Mozilla の**「編集」**—> **「設定」**—> **「言語」**ボタンをクリックします。「設定」ウィンドウに「言語」パネルが表示されます。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
   - リストに新しい言語を追加するには、**「追加...」**ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
   - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで**「上に移動」**ボタンをクリックします。
3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

関連概念:

- 692 ページの『DB2 インフォメーション・センター』

DB2 PDF 資料および印刷された資料

以下の表は、正式な資料名、資料番号、および PDF ファイル名を示しています。ハードコピー版の資料を注文するには、正式な資料名を知っておく必要があります。 PDF ファイルを印刷するには、PDF ファイル名を知っておく必要があります。

DB2 資料は、以下のカテゴリーに分類されています。

- DB2 中核情報
- 管理情報
- アプリケーション開発情報
- ビジネス・インテリジェンス情報
- DB2 Connect 情報
- 入門情報
- チュートリアル情報
- オプショナル・コンポーネント情報
- リリース・ノート

以下の表は、DB2 ライブラリー内の各資料について、その資料のハードコピー版を注文したり、PDF 版を印刷または表示したりするのに必要な情報を示しています。DB2 ライブラリー内の各資料に関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。

DB2 の基本情報

こうした資料の情報は、すべての DB2 ユーザーに基本的なもので、プログラマーおよびデータベース管理者にとって役立つ情報であるとともに、DB2 Connect、DB2 Warehouse Manager、または他の DB2 製品を使用するユーザーにとっても役

立つ内容です。

表 73. DB2 の基本情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database コマンド・リファレンス*」	SC88-9140	db2n0j81
「*IBM DB2 Universal Database 用語集*」	資料番号なし	db2t0j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 1 巻*」	GC88-9152 (ハードコピーなし)	db2m1j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 2 巻*」	GC88-9153 (ハードコピーなし)	db2m2j81
「*IBM DB2 Universal Database 新機能*」	SC88-9158	db2q0j81

管理情報

これらの資料の情報は、DB2 データベース、データウェアハウス、およびフェデレーテッド・システムを効果的に設計し、インプリメントし、保守するために必要なトピックを扱っています。

表 74. 管理情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database 管理ガイド: プランニング*」	SC88-9135	db2d1j81
「*IBM DB2 Universal Database 管理ガイド: インプリメンテーション*」	SC88-9133	db2d2j81
「*IBM DB2 Universal Database 管理ガイド: パフォーマンス*」	SC88-9134	db2d3j81
「*IBM DB2 Universal Database 管理 API リファレンス*」	SC88-9136	db2b0j81
「*IBM DB2 Universal Database 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*」	SC88-9142	db2dmj81
「*IBM DB2 Universal Database 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*」	SC88-9143	db2haj81
「*IBM DB2 Universal Database データウェアハウス・センター 管理ガイド*」	SC88-9165	db2ddj81
「*IBM DB2 Universal Database SQL リファレンス 第 1 巻*」	SC88-9155	db2s1j81
「*IBM DB2 Universal Database SQL リファレンス 第 2 巻*」	SC88-9156	db2s2j81

表 74. 管理情報 (続き)

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database システム・モニター ガイドおよびリファレンス*」	SC88-9157	db2f0j81

アプリケーション開発情報

これらの資料の情報は、 DB2 Universal Database (DB2 UDB) のアプリケーション開発者またはプログラマーが特に関心を持つ内容です。サポートされるさまざまなプログラミング・インターフェース (組み込み SQL、ODBC、JDBC、SQLJ、CLI など) を使用して DB2 UDB にアクセスするのに必要な資料とともに、サポートされる言語およびコンパイラーについても紹介されています。また、DB2 インフォメーション・センターをご使用の場合には、サンプル・プログラムのソース・コードの HTML バージョンにアクセスすることもできます。

表 75. アプリケーション開発情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database アプリケーション開発ガイド アプリケーションの構築および実行*」	SC88-9137	db2axj81
「*IBM DB2 Universal Database アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」	SC88-9138	db2a1j81
「*IBM DB2 Universal Database アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」	SC88-9139	db2a2j81
「*IBM DB2 Universal Database 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*」	SC88-9159	db2l1j81
「*IBM DB2 Universal Database 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*」	SC88-9160	db2l2j81
「*IBM DB2 Universal Database データウェアハウス・センター アプリケーション統合ガイド*」	SC88-9166	db2adj81
「*IBM DB2 Universal Database XML Extender 管理およびプログラミングのガイド*」	SC88-9172	db2sxj81

ビジネス・インテリジェンス情報

これらの資料の情報は、さまざまなコンポーネントを使用して、 DB2 Universal Database のデータウェアハウジング機能および分析機能を拡張する方法を説明しています。

表 76. ビジネス・インテリジェンス情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Warehouse Manager Standard Edition インフォメーション・カタログ・センター 管理ガイド*」	SC88-9167	db2dij81
「*IBM DB2 Warehouse Manager Standard Edition インストール・ガイド*」	GC88-9164	db2idj81
「*IBM DB2 Warehouse Manager Standard Edition DB2 Warehouse Manager を使用時の ETI ソリューション・コンバージョン・プログラムの管理*」	SC88-9894	iwhe1mstx80

DB2 Connect 情報

このカテゴリーの情報は、 DB2 Connect Enterprise Edition または DB2 Connect Personal Edition を使用して、メインフレーム・サーバーおよびミッドレンジ・サーバー上のデータにアクセスする方法を説明しています。

表 77. DB2 Connect 情報

資料名	資料番号	PDF ファイル名
「*IBM コネクティビティー 補足*」	資料番号なし	db2h1j81
「*IBM DB2 Connect Enterprise Edition 概説およびインストール*」	GC88-9145	db2c6j81
「*IBM DB2 Connect Personal Edition 概説およびインストール*」	GC88-9146	db2c1j81
「*IBM DB2 Connect ユーザーズ・ガイド*」	SC88-9147	db2c0j81

入門情報

このカテゴリーの情報は、サーバー、クライアント、および他の DB2 製品をインストールして構成する場合に役立ちます。

表 78. 入門情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database DB2 クライアント機能 概説およびインストール*」	GC88-9144 (ハードコピーなし)	db2itj81
「*IBM DB2 Universal Database DB2 サーバー機能 概説およびインストール*」	GC88-9148	db2isj81
「*IBM DB2 Universal Database DB2 Personal Edition 概説およびインストール*」	GC88-9150	db2i1j81
「*IBM DB2 Universal Database インストールおよび構成 補足*」	GC88-9149 (ハードコピーなし)	db2iyj81
「*IBM DB2 Universal Database DB2 Data Links Manager 概説およびインストール*」	GC88-9141	db2z6j81

チュートリアル情報

チュートリアル情報は、DB2 機能を紹介し、さまざまなタスクを実行する方法を示します。

表 79. チュートリアル情報

資料名	資料番号	PDF ファイル名
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介*」	資料番号なし	db2tuj81
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド*」	資料番号なし	db2taj81
「*インフォメーション・カタログ・センター チュートリアル*」	資料番号なし	db2aij81
「*Video Central for e-business チュートリアル*」	資料番号なし	db2twj81
「*Visual Explain チュートリアル*」	資料番号なし	db2tvj81

オプショナル・コンポーネント情報

このカテゴリーの情報は、 DB2 のオプショナル・コンポーネントを使用する方法について説明しています。

表 80. オプショナル・コンポーネント情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Cube Views Guide and Reference*」	SC18–7298	db2aax81
「*IBM DB2 Query Patroller インストール、管理、使用法のガイド*」	GC88–9154	db2dwj81
「*IBM DB2 Spatial Extender and Geodetic Extender ユーザーズ・ガイドおよびリファレンス*」	SC88-9171	db2sbj81
「*IBM DB2 Universal Database Data Links Manager 管理ガイドおよびリファレンス*」	SC88-9169	db2z0x82
「*DB2 Net Search Extender 管理およびユーザーズ・ガイド*」 **注:** この資料の HTML 版は、HTML ドキュメンテーション CD からインストールされません。	SH88-8546	N/A

リリース・ノート

リリース・ノートは、ご使用の製品のリリースおよびフィックスパック・レベルに特有の追加情報を紹介します。また、リリース・ノートには、各リリース、アップデート、およびフィックスパックで組み込まれた資料上の更新の要約も含まれています。

表 81. リリース・ノート

資料名	資料番号	PDF ファイル名
「*DB2 リリース・ノート*」	「注」を参照。	「注」を参照。
「*DB2 インストール情報*」	製品 CD-ROM でのみ参照可能。	使用できません。

注: リリース・ノートは以下の形式で入手できます。

- XHTML およびテキスト形式 (製品 CD 内)
- PDF 形式 (PDF ドキュメンテーション CD 内)

さらに、リリース・ノートの中で、『*既知の問題と予備手段*』および『*リリース間の非互換性*』に関する部分は DB2 インフォメーション・センターにも表示されます。

UNIX ベースのプラットフォームでテキスト形式でリリース・ノートを確認するには、 `Release.Notes` ファイルを参照してください。このファイルは、`DB2DIR/Readme/%L` ディレクトリーに収録されています。 `%L` はロケール名を表しています。 `DB2DIR` は以下になります。

- AIX オペレーティング・システムの場合: /usr/opt/db2_08_01
- その他のすべての UNIX ベースのオペレーティング・システムの場合: /opt/IBM/db2/V8.1

関連概念:

- 691 ページの『DB2 資料とヘルプ』

関連タスク:

- 711 ページの『PDF ファイルからの DB2 資料の印刷方法』
- 712 ページの『DB2 の印刷資料の注文方法』
- 713 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』

PDF ファイルからの DB2 資料の印刷方法

DB2 PDF ドキュメンテーション CD に収録されている DB2 資料を印刷することができます。 Adobe Acrobat Reader を使用すれば、資料全体または特定のページを印刷できます。

前提条件:

Adobe Acrobat Reader がインストールされていることを確認してください。 Adobe Acrobat Reader をインストールする必要がある場合、 Adobe Web サイト (www.adobe.com) から入手できます。

手順:

PDF ファイルから DB2 資料を印刷するには以下のようにします。

1. *DB2 PDF* ドキュメンテーション CD をドライブに挿入します。 UNIX オペレーティング・システムの場合、 DB2 PDF ドキュメンテーション CD をマウントします。 UNIX オペレーティング・システムで CD をマウントする方法については、「*概説およびインストール*」を参照してください。
2. index.htm を開きます。ブラウザー・ウィンドウにファイルが開きます。
3. 参照したい PDF のタイトルをクリックします。 Acrobat Reader で PDF が開きます。
4. **「ファイル」 → 「印刷」**を選択して、所要の資料の任意の部分を印刷します。

関連概念:

- 692 ページの『DB2 インフォメーション・センター』

関連タスク:

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (AIX)』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『HP-UX 上での CD-ROM のマウント』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Linux)』
- 712 ページの『DB2 の印刷資料の注文方法』

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Solaris)』

関連資料:

- 705 ページの『DB2 PDF 資料および印刷された資料』

DB2 の印刷資料の注文方法

ハードコピー版の資料を望む場合には、以下のいずれかの方法で注文できます。

印刷資料の注文方法:

一部の国または地域では、印刷された資料を注文することもできます。お客様がお住まいの国または地域でこのサービスが利用可能かどうかを確認するには、お住まいの国または地域の IBM Publications Web サイトをご覧ください。資料のご注文が可能な場合、以下のようにすることができます。

- 正規の IBM 製品販売業者または営業担当員に連絡してください。お客様がお住まいの地域の IBM 担当員の情報については、お手数ですが IBM の Web サイト (www.ibm.com/planetwide) の IBM Worldwide Directory of Contacts で確認してください。
- IBM Publications Center (http://www.ibm.com/shop/publications/order) にアクセスしてください。なお、IBM Publications Center から資料を注文できない国もあります。

DB2 製品がご利用可能になった時点で、印刷された資料は *DB2 PDF* ドキュメンテーション *CD* にある PDF 形式の資料と同じものです。さらに、 *DB2* インフォメーション・センター *CD* に収録されている印刷された資料の内容もまた、これらと同じです。ただし、DB2 インフォメーション・センター CD には、 PDF 資料にない追加情報も含まれます (たとえば、SQL 管理作業や HTML サンプル)。 DB2 PDF ドキュメンテーション CD に収録されている資料の中には、ハードコピーとしてご注文できない資料もあります。

注: DB2 インフォメーション・センターは、 PDF またはハードコピー の資料よりも頻繁に更新されます。ドキュメンテーションの更新が入手可能になった時点でインストールするか、 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) を参照して最新の情報を入手してください。

関連タスク:

- 711 ページの『PDF ファイルからの DB2 資料の印刷方法』

関連資料:

- 705 ページの『DB2 PDF 資料および印刷された資料』

DB2 ツールからコンテキスト・ヘルプを呼び出す

コンテキスト・ヘルプは、特定のウィンドウ、ノートブック、ウィザード、またはアドバイザーに関連したタスクまたはコントロールの情報を提供します。コンテキスト・ヘルプは、グラフィカル・ユーザー・インターフェースのある DB2 管理ツールおよび開発ツールから利用できます。コンテキスト・ヘルプには、以下の 2 種類があります。

- それぞれのウィンドウまたはノートブックにある**「ヘルプ」**ボタンからアクセス可能なヘルプ
- infopop (ポップアップ情報ウィンドウ)。これは、マウス・カーソルを特定のフィールドまたはコントロール上に置いたとき、またはウィンドウ、ノートブック、ウィザード、アドバイザー内でフィールドまたはコントロールを選択して F1 を押すと表示されます。

「ヘルプ」ボタンを押すと、概説、前提条件、およびタスク情報が表示されます。infopop は、それぞれのフィールドおよびコントロールについて説明します。

手順:

コンテキスト・ヘルプを呼び出すには、以下のようにします。

- ウィンドウおよびノートブックのヘルプを表示するには、いずれかの DB2 ツールを開始して、任意のウィンドウまたはノートブックを開きます。ウィンドウまたはノートブックの右下隅にある**「ヘルプ」**ボタンをクリックして、コンテキスト・ヘルプを呼び出します。

  また、それぞれの DB2 ツール・センターの上部にある**「ヘルプ」**メニュー項目からコンテキスト・ヘルプにアクセスすることもできます。

  ウィザードおよびアドバイザーでは、最初のページの「タスクの概要」リンクをクリックすると、コンテキスト・ヘルプを表示できます。
- ウィンドウまたはノートブック上の各コントロールの infopop ヘルプを表示するには、コントロールをクリックしてから、**F1** を押します。コントロールの詳細情報を示すポップアップ情報が、黄色いウィンドウに表示されます。

  注: フィールドまたはコントロールにマウス・カーソルを置いておくだけで infopops が表示されるようにするには、「ツール設定」ノートブックの**「文書 (Documentation)」**ページの**「infopops の自動表示」**チェック・ボックスを選択します。

  infopop に似た別のコンテキスト・ヘルプに、診断ポップアップ情報があります。これにはデータ入力規則が示されます。診断ポップアップ情報は、無効または不十分なデータが入力されたとき、紫色のウィンドウに表示されます。診断ポップアップ情報は、以下に関して表示されます。
  - 必須フィールド。
  - 日付フィールドのように、正確なフォーマットを必要とするデータのフィールド。

関連タスク:

- 702 ページの『DB2 インフォメーション・センターの呼び出し』
- 714 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』

- 714 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 715 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』
- 『DB2 インフォメーション・センターへのアクセス: Concepts help』
- 『DB2 UDB ヘルプの使用法: Common GUI help』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』
- 『DB2 コンテキスト・ヘルプと資料へのアクセスを設定する: Common GUI help』

コマンド行プロセッサーからメッセージ・ヘルプを呼び出す

メッセージ・ヘルプは、メッセージが出された原因と、エラーへの応答として実行すべきアクションを説明します。

手順:

メッセージ・ヘルプを呼び出すには、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。

```
? XXXnnnnn
```

ここで、*XXXnnnnn* は有効なメッセージ ID を表します。

たとえば、`? SQL30081` と入力すると、メッセージ SQL30081 に関するヘルプを表示します。

関連概念:

- 「*メッセージ・リファレンス 第 1 巻*」の『メッセージの概要』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーからコマンド・ヘルプを呼び出す

コマンド・ヘルプは、コマンド行プロセッサーでのコマンドの構文を説明します。

手順:

コマンド・ヘルプを呼び出すには、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。

```
? command
```

ここで *command* はキーワードまたはコマンド全体を表します。

たとえば、`? catalog` と入力すると、すべての CATALOG コマンドに関するヘルプが表示され、 `? catalog database` と入力すると、 CATALOG DATABASE コマンドのヘルプだけが表示されます。

関連タスク:

- 713 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 702 ページの『DB2 インフォメーション・センターの呼び出し』
- 714 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 715 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーから SQL 状態ヘルプを呼び出す

DB2 Univerrsal Database は、 SQL ステートメントの結果の原因となったと考えられる条件の 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 状態ヘルプを呼び出すには、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。

たとえば、`? 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。

関連タスク:

- 702 ページの『DB2 インフォメーション・センターの呼び出し』
- 714 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 714 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』

DB2 チュートリアル

DB2® チュートリアルは、 DB2 Universal Database のさまざまな機能について学習するのを支援します。このチュートリアルでは、アプリケーションの開発、SQL 照会のパフォーマンス調整、データウェアハウスの処理、メタデータの管理、および DB2 を使用した Web サービスの開発の各分野で、段階的なレッスンが用意されています。

はじめに:

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

チュートリアルの中で、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、それぞれのチュートリアルを参照してください。

DB2 Universal Database チュートリアル:

以下に示すチュートリアルのタイトルをクリックすると、そのチュートリアルを表示できます。

ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介
データウェアハウス・センターを使用して簡単なデータウェアハウジング・タスクを実行します。

ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド
データウェアハウス・センターを使用して高度なデータウェアハウジング・タスクを実行します。

インフォメーション・カタログ・センター・チュートリアル
インフォメーション・カタログを作成および管理して、インフォメーション・カタログ・センターを使用してメタデータを配置し使用します。

Visual Explain チュートリアル
Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 トラブルシューティング情報

DB2® 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 ドキュメンテーション
トラブルシューティング情報は、DB2 インフォメーション・センター、および DB2 ライブラリーに含まれる PDF 資料の中でご利用いただけます。DB2 インフォメーション・センターで、 (ブラウザー・ウィンドウの左側の) ナビゲーション・ツリーの「サポートおよびトラブルシューティング (Support and troubleshooting)」ブランチを参照すると、 DB2 トラブルシューティング・ドキュメンテーションの詳細なリストが見つかります。

DB2 Technical Support の Web サイト
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検索したい場合は、 DB2 Technical Support の Web サイトを参照してください。Technical Support サイトには、最新の DB2 出版物、TechNotes、プログラム診断依頼書 (APAR)、フィックスパック、DB2 内部エラー・コードの最新リスト、その他のリソース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。

DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/udb/winos2unix/support) にアクセスしてください。

DB2 Problem Determination Tutorial Series
DB2 製品で作業中に直面するかもしれない問題を素早く識別し、解決する方法に関する情報を見つけるには、 DB2 Problem Determination Tutorial Series の Web サイトを参照してください。あるチュートリアルでは、使用可能な DB2 問題判別機能およびツールを紹介し、それらをいつ使用すべきかを判断する助けを与えます。別のチュートリアルは、『データベース・エ

ンジン問題判別 (Database Engine Problem Determination)』、『パフォーマンス問題判別 (Performance Problem Determination)』、『アプリケーション問題判別 (Application Problem Determination)』などの関連トピックを扱っています。

DB2 Technical Support (http://www.ibm.com/software/data/support/pdm/db2tutorials.html) には、 DB2 問題判別チュートリアルがすべて揃っています。

関連概念:

- 692 ページの『DB2 インフォメーション・センター』
- 「*問題判別の手引き*」の『Introduction to Problem Determination - DB2 テクニカル・サポートのチュートリアル』

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 ユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 718 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 718 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 718 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

アクセスしやすい表示

DB2 ツールには、視力の弱いユーザー、その他の視力障害をもつユーザーのためにアクセシビリティーを向上させる機能が備わっています。これらのアクセシビリティー拡張機能には、フォント・プロパティーのカスタマイズを可能にする機能も含まれています。

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、メニューおよびテキストのフォントを変更する: Common GUI help を参照してください。

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

DB2 形式は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。 XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン・ドキュメンテーションにアクセスする場合にのみ使用できます。

関連概念:

- 719 ページの『ドット 10 進シンタックス・ダイアグラム』

関連タスク:

- 『キーボード・ショートカットおよびアクセラレーター: Common GUI help』
- 『メニューおよびテキストのフォントを変更する: Common GUI help』

ドット 10 進シンタックス・ダイアグラム

スクリーン・リーダーを使用してインフォメーション・センターを利用するユーザーのために、シンタックス・ダイアグラムがドット 10 進形式で提供されます。

ドット 10 進形式では、各シンタックス・エレメントは別々の行に書き込まれます。複数のシンタックス・エレメントが常に同時に存在する (または常に同時に不在の) 場合、単一のコンパウンド・シンタックス・エレメントとみなせるので同一行に表示できます。

各行は、ドット 10 進数で開始します。たとえば、3 または 3.1 ないしは 3.1.1 です。こうした数を適切に聞き取るには、スクリーン・リーダーが句読点を読み取るように設定されていることを確認してください。同じドット 10 進数を持つすべてのシンタックス・エレメント (たとえば、 3.1 という数値を持つすべてのシンタックス・エレメント) は、相互に排他的な代替エレメントです。 3.1 USERID および 3.1 SYSTEMID という行を聞き取る場合、シンタックスには両方ではなく USERID または SYSTEMID のどちらかが含まれることが分かります。

ドット 10 進レベルは、ネストのレベルを表示します。たとえば、ドット 10 進数 3 のシンタックス・エレメントの後に、一連のドット 10 進数 3.1 のシンタックス・エレメントが続きます。 3.1 の番号が付されたシンタックス・エレメントすべては、番号 3 の付されたシンタックス・エレメントに従属します。

シンタックス・エレメントに関する情報を追加するため、ドット 10 進数の次に特定のワードおよびシンボルが使用されます。 時折、こうしたワードおよびシンボルはエレメントの最初に表示される場合もあります。 簡単に識別するため、ワードやシンボルがシンタックス・エレメントの一部である場合には、円記号 (¥) 文字が先頭に付きます。 * シンボルはドット 10 進数の次に使用でき、シンタックス・エレメントが反復することを示します。 たとえば、ドット 10 進数 3 のシンタックス・エレメント *FILE は、3 ¥* FILE という形式になります。 3* FILE という形式は、シンタックス・エレメント FILE が反復されることを示します。 3* ¥* FILE という形式は、シンタックス・エレメント * FILE が反復されることを示します。

シンタックス・エレメントのストリングを分離するのに使用されるコンマなどの文字は、シンタックス内の分離する項目の直前に表示されます。 こうした文字は、それぞれの項目と同一行に表示するか、同じドット 10 進数を持つ関連する項目のある別の行に表示できます。 またその行には、シンタックス・エレメントに関する情報を提供する別のシンボルを表示することも可能です。 たとえば、複数の LASTRUN および DELETE シンタックス・エレメントを使用している場合には、5.1*、5.1 LASTRUN、および 5.1 DELETE という行は、エレメントをコンマで区切る必要があります。区切り文字が指定されないと、各シンタックス・エレメントを区切るのにブランクが使用されると想定されます。

シンタックス・エレメントの前に % シンボルが付く場合、他の箇所で定義されている参照であることを示します。 % シンボルの後のストリングは、リテラルではなくシンタックス・フラグメントの名前です。 たとえば、2.1 %OP1 という行は別のシンタックス・フラグメント OP1 を参照すべきことを意味します。

以下のワードおよびシンボルが、ドット 10 進数の次に使用されます。

- ? は、オプションのシンタックス・エレメントであることを表します。? シンボルが後に続くドット 10 進数は、対応するドット 10 進数のシンタックス・エレメント、および任意の従属のシンタックス・エレメントがオプションであることを示します。ドット 10 進数の付いたシンタックス・エレメントが 1 つしかない場合、 ? シンボルはそのシンタックス・エレメントと同じ行に表示されます (たとえば、5? NOTIFY)。ドット 10 進数の付いたシンタックス・エレメントが複数ある場合、 ? シンボルだけで行に表示され、その後にオプションのシンタックス・エレメントが続きます。たとえば、「5 ?、5 NOTIFY、および 5 UPDATE」という行を聞き取る場合、シンタックス・エレメント NOTIFY および UPDATE がオプションである、つまりそのいずれかを選択でき、どちらも選択しないこともできることが分かります。 ? シンボルは、線路型ダイアグラムのバイパス線に相当します。
- ! は、デフォルトのシンタックス・エレメントであることを表します。! シンボルおよびシンタックス・エレメントが後に続くドット 10 進数は、そのシンタックス・エレメントが、同じドット 10 進数を共用するシンタックス・エレメントすべてのデフォルト・オプションであることを示します。同じドット 10 進数を共用するシンタックス・エレメントのうち 1 つだけに、 ! シンボルを指定できます。たとえば、「2? FILE、2.1! (KEEP)、および 2.1 (DELETE)」という行を聞き取る場合、 FILE キーワードのデフォルト・オプションは (KEEP) になります。この例では、FILE キーワードを含めてもオプションを指定しない場合には、デフォルト・オプション KEEP が適用されます。デフォルト・オプションは、次に高位のドット 10 進数にも適用されます。 この例の場合、FILE キーワードが省略されると、デフォルトの FILE(KEEP) が使用されます。しかし、「2? FILE、2.1、2.1.1! (KEEP)、および 2.1.1 (DELETE)」という行を聞き取る場合、デフォルト・オプション KEEP は次に高位のドット 10 進数 2.1 (関連キーワードを持っていない) にのみ適用され、 2? FILE には適用されません。キーワード FILE が省略されると、どれも使用されません。
- * は、0 回以上反復できるシンタックス・エレメントを示します。 * シンボルが後に続くドット 10 進数は、このシンタックス・エレメントが 0 回以上使用できること、つまりオプションであり、なおかつ反復できることを表します。たとえば、5.1* データ域という行を聞き取る場合、 1 つまたは複数のデータ域を含めるか、またはデータ域を全く含めないことが可能です。「3*、3 HOST、および 3 STATE」という行を聞き取る場合、 HOST、STATE をどちらか一方または両方同時に含めるか、どちらも含めないことができます。

  注:

  1. ドット 10 進数の後にアスタリスク (*) が付き、ドット 10 進数の付いた項目が 1 つしかない場合には、同じ項目を複数回反復できます。
  2. ドット 10 進数の後にアスタリスクが付き、ドット 10 進数の付いた項目が複数ある場合、リストから複数の項目を使用できますが、各項目を複数回使用することはできません。 前述の例では、HOST STATE と書くことはできますが、HOST HOST とは書けません。
  3. * シンボルは、線路型シンタックス・ダイアグラムのループバック線に相当します。
- + は、1 回以上含める必要のあるシンタックス・エレメントであることを示します。 + シンボルが後に続くドット 10 進数は、このシンタックス・エレメントを 1 回以上含める必要があること、つまり少なくとも 1 回は含める必要があり、反

復できることを表します。たとえば、「6.1+ データ域」という行を聞き取る場合、データ域を少なくとも 1 回は含めなければなりません。「2+、2 HOST、および 2 STATE」という行を聞き取る場合には、 HOST、STATE、またはその両方を含める必要があります。 * シンボルと同様に、+ シンボルは、ドット 10 進数の付いた項目が 1 つしかない場合に限り、その特定の項目のみを反復できます。 * シンボルと同様、 + シンボルは線路型シンタックス・ダイアグラムのループバック線に相当します。

関連概念:

- 717 ページの『アクセス支援』

関連タスク:

- 『キーボード・ショートカットおよびアクセラレーター: Common GUI help』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『構文図の見方』

DB2 Universal Database 製品の共通基準認証

DB2 Universal Database は、 Common Criteria の評価検定レベル 4 (EAL4) で認証の評価を受けています。 Common Criteria の詳細については、以下の Common Criteria の Web サイトを参照してください。 http://niap.nist.gov/cc-scheme/

付録 G. 特記事項

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。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

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
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
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。

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。

商標

以下は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

以下は、それぞれ各社の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。

Pentium は、Intel Corporation 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、Sun Microsystems, Inc. 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。

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]

アクセシビリティー
  機能 717
  ドット 10 進シンタックス・ダイアグラム 719
アクセス・プラン
  索引の使用 26, 166
  複数述部の列相関のため 163
  方式 166
  ロックへの影響 78
アクティブ・アプリケーションの最大数構成パラメーター 442
アドバイザー
  設計アドバイザー 227
アプリケーション
  共用メモリーの使用 243
  制御ヒープの設定 398
  ノードでのコーディネーター・エージェントの最大数 439
アプリケーション単位の最大データベース・ファイル・オープン数構成パラメーター 444
アプリケーション・グループ・メモリー設定の最大サイズ構成パラメーター 399
アプリケーション・グローバル・メモリー、構成パラメーター 243
アプリケーション・コントロール・ヒープ・サイズ構成パラメーター 398
アプリケーション・サポート層ヒープ・サイズ構成パラメーター 413
アプリケーション・パフォーマンスのモデル化
  カタログ統計を使用 141
  手動で調整されたカタログ統計を使用 139
アプリケーション・プログラム 43
アプリケーション・プロセス
  ロックへの影響 77
一時表
  情報の使用、db2expln 658
イベント・スナップショット 299
印刷
  PDF ファイル 711
印刷された資料の注文 712
インスタンス・メモリー構成パラメーター 420
インストール
  インフォメーション・センター 694, 696, 699
インフォメーション・センター
  インストール 694, 696, 699
ウィザード
  設計アドバイザー 227
エージェント
  数に影響を与える構成パラメーター 294
  管理 294
  クライアント接続 295
  作業エージェント・タイプ 291
  説明 291
  パーティション・データベースでの 298
  メモリー使用 243
エージェントの最大数構成パラメーター 441
エージェント・プール・サイズ構成パラメーター 447
エージェント・プロセス
  エージェントの最大数 441
  エージェントの優先順位構成パラメーター 436
  同時エージェントの最大数 443
  applheapsz 構成パラメーター 404
  aslheapsz 構成パラメーター 413
エクステント
  DMS 表スペースでのエクステント・マップ・ページ (EMP) 17
  SMS 表スペースの 16
エスカレーション前のロック・リストの最大パーセント構成パラメーター 426
エラー・メッセージ
  パーティション・データベースへのノード追加 329
エンジン・ディスパッチ可能単位 (EDU)
  エージェント 291
  説明 37
オーバーフロー・レコード
  パフォーマンスの影響 272
  標準表における 20
オーバーヘッド
  削減のための行ブロッキング 90
オプティマイザー
  アクセス・プラン
    索引アクセス方式 169
    索引の使用 166
オプティマイザー *(続き)*
  アクセス・プラン *(続き)*
    ソートとグループ化の影響 191
    列相関のため 163
  結合
    最良のそれへの戦略 179
    説明 175
    パーティション・データベースで 185
  照会書き直し方法 156
  分散統計の使用 130
オンライン
  ヘルプへのアクセス 713

[カ行]

拡張記憶
  説明 37
カタログ統計
  いつ収集するか 109
  カタログ表の説明 121
  索引クラスター率 172
  収集
    更新 111
    索引統計 115
    特定の列の分散統計 113
    要件とメソッドの説明 112
  収集される詳細索引データ 135
  収集される情報 126
  手動更新のガイドライン 143
  手動更新の規則
    索引統計 147
    表とニックネーム 146
    分散 145
    列統計 144
  使用方法 109
  分散統計
    いつ収集するか 127
    拡張例の使用 131
    頻度 127
    分位 127
  モデル化の手動調整 139
  ユーザー定義関数の 138
  列のサブエレメントの 136
  を使用した実動データベースのモデル化 141
カタログ表
  説明 121
カタログ・キャッシュ・サイズ構成パラメーター 386
カタログ・ノード 43

ガバナー
開始と停止 304
規則エレメント 310
構成 306
構成ファイル規則の説明 307
構成ファイルの例 315
作成するログ・ファイル 316
説明 303
デーモンの説明 305
ログ・ファイルへの照会 320
環境変数
概要 571
キーボード・ショートカット
サポート 717
キャパシティー
拡張方法 321
行
ロック・タイプ 53
行のブロッキング
指定 90
クライアント入出力ブロック・サイズ構成パラメーター 416
クライアント・サポート
クライアント入出力ブロック・サイズ構成パラメーター 416
TCP/IP サービス名構成パラメーター 513
tpname 構成パラメーター 514
クラスタリング、索引の 20
グループ化の影響、アクセス・プランへの 191
グループへのコミット数構成パラメーター 469
結合
オプティマイザーによる副照会変換 158
外部表のブロードキャスト 185
共用集約 158
最良のそれへのオプティマイザーの戦略 179
冗長の除去 158
説明 175
タイプ
外部表の指示 185
内部表の指示 185
内部表のブロードキャスト 185
ネスト・ループ、説明 176
パーティション・データベースで 185
パーティション・データベースでの表キュー戦略 183
ハッシュ、説明 176
表示された db2expln 情報 660
併置 185
方式、リスト 176
マージ、説明 176
権限
グループ名の定義
システム管理権限グループ名構成パラメーター 551
システム制御権限グループ名構成パラメーター 552
システム保守権限グループ名構成パラメーター 553
検索引き数述部
定義された 173
コーディネーター・エージェント
接続コンセントレーターの使用 295
説明 31
コーディネーター・エージェントの最大数構成パラメーター 439
コーディネート世界時 523
コード・ページ
データベース構成パラメーター 494
更新
HMTL 文書 703
構成
調整
パラメーター 363
データベース・パラメーターの変更 364
動的 367
パラメーターのサマリー、データベースの 370
パラメーターのサマリー、データベース・マネージャーの 370
構成パラメーター
数に影響を与える 294
自動 363
照会最適化に影響を与える 153
説明 361
認証 542
認証 (DAS) 558
分離 31, 449
agentpri 436
agent_stack_sz 402
alt_collate 493
appgroup_mem_sz 399
applheapsz 404
app_ctl_heap_sz 398
archretrydelay 463
aslheapsz 413
audit_buf_sz 417
autonomic_switches 510
autorestart 475
avg_appls 438
backup_pending 500
blk_log_dsk_ful 464
catalogcache_sz 386
catalog_noauth 543
chngpgs_thresh 428
clnt_krb_plugin 544
構成パラメーター *(続き)*
clnt_pw_plugin 545
codepage 494
codeset 494
collate_info 494
comm_bandwidth 533
conn_elapse 517
contact_host 559
cpuspeed 534
dasadm_group 561
das_codepage 560
das_territory 560
database_consistent 501
database_level 495
database_memory 388
datalinks 496
db2system 561
dbheap 389
dftdbpath 545
dft_account_str 535
dft_degree 503
dft_extent_sz 429
dft_loadrec_ses 476
dft_monswitches 531
dft_mttb_types 504
dft_prefetch_sz 430
dft_queryopt 504
dft_refresh_age 505
dft_sqlmathwarn 505
diaglevel 527
diagpath 528
dir_cache 418
discover 515
discover (DAS) 562
discover_db 516
discover_inst 516
dlchktime 424
dl_expint 497
dl_num_copies 497
dl_time_drop 498
dl_token 498
dl_wt_iexpint 499
dyn_query_mgmt 492
estore_seg_sz 37, 431
exec_exp_task 563
failarchpath 464
fcm_num_anchors 518
fcm_num_buffers 519
fcm_num_connect 520
fcm_num_rqb 521
federated 536
fed_noauth 547
fenced_pool 448
groupheap_ratio 401
group_plugin 547
hadr_db_role 476

構成パラメーター *(続き)*
hadr_local_host 477
hadr_local_svc 478
hadr_remote_host 478
hadr_remote_inst 479
hadr_remote_svc 479
hadr_syncmode 480
hadr_timeout 481
health_mon 529
indexrec 481
instance_memory 420
intra_parallel 525
java_heap_sz 421
jdk_64_path 563
jdk_path 536
jdk_path (DAS) 564
local_gssplugin 548
locklist 390
locktimeout 425
logarchmeth1 465
logarchmeth2 466
logarchopt1 466
ノード間リカバリーの例 685
logarchopt2 467
logbufsz 393
logfilsiz 452
loghead 453
logindexbuild 467
logpath 453
logprimary 454
logretain 468
logsecond 456
log_retain_status 501
maxagents 37, 441
maxappls 442
maxcagents 443
maxfilop 444
MAXLOCKS 426
maxtotfilop 445
max_connections 37, 438
max_connretries 522
max_coordagents 439
max_querydegree 525
max_time_diff 523
mincommit 469
min_dec_div_3 414
min_priv_mem 405
mirrorlogpath 458
mon_heap_sz 422
multipage_alloc 501
newlogpath 459
nname 512
nodetype 537
notifylevel 530
numarchretry 470
numdb 37, 537
構成パラメーター *(続き)*
numsegs 435
num_db_backups 483
num_estore_segs 37, 432
num_freqvalues 507
num_initagents 446
num_initfenced 450
num_iocleaners 432
num_ioservers 434
num_poolagents 447
num_quantiles 508
overflowlogpath 461
pckcachesz 394
priv_mem_thresh 406
query_heap_sz 407
rec_his_retentn 484
release 495
restore_pending 502
resync_interval 488
rollfwd_pending 502
rqrioblk 416
sched_enable 565
sched_userid 565
seqdetect 435
sheapthres 408
sheapthres_shr 396
smtp_server 566
softmax 471
sortheap 410
spm_log_file_sz 488
spm_log_path 489
spm_max_resync 490
spm_name 490
srvcon_auth 548
srvcon_gssplugin_list 549
srvcon_pw_plugin 550
srv_plugin_mode 551
start_stop_time 524
stat_heap_sz 411
stmtheap 412
svcename 513
sysadm_group 551
sysctrl_group 552
sysmaint_group 553
sysmon_group 554
territory 496
tm_database 491
toolscat_db 567
toolscat_inst 567
toolscat_schema 568
tpname 514
tp_mon_name 539
trackmod 485
trust_allclnts 555
trust_clntauth 556
tsm_mgmtclass 485
構成パラメーター *(続き)*
tsm_nodename 486
tsm_owner 486
tsm_password 487
userexit 473
user_exit_status 502
use_sna_auth 557
util_heap_sz 397
util_impact_lim 541
vendoropt 474
ノード間リカバリーの例 685
構成ファイル
説明 361
ロケーション 361
構成ファイルリリース・レベル構成パラメーター 495
高速コミュニケーション・マネージャー (FCM)
説明 37
コマンド
db2adutl
ノード間リカバリーの例 685
コマンド・ヘルプ
呼び出し 714
コミット
グループ化するコミット数 (mincommit) 469
コントロール・センター
イベント解析プログラム 299
スナップショット・モニター 299
コンパイラー
書き直し
暗黙の述部の追加 162
相関副照会 160
ビューのマージ 158
コンパウンド SQL
使用方法 96

[サ行]

サーバー・インスタンスのディスカバリー構成パラメーター 516
最初のアクティブ・ログ・ファイル構成パラメーター 453
最適化
パーティション内並列処理 193
MDC 表の最適化 195
最適化クラス
設定 85
選択 81
リストと説明 83
再編成
表
実行時の決定 272
索引
いつ作成するか 279

索引 *(続き)*
カタログ統計の収集 115
管理 277, 285
クラスター率 172
クラスタリング 20
計画 279
構造 26
再編成 287
索引再作成時点構成パラメーター 481
索引の手動更新の規則 147
収集される詳細統計データ 135
スキャン 26
設計に役立つウィザード 227
それを使用するデータ・アクセス方式 169
タイプ 2 の説明 285
タイプがネクスト・キー・ロッキングに与える影響 79
デフラグ、オンライン 289
パフォーマンスのヒント 282
標準表での管理 20
ブロック索引スキャンのロック・モード 74
利点 277
MDC 表の管理 23
索引再作成時点構成パラメーター 481
索引スキャン
検索プロセス 26
使用法 26
それを使ってデータにアクセス 166
前のリーフ・ポインター 26
サマリー表
マテリアライズ照会表を参照してください。 196
時間
デッドロック構成パラメーター、チェック・インターバル 424
ノード間の最大差 523
システム管理スペース (SMS)
説明 16
自動構成パラメーター 363
自動再始動使用可能構成パラメーター 475
自動サマリー表
説明 196
自動統計収集 120
シャドー・ページング、長いオブジェクト 27
述部
暗黙
オプティマイザーによる追加 162
オプティマイザーによる変換 156
適用 160
特性 173
順次プリフェッチ
説明 259
照会
調整
指針 92
SELECT ステートメント 94
SELECT ステートメントの制限 87
REOPT BIND オプションを使用した最適化 165
照会最適化
構成パラメーター 153
パーティション・グループへの影響 103
照会の最大並列処理多重度構成パラメーター 525
照会最適化への影響 153
身体障害 717
ステートメント・ヒープ・サイズ構成パラメーター 412
ステートメント・レベルの分離の指定 48
ストアード・プロシージャー
使用方法 98
スナップショット
特定時点でのモニター 299
スペース・マップ・ページ (SMP)、DMS 表スペース 17
スレッド
説明 31
静的 SQL
最適化クラスの設定 85
制約
Explain 表 615
設計
設計アドバイザー 227
設計アドバイザー 7, 227, 279
制限と制約事項 232
パーティション・データベースの移行のため 231
ワークロードの定義 230
接続
経過時間 517
接続経過時間構成パラメーター 517
接続コンセントレーター
クライアント接続の改善 295
使用例 295
パーティション・データベースでのエージェントの使用 298
ソート
アクセス・プランへの影響 191
管理 268
共用ソートのソートヒープのしきい値 396
ソート・ヒープしきい値構成パラメーター 408
ソート・ヒープ・サイズ構成パラメーター 410
操作
オプティマイザーによるマージまたは移動 156

[タ行]

タイプ 2 索引
説明 26
ネクスト・キー・ロッキング 79
利点 285
タイムアウトの開始および停止構成パラメーター 524
チュートリアル 715
トラブルシューティングと問題判別 716
調整
構成パラメーター 363
長フィールド
キャッシング動作 290
通信
接続経過時間 517
データの再分散
エラー・リカバリー 339
指示 336
指針 333
必要性の判別 335
プロセスの説明 333
ログ・スペース所要量 338
データの挿入
索引に基づいて表をクラスター化するとき 28
処理 28
データベース 43
構成パラメーターのサマリー 370
照合情報 494
テリトリー・コード構成パラメーター 495
並行アクティブ・データベースの最大数 537
リリース・レベル構成パラメーター 495
autorestart 構成パラメーター 475
backup_pending 構成パラメーター 500
codepage 構成パラメーター 494
codeset 構成パラメーター 494
territory 構成パラメーター 496
データベース管理スペース (DMS)
説明 17
表スペース・アドレス・マップ 18
データベース共用メモリー・サイズ構成パラメーター 388
データベース・グローバル・メモリー
構成パラメーター 243

データベース・システム・モニター
デフォルトのデータベース・システム・モニター・スイッチ構成パラメーター 531
データベース・ディレクトリー
構造の説明 13
データベース・テリトリー・コード構成パラメーター 495
データベース・パーティション
追加 322
データベース・パーティション・サーバー
複数パーティション処理での 31
データベース・バックアップの数構成パラメーター 483
データベース・ヒープ構成パラメーター 389
データベース・マネージャー 43
共用メモリーの使用 240
構成パラメーターのサマリー 370
タイムアウトの開始 524
タイムアウトの停止 524
マシン・ノード・タイプ構成パラメーター 537
データベース・モニター
使用 299
データベース・ログ・パスの変更構成パラメーター 459
データ・サンプリング
統計収集 116
TABLESAMPLE の使用 92
データ・ストリーム情報
db2expln による表示 662
データ・ソース
入出力速度とパフォーマンス 206
データ・ページ、標準表における 20
データ・リンクのコピー数構成パラメーター 497
データ・リンク・アクセス・トークン満了インターバル構成パラメーター 497
データ・リンク・サポートの使用可能化構成パラメーター 496
データ・リンク・トークン・アルゴリズム構成パラメーター 498
ディスカバリー・モード構成パラメーター 515
ディスク
ストレージ・パフォーマンス要因 12
ディレクトリー・キャッシュ・サポート構成パラメーター
説明 418
デッドロック
検査 424
検出機能 11
説明 11
パフォーマンスへの影響 55
dlchktime 構成パラメーター 424
デフォルト・データベース・パス構成パラメーター 545
デフラグ
索引 289
統計
自動収集 119, 120
統計収集
サンプリング 116
統計プロファイル
生成 117
同時エージェントの最大数構成パラメーター 443
動的 SQL
最適化クラスの設定 85
動的構成 367
特定時点でのモニター 299
ドット 10 進シンタックス・ダイアグラム 719
トラブルシューティング
オンライン情報 716
チュートリアル 716
トランザクション処理モニター
トランザクション・プロセッサー・モニター名構成パラメーター 539
トリガー
Explain 表 615
ドロップ後のデータ・リンク時間構成パラメーター 498

[ナ行]

入出力並列処理
管理 266
認証
全クライアントのトラステッド化構成パラメーター 555
トラステッド・クライアント認証構成パラメーター 556
SNA 認証の使用構成パラメーター 557
認証 DAS 構成パラメーター 558
ネクスト・キーロック
索引タイプ、影響 79
索引を変換して最小化 287
タイプ 2 索引 285
ネスト・ループ結合 176
ノード 43
コーディネーター・エージェントの最大数 439
最大の時刻差 523
接続経過時間 517
ノード間最大時差構成パラメーター 523
ノード間データベース・リカバリーの例 685
ノード接続再試行回数構成パラメーター 522

[ハ行]

パーティション
追加
実行中のシステムへの 324
停止中のシステムへの 327
NT システムへの 325
ドロップ 330
パーティション内並列処理
最適化ストラテジー 193
パーティション内並列処理機能の使用可能化構成パラメーター 525
パーティション・グループが照会最適化に与える影響 103
パーティション・データベース
結合戦略 183
結合方式 185
データの再分散、エラー・リカバリー 339
ノード追加時のエラー 329
非相関化、照会の 160
複製マテリアライズ照会表 181
バインド
構成パラメーターの変更 364
分離レベル 48
バックアップ
変更されたページの追跡 485
バックアップ・ペンディング標識構成パラメーター 500
ハッシュ結合
説明 176
そのパフォーマンス調整 176
バッファー・プール
開始時のメモリーの割り振り 255
照会最適化への影響 153
使用方法 248
データ・ページ管理 254
複数
管理 255
ページ・サイズ 255
利点 255
ブロック・ベース、プリフェッチのパフォーマンス 261
ページ・クリーナー、チューニング 251
ページ・クリーニング方法 253
ラージ、利点 255
2 次 250
パフォーマンス
エレメント 3
向上プロセスの開発 5
最適化クラスの調整 85
チューニングの限度 7
調整 4
クイック・スタートのヒント 7
ユーザー入力 6

パフォーマンス *(続き)*
ディスク装置要因 12
フェデレーテッド・データベース・システム 199
db2batch ベンチマーク・ツール 350
REOPT BIND オプションを使用した照会最適化 165
非相関化、照会の
コンパイラーによる書き直し 160
ビュー
オプティマイザーによる述部プッシュダウン 160
オプティマイザーによるマージ 158
表
アクセス
パス 68
db2expln によって表示された情報 653
キュー、パーティション・データベースにおける結合のための戦略 183
再編成 275
インプレース、オンライン・モードでの 270
従来の、オフライン・モードでの 270
必要性の削減 270
必要性の判別 272
標準
管理 20
マルチディメンション・クラスタリング 23
ロック・タイプ 53
ロック・モード
標準表用の 68
MDC 表の RID スキャンおよび表スキャンの 70
表スペース
オーバーヘッド 103
照会最適化への影響 103
ロック・タイプ 53
DMS 17
TRANSFERRATE の設定 103
プール内エージェントの初期数構成パラメーター 446
プール・サイズの制御、エージェントの 447
フェデレーテッド・データベース
グローバル最適化 206
コンパイラー・フェーズ 199
サーバー・オプション 106
システム・サポート構成パラメーター 536
照会が評価される場面の分析 204
照会情報 668
照会の db2expln 出力 680
照会のグローバル分析 209
フェデレーテッド・データベース *(続き)*
プッシュダウン分析 199
並行性制御 43
フェデレーテッド・データベース認証をバイパス構成パラメーター 547
副照会
相関
書き直し方法 160
プッシュダウン分析
フェデレーテッド・データベース照会 199
フリー・スペース制御レコード (FSCR)
標準表における 20
MDC 表の 23
プリコンパイル
分離レベル 48
プリフェッチ
順次 259
順次リスト 263
説明 258
入出力サーバー構成 263
パーティション内並列処理のパフォーマンス 258
ブロック・ベースのバッファー・プール 261
並列入出力 264
プロセス・モデル
概要 31
更新用 30
SQL コンパイラーの 149
ブロック・ベースのバッファー・プール 261
プロトコル
NetBIOS ワークステーション名構成パラメーター 512
TCP/IP サービス名構成パラメーター 513
分位分散統計 127
分散統計
拡張例の使用 131
手動更新の規則 145
説明 127
のオプティマイザーの使用 130
文書
表示 702
分離レベル
指定 48
ステートメント・レベル 48
パフォーマンスへの影響 44
並行性制御のロック 51
ページ、データ 20
ページ・クリーナー
チューニング番号 251
並行アクティブ・データベースの最大数構成パラメーター 537
並行性
ロッキングに影響を与える要因 76
並行性制御
アクティブ・アプリケーションの最大数 442
に関する一般的な問題 43
フェデレーテッド・データベースの 43
並列処理
影響
dft_degree 構成パラメーター 99
intra_parallel 構成パラメーター 99
max_querydegree 構成パラメーター 99
照会の最大並列処理多重度構成パラメーター 525
程度の設定 99
入出力
管理 266
サーバー構成 263
パーティション内
最適化ストラテジー 193
パーティション内並列処理機能の使用可能化構成パラメーター 525
非 SMP 環境 99
並列処理、db2expln 出力により表示される情報 665
ヘルス・モニター構成パラメーター 529
ヘルプ
コマンドの
呼び出し 714
表示 702, 704
メッセージの
呼び出し 714
SQL ステートメントの
呼び出し 715
変更されたページの追跡使用可能化構成パラメーター 485
ベンチマーク
概要 347
準備 348
ステップの要約 356
テスト方式 347
テスト・プロセス 356
レポートの例 358
db2batch ツール 350
SQL ステートメント 348
方式
ネスト・ループ結合 176

[マ行]

マージ結合 176
マップ・ページ
エクステント 17
スペース 17

マテリアライズ照会表 (MQT)
自動サマリー表 196
複製、パーティション・データベースで 181
マルチサイト更新 43
マルチディメンション・クラスタリング (MDC)
最適化ストラテジー 195
表および索引の管理 23
ミラー・ログ・パス構成パラメーター 458
メッセージ・ヘルプ
呼び出し 714
メモリー
インスタンス・メモリー構成パラメーター 420
影響を与えるパラメーターの調整 245
開始時のバッファー・プール割り振り 255
グローバル、コンポーネント 243
使用の編成 237
ステートメント・ヒープ・サイズ構成パラメーター 412
ソート・ヒープしきい値構成パラメーター 408
ソート・ヒープ・サイズ構成パラメーター 410
パッケージ・キャッシュ・サイズ構成パラメーター 394
割り振られるタイミング 237
applheapsz 構成パラメーター 404
aslheapsz 構成パラメーター 413
dbheap 構成パラメーター 389
メモリー要件
FCM バッファー・プール 242
メモリー・モデル
説明 37
データベース・マネージャー共用メモリー 240
文字の変換
アプリケーション・パフォーマンスへの影響 97
モニター
方法 299
モニター・スイッチ
更新 299
問題判別
オンライン情報 716
チュートリアル 716

[ヤ行]

ユーザー定義関数 (UDF)
の統計の入力 138
ユーザー出口使用可能構成パラメーター 473
ユーザー出口状況標識構成パラメーター 502
「有効期限切れタスクの実行」構成パラメーター 563
呼び出し
コマンド・ヘルプ 714
メッセージ・ヘルプ 714
SQL ステートメント・ヘルプ 715

[ラ行]

ラージ・オブジェクト (LOB) データ・タイプ
キャッシング動作 290
リカバリー
索引再作成時点構成パラメーター 481
自動再始動使用可能構成パラメーター 475
データベース・バックアップの数構成パラメーター 483
ノード間の例 685
バックアップ・ペンディング標識構成パラメーター 500
ユーザー出口状況標識構成パラメーター 502
リストア・ペンディング構成パラメーター 502
ロード・リカバリー・セッションのデフォルト数構成パラメーター 476
ロールフォワード・ペンディング標識構成パラメーター 502
ログ保存状況標識構成パラメーター 501
リカバリー範囲およびソフト・チェックポイント・インターバル構成パラメーター 471
リカバリー履歴の保存期間構成パラメーター 484
リスト・プリフェッチ 263
リモート・データ・サービス・ノード名構成パラメーター 512
レコード ID (RID)、標準表における 20
レジストリー変数
概要 571
DB2ACCOUNT 572
DB2ADMINSERVER 607
DB2AFFINITIES 591
DB2ASSUMEUPDATE 591
DB2ATLD_PWFILE 585
DB2BIDI 572
DB2BPVARS 591
DB2BQTIME 583
DB2BQTRY 583
DB2CHECKCLIENTINTERVAL 579
DB2CHGPWD_ESE 585
レジストリー変数 (続き)
DB2CHKPTR 591
DB2CHKSQLDA 591
DB2CODEPAGE 572
DB2COMM 579
DB2CONNECT_IN_APP_PROCESS 575
DB2DBDFT 572
DB2DBMSADDR 572
DB2DEFPREP 607
DB2DISCOVERYTIME 572
DB2DMNBCKCTLR 607
DB2DOMAINLIST 575
DB2ENVLIST 575
DB2GRAPHICUNICODESERVER 572
DB2INCLUDE 572
DB2INSTANCE 575
DB2INSTDEF 572
DB2INSTOWNER 572
DB2INSTPROF 575
DB2IQTIME 583
DB2JD_PORT_NUMBER 579
DB2LDAPCACHE 607
DB2LDAPHOST 607
DB2LDAP_BASEDN 607
DB2LDAP_CLIENT_PROVIDER 607
DB2LDAP_SEARCH_SCOPE 607
DB2LIBPATH 575
DB2LOADREC 607
DB2LOCALE 572
DB2LOCK_TO_RB 607
DB2MAXFSCRSEARCH 591
DB2MEMDISCLAIM 591
DB2MEMMAXFREE 591
DB2NBADAPTERS 579
DB2NBBRECVNCBS 579
DB2NBCHECKUPTIME 579
DB2NBDISCOVERRCVBUFS 572
DB2NBINTRLISTENS 579
DB2NBRECVBUFFSIZE 579
DB2NBRESOURCES 579
DB2NBSENDNCBS 579
DB2NBSESSIONS 579
DB2NBXTRANCBS 579
DB2NETREQ 579
DB2NODE 575
DB2NOEXITLIST 607
DB2NTMEMSIZE 591
DB2NTNOCACHE 591
DB2NTPRICLASS 591
DB2NTWORKSET 591
DB2OPTIONS 572
DB2PATH 575
DB2PORTRANCE 585
DB2PRIORITIES 591
DB2REMOTEPREG 607
DB2RETRY 579

レジストリー変数 *(続き)*
DB2RETRYTIME 579
DB2ROUTINE_DEBUG 607
DB2RQTIME 583
DB2SERVICETPINSTANCE 579
DB2SLOGON 572
DB2SORCVBUF 607
DB2SORT 607
DB2SOSNDBUF 579
DB2SYSPLEX_SERVER 579
DB2SYSTEM 607
DB2TCPCONNMGRS 579
DB2TERRITORY 572
DB2TIMEOUT 572
DB2TRACEFLUSH 572
DB2TRACENAME 572
DB2TRACEON 572
DB2TRCSYSERR 572
DB2YIELD 572
DB2_ALLOCATION_SIZE 591
DB2_ANTIJOIN 586
DB2_APM_PERFORMANCE 591
DB2_AVOID_PREFETCH 591
DB2_AWE 591
DB2_BINSORT 591
DB2_CLPPROMPT 583
DB2_CORRELATED_ PREDICATES 586
DB2_DJ_COMM 607
DB2_DOCHOST 607
DB2_DOCPORT 607
DB2_ENABLE_BUFPD 591
DB2_ENABLE_LDAP 607
DB2_EXTENDED_ OPTIMIZATION 591
DB2_FALLBACK 607
DB2_FMP_COMM_HEAPSZ 607
DB2_FORCE_FCM_BP 585
DB2_FORCE_NLS_CACHE 579
DB2_GRP_LOOKUP 607
DB2_HASH_JOIN 586
DB2_INDEX_TYPE2 572
DB2_INLIST_TO_NLJN 586
DB2_KEEPTABLELOCK 591
DB2_LGPAGE_BP 591
DB2_LIC_STAT_SIZE 572
DB2_LIKE_VARCHAR 586
DB2_MINIMIZE_LISTPREFETCH 586
DB2_MMAP_READ 591
DB2_MMAP_WRITE 591
DB2_NEWLOGPATH2 607
DB2_NEW_CORR_SQ_FF 586
DB2_NO_FORK_CHECK 591
DB2_NO_MPFA_FOR_NEW_DB 591
DB2_NUM_FAILOVER_NODES 585
DB2_OBJECT_TABLE_ENTRIES 591
レジストリー変数 *(続き)*
DB2_OVERRIDE_BPF 591
DB2_PARALLEL_IO 575
DB2_PARTITIONEDLOAD__DEFAULT 585
DB2_PINNED_BP 591
DB2_PRED_FACTORIZE 586
DB2_REDUCED_ OPTIMIZATION 586
DB2_SCATTERED_IO 591
DB2_SELECTIVITY 586
DB2_SKIPDELETED 591
DB2_SMS_TRUNC_TMP_TABLE_ THRESH 591
DB2_SORT_AFTER_TQ 591
DB2_TRUSTED_BINDIN 591
DB2_USE_ALTERNATE_PAGE_ CLEANING 591
DB2_USE_PAGE_CONTAINER_TAG 575
DB2_VENDOR_INI 607
DB2_VIEW_REOPT_VALUES 572
DB2_VI_DEVICE 579
DB2_VI_ENABLE 579
DB2_VI_VIPL 579
DB2_XBSA_LIBRARY 607
DLFM_ASNCOPYD_PORT 605
DLFM_BACKUP_DIR_NAME 605
DLFM_BACKUP_TARGET_ LIBRARY 605
DLFM_GC_MODE 605
DLFM_INSTALL_PATH 605
DLFM_PORT 605
DLFM_START_ASNCOPYD 605
DLFM_TSM_MGMTCLASS 605
列
サブエレメントの統計の収集 136
統計の手動更新の規則 144
特定の分散統計の収集 113
ロールフォワード・ユーティリティー
ロールフォワード・ペンディング標識 502
ロールフォワード・リカバリー
定義 27
ロギング
循環の定義 27
ログ・レコードの保存の定義 27
ログ
オーバーフロー・ログ・パス構成パラメーター 461
ガバナーによる作成 316
最初のアクティブ・ログ・ファイル構成パラメーター 453
ミラー・ログ・パス構成パラメーター 458
ユーザー出口使用可能構成パラメーター 473
ログ *(続き)*
リカバリー範囲およびソフト・チェックポイント・インターバル構成パラメーター 471
ログ保存使用可能構成パラメーター 468
ログ保存状況標識構成パラメーター 501
ログ・ディスク・フル時のブロック構成パラメーター 464
ログ・バッファー・サイズ構成パラメーター 393
ログ・ファイルのサイズ構成パラメーター 452
ログ・ファイルのロケーション構成パラメーター 453
1 次ログ・ファイル数構成パラメーター 454
2 次ログ・ファイル数構成パラメーター 456
newlogpath 構成パラメーター 459
ログ・ディスク・フル時のブロック (blk_log_dsk_ful) 構成パラメーター 464
ログ・バッファー 27
ログ・ファイル・スペース
データ再分散に必要な 338
ロッキング
エスカレーション前のロック・リストの最大パーセント 426
調整 60
デッドロック・チェック・インターバル構成パラメーター 424
ロック・リストの最大ストレージ 390
ロック
アプリケーションのタイプが与える影響 77
意図的共用 (IS) モード 53
意図的なし (IN) モード 53
意図排他 (IX) モード 53
意図排他共用 (SIX) モード 53
エスカレーション
定義された 51
訂正 62
防止 62
共用 (S) モード 53
更新 (U) モード 53
据え置き 64
タイプ 53
タイプ互換性表 66
データ・アクセス・プランが与える影響 78
デッドロック 11
ネクスト・キー・ロッキング 79
排他的 (X) モード 53
パフォーマンス要因 55

ロック *(続き)*
標準表用のモードおよびアクセス・パス 68
ブロック索引スキャンでのモード 74
MDC 表の表スキャンおよび RID 索引スキャンのロック・モード 70
superexclusive (Z) モード 53
ロック待機
パフォーマンスへの影響 55
ロックの互換性
パフォーマンスへの影響 55
ロックの変換
パフォーマンスへの影響 55
ロック・エスカレーション 55
ロック・リスト用最大ストレージ構成パラメーター 390
論理ノード、「データベース・パーティション・サーバー」を参照 31
論理パーティション
複数 31

[ワ行]

ワークロード
設計アドバイザー 227
設計アドバイザーの 230

[数字]

10 進数演算機構
10 進数除算の位取り 3 構成パラメーター 414

A

ADVISE_INDEX 表 634
ADVISE_INSTANCE 表 637
ADVISE_MQT 表 638
ADVISE_PARTITION 表 640
ADVISE_TABLE 表 641
ADVISE_WORKLOAD 表 642
agentpri 構成パラメーター 436
agent_stack_sz 構成パラメーター 402
ALTER TABLESPACE ステートメント
例 103
alt_collate 構成パラメーター 493
APPC トランザクション・プログラム名構成パラメーター 514
APPEND モードの挿入処理 28
appgroup_mem_sz 構成パラメーター 399
APPLHEAPSZ 構成パラメーター
使用法 404
app_ctl_heap_sz 398
archretrydelay 構成パラメーター 463
aslheapsz 構成パラメーター 413
audit_buf_sz 構成パラメーター 417
authentication 構成パラメーター 542
autonomic_switches 構成パラメーター 510
AUTO_PROF_UPD
使用 117
AUTO_STATS_PROF
使用 117
avg_appls 構成パラメーター 438

B

backup_pending 構成パラメーター 500
blk_log_dsk_ful 構成パラメーター 464

C

catalogcache_sz 構成パラメーター 386
catalog_noauth 構成パラメーター 543
chngpgs_thresh 構成パラメーター 428
clint_krb_plugin 構成パラメーター 544
clnt_pw_plugin 構成パラメーター 545
codepage 構成パラメーター 494
codeset 構成パラメーター 494
collate_info 構成パラメーター 494
comm_bandwidth 構成パラメーター
照会最適化への影響 153
説明 533
conn_elapse 構成パラメーター 517
contact_host 構成パラメーター 559
cpuspeed 構成パラメーター
照会最適化への影響 153
説明 534
CURRENT EXPLAIN MODE 特殊レジスター
Explain データのキャプチャー 223
CURRENT EXPLAIN SNAPSHOT 特殊レジスター
Explain 情報のキャプチャー 223

D

DAS 構成パラメーター
認証 558
contact_host 559
dasadm_group 561
das_codepage 560
das_territory 560
db2system 561
exec_exp_task 563
jdk_64_path 563
jdk_path 564
sched_enable 565
sched_userid 565
smtp_server 566
DAS 構成パラメーター *(続き)*
toolscat_db 567
toolscat_inst 567
toolscat_schema 568
dasadm_group 構成パラメーター 561
das_codepage 構成パラメーター 560
das_territory 構成パラメーター 560
database_consistent 構成パラメーター 501
database_level 構成パラメーター 495
database_memory 構成パラメーター 388
DATALINK データ・タイプ
構成パラメーター 496
DB2 アーキテクチャーの概要 9
DB2 インフォメーション・センター 692
呼び出し 702
DB2 資料
PDF ファイルの印刷 711
DB2 資料の注文 712
DB2 チュートリアル 715
DB2ACCOUNT 572
DB2ADMINSERVER 607
db2adutl コマンド
ノード間リカバリーの例 685
db2advis 7, 279
DB2AFFINITIES 591
DB2ASSUMEUPDATE 591
DB2ATLD_PWFILE 585
db2batch ベンチマーク・ツール
テストの作成 350
例 352
DB2BIDI 572
DB2BPVARS 591
DB2BQTIME 583
DB2BQTRY 583
DB2CHECKCLIENTINTERVAL 579
DB2CHGPWD_ESE 585
DB2CHKPTR 591
DB2CHKSQLDA 591
DB2CLIINIPATH 607
DB2CODEPAGE 572
DB2COMM 579
DB2CONNECT_IN_APP_PROCESS 575
DB2DBDFT 572
DB2DBMSADDR 572
DB2DEFPREP 607
DB2DISCOVERYTIME 572
DB2DMNBCKCTLR 607
DB2DOMAINLIST 575
db2empfa コマンド 16
DB2ENVLIST 575
db2exfmt ツール 683
db2expln ツール
構文およびパラメーター 644
出力 651

db2expln ツール *(続き)*
出力サンプル
完全な並列処理による複数パーティションのプラン 678
説明 671
パーティション間並列処理による複数パーティションのプラン 675
パーティション内並列処理による単一パーティション・プラン 674
フェデレーテッド・データベース・プラン 680
並列処理を使用しない 672
使用上の注意 649
表示される情報
一時表 658
結合 660
集約 665
挿入、更新、削除 663
その他 669
データ・ストリーム 662
表アクセス 653
並列処理 665
ブロックおよび RID 準備の情報 664
DB2GRAPHICUNICODESERVER 572
DB2INCLUDE 572
DB2INSTANCE 575
DB2INSTDEF 572
DB2INSTOWNER 572
DB2INSTPROF 575
DB2IQTIME 583
DB2JD_PORT_NUMBER 579
DB2LDAPCACHE 607
DB2LDAPHOST 607
DB2LDAP_BASEDN 607
DB2LDAP_CLIENT_PROVIDER 607
DB2LDAP_SEARCH_SCOPE 607
DB2LIBPATH 575
DB2LOADREC 607
DB2LOCALE 572
DB2LOCK_TO_RB 607
DB2MAXFSCRSEARCH 591
DB2MEMDISCLAIM 591
DB2MEMMAXFREE 591
DB2NBADAPTERS 579
DB2NBCHECKUPTIME 579
DB2NBDISCOVERRCVBUFS 572
DB2NBINTRLISTENS 579
DB2NBRECVBUFFSIZE 579
DB2NBRECVNCBS 579
DB2NBRESOURCES 579
DB2NBSENDNCBS 579
DB2NBSESSIONS 579
DB2NBXTRANCBS 579
DB2NODE 575
サーバー追加時のエクスポート 324, 325, 327
DB2NOEXITLIST 607
DB2NTMEMSIZE 591
DB2NTNOCACHE 591
DB2NTPRICLASS 591
DB2NTWORKSET 591
DB2PATH 575
DB2PORTRANGE 585
DB2PRIORITIES 591
DB2REMOTEPREG 607
DB2RETRY 579
DB2RETRYTIME 579
DB2ROUTINE_DEBUG 607
DB2RQTIME 583
DB2SERVICETPINSTANCE 579
DB2SORCVBUF 607
DB2SORT 607
DB2SOSNDBUF 579
DB2SYSPLEX_SERVER 579
DB2SYSTEM 607
db2system 構成パラメーター 561
DB2TCPCONNMGRS 579
DB2TERRITORY 572
DB2_ALLOCATION_SIZE 591
DB2_ANTIJOIN 586
DB2_APM_PERFORMANCE 591
DB2_AVOID_PREFETCH 591
DB2_AWE 591
DB2_BINSORT 591
DB2_CLPPROMPT 583
DB2_CORRELATED_ PREDICATES 586
DB2_DJ_COMM 607
DB2_DOCHOST 607
DB2_DOCPORT 607
DB2_ENABLE_BUFPD 591
DB2_ENABLE_LDAP 607
DB2_EVALUNCOMMITTED 591
DB2_EXTENDED_ OPTIMIZATION 591
DB2_FALLBACK 607
DB2_FMP_COMM_HEAPSZ 607
DB2_FORCE_FCM_BP 585
DB2_FORCE_NLS_CACHE 579
DB2_GRP_LOOKUP 607
DB2_HASH_JOIN 586
DB2_INDEX_TYPE2 572
DB2_INLIST_TO_NLJN 586
DB2_KEEPTABLELOCK 591
DB2_LGPAGE_BP 591
DB2_LIC_STAT_SIZE 572
DB2_LIKE_VARCHAR 586
DB2_MINIMIZE_LISTPREFETCH 586
DB2_MMAP_READ 591
DB2_MMAP_WRITE 591
DB2_NEWLOGPATH2 607
DB2_NEW_CORR_SQ_FF 586
DB2_NO_FORK_CHECK 591
DB2_NO_MPFA_FOR_NEW_DB 591
DB2_NUM_FAILOVER_NODES 585
DB2_OBJECT_TABLE_ENTRIES 591
DB2_OVERRIDE_BPF 591
DB2_PARALLEL_IO 575
DB2_PARTITIONEDLOAD_DEFAULT 585
DB2_PINNED_BP 591
DB2_PRED_FACTORIZE 586
DB2_REDUCED_ OPTIMIZATION 586
DB2_SCATTERED_IO 591
DB2_SELECTIVITY 586
DB2_SMS_TRUNC_TMP_TABLE_THRESH 591
DB2_SORT_AFTER_TQ 591
DB2_TRUSTED_BINDIN 591
DB2_USE_ALTERNATE_PAGE_CLEANING 591
使用法 253
DB2_USE_PAGE_CONTAINER _TAG 575
DB2_USE_PAGE_CONTAINER_TAG 575
DB2_VENDOR_INI 607
DB2_VIEW_REOPT_VALUES 572
DB2_VI_DEVICE 579
DB2_VI_ENABLE 579
DB2_VI_VIPL 579
DB2_XBSA_LIBRARY 607
DBHEAP 構成パラメーター 389
dftdbpath 構成パラメーター 545
dft_account_str 構成パラメーター 535
dft_degree 構成パラメーター 503
照会最適化への影響 153
dft_extent_sz 構成パラメーター 429
dft_loadrec_ses 構成パラメーター 476
dft_monswitches 構成パラメーター 531
dft_mon_bufpool 構成パラメーター 531
dft_mon_lock 構成パラメーター 531
dft_mon_sort 構成パラメーター 531
dft_mon_stmt 構成パラメーター 531
dft_mon_table 構成パラメーター 531
dft_mon_timestamp 構成パラメーター 531
dft_mon_uow 構成パラメーター 531
dft_mttb_types 構成パラメーター 504
dft_prefetch_sz 構成パラメーター 430
dft_queryopt 構成パラメーター 504
dft_refresh_age 構成パラメーター 505
dft_sqlmathwarn 構成パラメーター 505
diaglevel 構成パラメーター 527
diagpath 構成パラメーター 528
dir_cache 構成パラメーター 418
discover (DAS) 構成パラメーター 562
discover 構成パラメーター 515
discover_db 構成パラメーター 516
discover_inst 構成パラメーター 516
dlchktime 構成パラメーター 424
DLFM_ASNCOPYD_PORT 605

DLFM_BACKUP_DIR_NAME 605
DLFM_BACKUP_TARGET 605
DLFM_BACKUP_TARGET_LIBRARY 605
DLFM_GC_MODE 605
DLFM_INSTALL_PATH 605
DLFM_PORT 605
DLFM_START_ASNCOPYD 605
DLFM_TSM_MGMTCLASS 605
dl_expint 構成パラメーター 497
dl_num_copies 構成パラメーター 497
dl_time_drop 構成パラメーター 498
dl_token 構成パラメーター 498
dl_wt_iexpint 構成パラメーター 499
DMS 装置
  キャッシング動作 290
  バッファリングの動作 290
dynexpln ツール
  構文およびパラメーター 651
  説明された出力 651
dyn_query_mgmt 構成パラメーター
  Query Patroller 用 492

E

estore_seg_sz 構成パラメーター
  説明 431
  メモリー管理 37
exec_exp_task 構成パラメーター 563
Explain インスタンス 217
Explain 機能
  収集された情報を使用 216
  情報のキャプチャー 223
  情報の分析 226
  スナップショットの作成 223
  説明 213
  表示される情報
    インスタンス 221
    データ演算子 220
    データ・オブジェクト 219
Explain ツール
  概要 214
  使用 643
  db2exfmt 214
  db2expln 214
  dynexpln 214
  Visual Explain 214
Explain 表
  概要 615
  データのフォーマット・ツール 683
  編成 217
EXPLAIN_ARGUMENT 表 616
EXPLAIN_INSTANCE 表 620
EXPLAIN_OBJECT 表 622
EXPLAIN_OPERATOR 表 625
EXPLAIN_PREDICATE 表 627
EXPLAIN_STATEMENT 表 629
EXPLAIN_STREAM 表 632

F

failarchpath 構成パラメーター 464
FCM バッファー・プール
  図 242
  メモリー要件 242
fcm_num_anchors 構成パラメーター 518
fcm_num_buffers 構成パラメーター 519
fcm_num_connect 構成パラメーター 520
fcm_num_rqb 構成パラメーター 521
federated 構成パラメーター 536
fed_noauth 構成パラメーター 547
fenced プロセスの最大数構成パラメーター 448
fenced プロセスの初期数構成パラメーター 450
fenced_pool 構成パラメーター 448
FOR FETCH ONLY 文節
  照会調整の 87
FOR READ ONLY 文節
  照会調整の 87

G

groupheap_ratio 構成パラメーター 401
group_plugin 構成パラメーター 547

H

hadr_db_role 構成パラメーター 476
hadr_local_host 構成パラメーター 477
hadr_local_svc 構成パラメーター 478
hadr_remote_host 構成パラメーター 478
hadr_remote_inst 構成パラメーター 479
hadr_remote_svc 構成パラメーター 479
hadr_syncmode 構成パラメーター 480
hadr_timeout 構成パラメーター 481
health_mon 構成パラメーター 529
HTML 文書
  更新 703

I

INCLUDE 文節
  索引に必要なスペースでの影響 20
indexrec 構成パラメーター 481
instance_memory 構成パラメーター 420
intra_parallel 構成パラメーター 525
IS (意図共用) モード 53

J

Java Development Kit インストール・パス (DAS) 構成パラメーター 564
Java Development Kit インストール・パス 構成パラメーター 536
Java インタープリター最大ヒープ・サイズ構成パラメーター 421
java_heap_sz 構成パラメーター 421
jdk_64_path 構成パラメーター 563
jdk_path DAS 構成パラメーター 564
jdk_path 構成パラメーター 536

K

keepfenced 構成パラメーター 449

L

LOB (ラージ・オブジェクト) データ・タイプ
  キャッシング動作 290
local_gssplugin 構成パラメーター 548
LOCK TABLE ステートメント
  ロック・エスカレーションの最小化 62
locklist 構成パラメーター
  照会最適化への影響 153
  説明 390
LOCKSIZE 節 51
locktimeout 構成パラメーター 425
logarchmeth1 構成パラメーター 465
logarchmeth2 構成パラメーター 466
logarchopt1 構成パラメーター 466
  ノード間リカバリーの例 685
logarchopt2 構成パラメーター 467
LOGBUFSZ 構成パラメーター 393
logfilsiz 構成パラメーター 452
loghead 構成パラメーター 453
logindexbuild 構成パラメーター 467
logpath 構成パラメーター 453
logprimary 構成パラメーター 454
logretain 構成パラメーター 468
logsecond 構成パラメーター 456
log_retain_status 構成パラメーター 501

M

maxagents 構成パラメーター 441
  メモリー管理 37
  メモリー使用への影響 237
maxappls 構成パラメーター 442
  メモリー管理 37
  メモリー使用への影響 237
maxcagents 構成パラメーター 443

maxcoordagents 構成パラメーター 237
maxfilop 構成パラメーター 444
maxlocks 構成パラメーター 426
maxtotfilop 構成パラメーター 445
max_connretries 522
max_coordagents 構成パラメーター 439
max_logicagents 構成パラメーター 438
max_querydegree 構成パラメーター 525
max_time_diff 構成パラメーター 523
mincommit 構成パラメーター 469
MINPCTUSED 文節
オンライン索引デフラグ 20
min_dec_div_3 構成パラメーター 414
min_priv_mem 構成パラメーター 405
mirrorlogpath 構成パラメーター 458
mon_heap_sz 構成パラメーター 422
multipage_alloc 構成パラメーター 501
メモリーへの影響 16
SMS 表スペースでの設定 16

N

NetBIOS
ワークステーション名構成パラメーター 512
newlogpath 構成パラメーター 459
nname 構成パラメーター 512
nodetype 構成パラメーター 537
notifylevel 構成パラメーター 530
numarchretry 構成パラメーター 470
numdb 構成パラメーター 537
メモリー管理 37
メモリー使用への影響 237
numinitagents 構成パラメーター 446
numsegs 構成パラメーター 435
num_db_backups 構成パラメーター 483
num_estore_segs 構成パラメーター
説明 432
メモリー管理 37
num_freqvalues 構成パラメーター 507
num_initfenced 構成パラメーター 450
num_iocleaners 構成パラメーター 432
num_ioservers 構成パラメーター 434
num_poolagents 構成パラメーター 447
num_quantiles 構成パラメーター 508
NW (ネクスト・キー弱排他) モード 53

O

OPTIMIZE FOR 文節
照会調整の 87
overflowlogpath 構成パラメーター 461

P

pckcachesz 構成パラメーター 394
PCTFREE 文節
クラスタリングのスペースを保存する 20
priv_mem_thresh 構成パラメーター 406

Q

query_heap_sz 構成パラメーター 407

R

rec_his_retentn 構成パラメーター 484
release 構成パラメーター 495
REORG INDEXES コマンド 287
REORG TABLE コマンド
インプレース、オンライン・モードでの 275
従来の、オフライン・モードでの 275
REORG 方式の選択 275
REORGANIZE TABLE コマンド
索引および表 287
restore_pending 構成パラメーター 502
resync_interval 構成パラメーター 488
REXX 言語
分離レベルの指定 48
rollfwd_pending 構成パラメーター 502
rqrioblk 構成パラメーター 416
RUNSTATS
サンプリング統計 116
自動統計収集 119, 120
収集する統計 109
使用 112

S

sched_enable 構成パラメーター 565
sched_userid 構成パラメーター 565
SELECT ステートメント
出力の優先順位付け 87
DISTINCT 文節の除去 160
seqdetect 構成パラメーター 435
SET CURRENT QUERY OPTIMIZATION ステートメント 85
sheapthres 構成パラメーター 408
sheapthres_shr 構成パラメーター 396
SIX (意図的排他共用) モード 53
SMS コンテナーのデフォルト数構成パラメーター 435
smtp_server 構成パラメーター 566
softmax 構成パラメーター 471
sortheap 構成パラメーター
照会最適化への影響 153
sortheap 構成パラメーター *(続き)*
説明 410
spm_log_file_sz 構成パラメーター 488
spm_log_path 構成パラメーター 489
spm_max_resync 構成パラメーター 490
spm_name 構成パラメーター 490
SQL Explain 213
SQL コンパイラー
プロセスの説明 149
SQL ステートメント
ステートメント・ヒープ・サイズ構成パラメーター 412
ベンチマーク 348
SQL ステートメント・ヘルプ
呼び出し 715
SQLDBCON 構成ファイル 361
srvcon_auth 構成パラメーター 548
srvcon_gssplugin_list 構成パラメーター 549
srvcon_pw_plugin 構成パラメーター 550
srv_plugin_mode 構成パラメーター 551
start_stop_time 構成パラメーター 524
stat_heap_sz 構成パラメーター 411
stmtheap 構成パラメーター 412
stmtheap 構成パラメーターが照会最適化に与える影響 153
svcename 構成パラメーター 513
sysadm_group 構成パラメーター 551
sysctrl_group 構成パラメーター 552
sysmaint_group 構成パラメーター 553
sysmon_group 構成パラメーター 554

T

TABLESAMPLE
使用 92
TCP/IP サービス名構成パラメーター 513
territory 構成パラメーター 496
Tivoli Storage Manager (TSM)
管理クラス構成パラメーター 485
所有者名構成パラメーター 486
ノード名構成パラメーター 486
パスワード構成パラメーター 487
tm_database 構成パラメーター 491
toolscat_db 構成パラメーター 567
toolscat_inst 構成パラメーター 567
toolscat_schema 構成パラメーター 568
tpname 構成パラメーター 514
tp_mon_name 構成パラメーター 539
trackmod 構成パラメーター 485
trust_allclnts 構成パラメーター 555
trust_clntauth 構成パラメーター 556
tsm_mgmtclass 構成パラメーター 485
tsm_nodename 構成パラメーター 486
tsm_owner 構成パラメーター 486
tsm_password 構成パラメーター 487

U

userexit データベース構成パラメーター
473
user_exit_status 構成パラメーター 502
use_sna_auth 構成パラメーター 557
util_heap_sz 構成パラメーター 397
util_impact_lim 構成パラメーター
説明 541

V

vendoropt 構成パラメーター 474
ノード間リカバリーの例 685

W

W (弱排他) ロック・モード 53
WHERE 文節
述部用語の定義 173
Windows
パーティションの追加 325

X

X (排他) モード 53

IBM と連絡をとる

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Universal Database 製品に関する情報は、
http://www.ibm.com/software/data/db2/udb から入手できます。

このサイトには、技術ライブラリー、資料の注文方法、製品のダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースへのリンクに関する最新情報が掲載されています。

米国以外の国で IBM に連絡する方法については、 IBM Worldwide ページ (www.ibm.com/planetwide) にアクセスしてください。

Printed in Japan

SC88-9134-01

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12