

**IBM DB2 10.1
for Linux, UNIX, and Windows**

**問題判別およびデータベース・
パフォーマンスのチューニング**

IBM

**IBM DB2 10.1
for Linux, UNIX, and Windows**

**問題判別およびデータベース・
パフォーマンスのチューニング**

IBM

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、821 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

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

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

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

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

原典： SC27-3889-00
IBM DB2 10.1
for Linux, UNIX, and Windows
Troubleshooting and Tuning Database
Performance

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.4

© Copyright IBM Corporation 2006, 2012.

目次

本書について	vii
本書の構成	vii

第 1 部 パフォーマンスの概要 1

第 1 章 パフォーマンス・チューニングのツールと方法 5

ベンチマーク・テスト	5
ベンチマークの準備	6
ベンチマーク・テストの作成	7
ベンチマーク・テストの実行	9
ベンチマーク・テストの分析例	11

第 2 章 パフォーマンス・モニターのツールと方法 13

システム・パフォーマンスの操作モニター	13
システム・パフォーマンスのモニター・エレメントの基本セット	15
モニター・データ内の異常値	19
ガバナー・ユーティリティ	20
ガバナーの開始	20
ガバナー・デーモン	21
ガバナー構成ファイル	22
ガバナーの規則節	26
ガバナー・ログ・ファイル	31
ガバナーの停止	35

第 3 章 パフォーマンスに影響を及ぼす要因 37

システム体系	37
DB2 アーキテクチャーおよびプロセスの概要	37
DB2 プロセス・モデル	39
データベース・エージェント	45
良好なパフォーマンスのための構成	56
インスタンス構成	64
表スペースの設計	65
ディスク・ストレージのパフォーマンス要因	65
表スペースが照会の最適化に与える影響	65
データベース設計	68
表	68
索引	73
パーティションとクラスタリング	89
フェデレーテッド・データベース	97
リソースの利用	98
メモリーの割り振り	98
セルフチューニング・メモリーの概要	106
バッファー・プール管理	115
最初のユーザー接続のシナリオでのデータベース非アクティブ化動作	136
ソート・パフォーマンスのチューニング	137

データ編成	139
表の再編成	140
索引の再編成	152
表および索引を再編成するタイミングの決定	157
表および索引の再編成のコスト	161
表および索引の再編成の必要性を少なくする	162
自動再編成	164
ExampleBANK のシナリオ: 表および索引のスペースの再利用	167
アプリケーション設計	172
アプリケーションのプロセス、並行性、およびリカバリー	172
並行性の問題	174
最適なパフォーマンスを実現する照会の作成とチューニング	189
挿入パフォーマンスの向上	211
効率的な SELECT ステートメント	211
SELECT ステートメントの制限のガイドライン	213
オーバーヘッド削減のための行ブロッキングの指定	216
照会でのデータ・サンプリング	218
アプリケーションの並列処理	219
ロック管理	220
ロックおよび並行性の制御	221
ロックの細分性	222
ロック属性	223
ロックングに影響を与える要因	224
ロック・タイプの互換性	226
次キー・ロックング	227
標準の表のロック・モードおよびアクセス・プラン	227
MDC 表および ITC 表と RID 索引スキャンのロック・モード	231
MDC ブロック索引スキャンのロック・モード	236
パーティション表でのロック動作	240
ロックの変換	242
ロックの待機とタイムアウト	242
デッドロック	244
照会の最適化	246
SQL および XQuery コンパイラーの処理	246
データ・アクセス方式	271
照会の述部の処理	283
結合	286
照会最適化におけるソートとグループ化の影響	302
最適化ストラテジー	304
マテリアライズ照会表による照会最適化の改善	316
Explain 機能	318
照会アクセス・プランの最適化	371
統計ビュー	455
カタログ統計	467
RUNSTATS の影響の最小化	515

データ圧縮とパフォーマンス	516
DML のパフォーマンスを向上させるためのロギング・オーバーヘッドの低減	517
インライン LOB によってパフォーマンスを改善する	518

第 4 章 パフォーマンス・チューニング手段の設定 521

設計アドバイザー	521
設計アドバイザーの使用	525
設計アドバイザーのワークロードの定義	525
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・データベースへの変換	527
設計アドバイザーの制限と制約事項	527

第 2 部 問題のトラブルシューティングのための技法 531

第 5 章 トラブルシューティングのためのツール 535

db2cklog ツールを使用したアーカイブ・ログ・ファイルの検査	535
db2dart ツールの概要	538
INSPECT と db2dart の比較	538
db2diag ツールを使用した db2diag ログ・ファイルの分析	541
db2greg を使用したグローバル・レジストリーの表示および変更 (UNIX)	546
製品のバージョンとサービス・レベルの識別	546
db2look を使用したデータベースの模造	547
システムにインストールされている DB2 製品のリスト表示 (Linux および UNIX)	551
db2pd コマンドを使用したモニターおよびトラブルシューティング	553
db2support コマンドを使用した環境情報の収集	568
DB2 コピーの検証	573
基本的なトレース診断	573
DB2 トレース	574
DRDA トレース・ファイル	578
JDBC トレース	587
CLI トレース・ファイル	587
プラットフォーム固有のツール	603
診断ツール (Windows)	603
診断ツール (Linux および UNIX)	604

第 6 章 DB2 データベースのトラブルシューティング 607

DB2 についてのデータの収集	607
データ移動問題についてのデータの収集	608
DAS およびインスタンス管理の問題についてのデータの収集	609
特定のパフォーマンス上の問題に関する診断データの収集	610
DB2 についてのデータの分析	613

持続されたトラップのリカバリー	613
ロード操作に関する db2diag ログ項目の識別	615
管理用タスク・スケジューラーのトラブルシューティング	618
データベースが現在使用中であるために操作が失敗する	619
圧縮のトラブルシューティング	620
データ・コンプレッション・ディクショナリーが自動的に作成されない	620
行圧縮によって一時表用のディスク・ストレージ・スペースが削減されない	622
データ・レプリケーション・プロセスが、圧縮された行イメージを圧縮解除できない	623
グローバル変数問題のトラブルシューティング	626
不整合のトラブルシューティング	628
データ不整合のトラブルシューティング	628
索引とデータの不整合のトラブルシューティング	628
不整合状態にあるデータベースのトラブルシューティング	629
DB2 データベース・システムのインストールのトラブルシューティング	630
インストール問題についてのデータの収集	631
インストールの問題に関するデータの分析	632
インスタンス作成の問題に関する診断情報を収集する	633
認識済みの問題と解決策	634
ライセンス発行のトラブルシューティング	636
DB2 ライセンス準拠レポートの分析	636
ロッキング問題の診断および解決	638
ロック待機問題の診断	640
デッドロック問題の診断	643
ロック・タイムアウト問題の診断	646
ロック・エスカレーション問題の診断	649
SQL パフォーマンスのトラブルシューティング	653
個別の SQL 照会は正常に実行されるが、複数の照会の実行時にパフォーマンスが低下する	653
パフォーマンスの概要	655
ソート・パフォーマンスのチューニング	657
RUNSTATS の影響の最小化	660
データ圧縮とパフォーマンス	661
最適化のトラブルシューティング	662
最適化ガイドラインおよびプロファイルのトラブルシューティング	662
パーティション・データベース環境のトラブルシューティング	664
127.0.0.2 に関連した FCM 問題 (Linux および UNIX)	664
暗号化ファイル・システムにおけるデータベース・パーティションの作成 (AIX)	664
データ再配分時の表の状態のトラブルシューティング	665
スクリプトのトラブルシューティング	667
フィックス・パック 1 の適用後にセクション実行時統計を収集するための静的セクションの再コンパイル	667

ストレージ・キー・サポートのトラブルシューティング	668
---------------------------	-----

第 7 章 DB2 pureScale Feature のトラブルシューティング 669

DB2 pureScale Feature での問題の診断方法	669
手動によるトレースおよびログ・ファイルの収集	671
インストール、インスタンス作成、およびロールバック	673
DB2 セットアップ・ウィザードが起動しない	673
インストールおよびインスタンス作成時の問題に関する診断データの収集	673
インスタンス・リソースの作成でエラーが戻される	675
DB2 コマンドによる Tivoli SA MP の状態の調査	676
インスタンス・リソースの作成でインストールがハングする	678
DB2 pureScale Feature のホスト検証の問題に関するよくある質問	680
DB2 pureScale Feature のインストール、インスタンス作成、およびロールバックの問題に関するよくある質問	683
DB2 クラスター・ファイル・システムの問題に関するよくある質問	685
RSCT ピア・ドメインの問題に関するよくある質問	686
DB2 pureScale Feature での応答ファイル・インストールの問題に関するよくある質問	686
DB2 pureScale Feature のライセンスの問題に関するよくある質問	687
インストール後	688
インスタンスが開始しない	688
インストール後の問題に関するよくある質問	700
ホストまたはメンバーの問題	701
db2instance -list コマンドからのインスタンス状況情報の要約	702
db2cluster コマンドのトラブルシューティング・オプション	731
クラスター・マネージャー・リソース・モデルの修復	731
クラスター・マネージャー・ドメインの修復	732
正常でないホストに対応するための処理の設定	733
アンインストール	734
不完全な DB2 pureScale インスタンス・ドロップのクリーンアップ	735
IBM Reliable Scalable Cluster Technology ピア・ドメインのクリーンアップ	735
DB2 pureScale Feature のアンインストールの問題に関するよくある質問	736

第 8 章 DB2 Text Search のトラブルシューティング 739

テキスト検索操作での DB2 トレース機能の使用	739
DB2 Text Search サーバーのロギングとトレース	739
DB2 Text Search 索引更新のキューのモニター	741

DB2 Text Search のトラブルシューティングのヒント	743
----------------------------------	-----

第 9 章 トラブルシューティングについて 745

詳細情報	745
診断データ・ディレクトリー・パス	747
管理通知ログ	753
DB2 診断 (db2diag) ログ・ファイル	756
DB2 データベースの診断と OS の診断の結合	763
db2cos (コールアウト・スクリプト) 出力ファイル	766
ダンプ・ファイル	768
First Occurrence Data Capture 情報	769
内部戻りコード	784
メッセージの概要	786
プラットフォーム固有のエラー・ログ情報	788
トラップ・ファイル	793

第 10 章 知識ベースの検索 795

既知の問題の検索方法	795
トラブルシューティングのリソース	796

第 11 章 DB2 製品フィックスの入手 797

フィックスの取得	797
フィックスパック、暫定フィックスパック、およびテスト・フィックス	797
テスト・フィックスの適用	799

第 12 章 サポート 801

IBM ソフトウェア・サポートへの連絡	801
IBM との情報交換	802
サポート更新情報のサブスクライブ	804

第 3 部 付録 807

付録 A. DB2 技術情報の概説 809

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)	810
コマンド行プロセッサから SQL 状態ヘルプを表示する	812
異なるバージョンの DB2 インフォメーション・センターへのアクセス	813
コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新	813
コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新	815
DB2 チュートリアル	817
DB2 トラブルシューティング情報	817
ご利用条件	818

付録 B. 特記事項 821

索引 825

本書について

本書には、DB2[®] データベース・クライアントおよびサーバーでのデータベース・パフォーマンスのチューニングと問題の解決に関する情報を記載しています。

これによって、以下のことを行う助けが得られます。

- パフォーマンスの監視とチューニングのストラテジーを作成する
- 日常の操作のトラブルシューティング・ストラテジーを作成する
- データベース・サーバーの構成を調整する
- 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る
- 問題とエラーを簡便な方法で識別する
- 問題を症状に基づいて解決する
- 使用可能な診断ツールについて知る

本書の対象読者

本書は、DB2 データベース・クライアントおよびサーバーのデータベース・パフォーマンスのチューニングおよび問題のトラブルシューティングを行おうとしている、お客様、ユーザー、システム管理者、データベース管理者 (DBA)、通信スペシャリスト、アプリケーション開発者、およびテクニカル・サポート担当員を対象としています。本書を使用するには、以下についてよく理解している必要があります。

- 通信、リレーショナル・データベース、ローカル・エリア・ネットワーク (LAN) のそれぞれの概念
- ハードウェアおよびソフトウェアの要件とオプション
- ネットワークの全体構成
- ネットワーク上で稼働するアプリケーション・プログラムおよびその他の機能
- DB2 データベースの基本的な管理タスク
- インストール済み製品のガイド「概説およびインストール」で説明されている、インストールおよび初期作業についての情報

本書の構成

データベース・システムのパフォーマンスの監視とチューニングを支援するため、このセクションで提供される情報には、データベースのパフォーマンスに影響する要素を理解するのに必要な背景となる情報と、システムのパフォーマンスのチューニングを行うための指示が含まれます。DB2 データベース・ソフトウェアにおける問題の理解、特定、および解決を支援するため、トラブルシューティングおよびサポートの情報には、DB2 データベース製品に付属の問題判別リソースの使用に関する指示が含まれています。

第 1 部 - データベース・パフォーマンスのチューニング

データベース・アプリケーションの実行速度が遅いような気がするユーザーが報告してくる状況は、データベース管理者が直面することがあるものです。ここで提供される情報は、履歴の結果と比較してデータベース・システム・パフォーマンスの客観的評価を取得するためのパフォーマンス監視ストラテジーを作成する方法、データベース・サーバーの構成を調整する方法、およびデータベース・サーバーを使用するアプリケーションに変更を加える方法を説明します。すべての目標は、処理コストを増大させずに、またユーザーに対するサービスを低下させることなく、データベース・システムのパフォーマンスを改善することです。

- 第 1 章の『パフォーマンス・チューニングのツールと方法』は、パフォーマンスの改善を支援するためにベンチマーク・テスト・プログラムを設計し実装する方法を説明します。
- 第 2 章の『パフォーマンス・モニターのツールと方法』は、主要なシステム・パフォーマンス・データを定期的に収集する操作モニター・ストラテジーの重要性に関する情報を提供します。
- 第 3 章の『パフォーマンスに影響を及ぼす要因』には、データベース・システムのパフォーマンスに影響する種々の要因についての情報があります。これらの要因のあるものは、チューニングしたり構成しなおしたりすることが可能です。
- 第 4 章の『パフォーマンス・チューニング手段の設定』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 DB2 設計アドバイザー・ツールを説明します。

第 2 部 - 問題のトラブルシューティング

自分で問題を解決するのを支援するため、このセクションの情報は、問題の原因を識別する方法、診断情報を収集する方法、フィックスを取得できる場所、および追加情報をどの知識ベースで検索するかを説明します。IBM ソフトウェア・サポートに連絡を取る必要がある場合は、ここにサポートに連絡する方法、およびサービス技術者が問題への対処を支援するためにどの診断情報を必要とするかが説明されています。

- 第 5 章の『トラブルシューティングのためのツール』は、問題解決のための体系的なアプローチを支援するトラブルシューティング・ツールを説明します。その目標は、何かが期待したとおりに機能しない理由と問題の解決方法を判別することです。
- 第 6 章の『DB2 データベースのトラブルシューティング』は、発生する可能性のある種々の問題とそのトラブルシューティングを行う方法に関する情報を提供します。
- 第 7 章の『DB2 Connect™ のトラブルシューティング』は、発生する可能性のある種々の問題とそのトラブルシューティングを行う方法に関する情報を提供します。
- 第 8 章の『知識ベースの検索』は、IBM 知識ベースを検索することによって問題の解決方法を見つける方法について説明します。この章では、使用可能なリソース、サポート・ツール、および検索方式を使用することによって結果を最適化する方法について説明しています。
- 第 9 章の『DB2 製品フィックスの入手』は、問題の解決のために既に入手可能であるかもしれない製品フィックスを入手することに関する情報を提供します。フィックスは、ここに概説されているステップに従うことによって取得できます。

- 第 10 章の『トラブルシューティングについて』は、以下の情報が DB2 データベース・サーバーでの問題を効率的にトラブルシューティングするのに必要な概念情報を取得するのにどのように役立つかを説明します。
- 第 11 章の『IBM ソフトウェア・サポートへの連絡』は、IBM ソフトウェア・サポートに連絡する方法、および製品の不具合とデータベースの問題を解決するためにどんな情報が必要かに関する情報を提供します。

第 3 部 - 付録

- 付録 A 『DB2 技術情報の概説』
- 付録 B 『特記事項』

第 1 部 パフォーマンスの概要

パフォーマンスとは、コンピューター・システムが特定のワークロードに応答する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、およびリソースの使用率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有の程度

一般に、システムを調整することにより、その費用対効果の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
- 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・チューニングにより、リソースをより効率的に使用できたり、システムに対してより多くのユーザーを追加できたりするなどのはっきりとした効果が現れる場合があります。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には、はっきりとは分からないものもあります。

パフォーマンス・チューニングのガイドライン

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際に、以下の指針に留意してください。

- **収穫逡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。
- **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うと、実際にはそれ以降のチューニング作業が行いにくくなる可能性があります。
- **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リソース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、その変更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。パフォーマンス・チューニングでは、様々なシステム・リソース間のトレードオフが必要になります。例えば、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。
- **一度に 1 つのパラメーターを変更する:** 一度に複数の要素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。

- **レベルごとに測定と構成を行う:** 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。システム・レベルには、以下のものがあります。
 - 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 および XQuery 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。サービスをハードウェアまたはソフトウェアに適用する前に、システムのモニターとチューニングに余分な時間をかけないでください。
- **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。
- **チューニングを開始する前にフォールバック手順を適所に配置する:** チューニングをしたものの予期しない性能低下が生じる場合、別の方法を試みる前に実行した変更内容を元に戻す必要があります。元の設定を保存して、維持しない変更内容を簡単に取り消せるようにします。

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、反復的アプローチです。このパフォーマンス・モニターの結果に基づいて、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、データを使用するアプリケーションの種類に関する知識、およびデータ・アクセスのパターンについての理解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。

パフォーマンスの改善プロセスすべてには、以下の基本的なステップが含まれています。

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 つの調整を行います。

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、ハードウェアをアップグレードする時期である可能性があります。

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でない場合は、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。以下のような幾つかの簡単な質問に答えることから開始します。

- 「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？
- 特定のタイプ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されますか？
- 何らかの発生パターンがありますか？ 例えば、その問題は一日の特定の時刻に、または連続的に発生しますか？

パフォーマンス・チューニングの限界

パフォーマンス・チューニングの効果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けると必要のある時間と費用について考慮する際、余分の時間と費用をどの程度かけるとシステムのユーザーにとって役立つかを評価してください。

システムで応答時間やスループットに関する問題が生じる場合には、チューニングによってパフォーマンスが改善される場合が少なくありません。しかし、さらに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。その場合、目標と期待値を見直す必要があります。パフォーマンスをさらに大幅に改善するには、ディスク装置の増加、CPUs 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。

第 1 章 パフォーマンス・チューニングのツールと方法

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一つです。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す。

ベンチマーク・テストは、現行のパフォーマンスを調べるためにシステムに対して行うもので、それを使用してアプリケーションのパフォーマンスを向上させることができます。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る場合があります。

特定の種類の情報を得るために、さまざまなタイプのベンチマーク・テストが使用されます。例えば、

- インフラストラクチャー・ベンチマーク は、特定の限定された実験条件下でのデータベース・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、実稼働環境をさらに反映した条件下でのデータベース・マネージャーのスループット能力を調べるものです。

構成パラメーターをチューニングするためのベンチマーク・テストは、制御された条件に基づいています。そのようなテストには、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、システム構成 (およびおそらく SQL) を変えながら、繰り返しアプリケーションから SQL を実行することが関係します。

同じアプローチを、索引、表スペース構成、ハードウェア構成など、パフォーマンスに影響を与える他の要因をチューニングするためにも使用できます。

ベンチマーク・テストは、データベース・マネージャーがさまざまな条件下でどのように応答するかを理解する上で役立ちます。デッドロック処理、ユーティリティー・パフォーマンス、データをロードするさまざまな方式、ユーザーがさらに追加される場合のトランザクション率の特性、さらにはデータベース製品の新規リリースを使用したときのアプリケーションへの影響をそれぞれテストするようなシナリオを作成できます。

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。

良いベンチマークには、次のような特性があります。

- テストは反復可能である
- テストを繰り返す場合、毎回同じシステム状態で開始する
- システム内で他の機能やアプリケーションが意図せずにアクティブになることがない
- ベンチマーク・テストに使用されるハードウェアおよびソフトウェアが実稼働環境と一致している

開始されたアプリケーションは、アイドル状態であっても、メモリーを使用する点に注意してください。そのために、ページングによってベンチマークの結果がゆがめられる可能性が大きくなり、再現性基準に違反します。

ベンチマークの準備

パフォーマンスのベンチマーク・テストを開始する前に、満たしておく必要があるいくつかの前提条件があります。

パフォーマンスのベンチマーク・テストを開始する前に、以下のようにしてください。

- アプリケーションを実行する対象となるデータベースの論理設計と物理設計の両方を完成させます。
- 表、ビュー、および索引を作成します。
- 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、適切な統計が得られるようにします。
- 実動サイズのデータベースに対する実行を計画して、典型的なメモリー所要量でアプリケーションをテストできるようにします。これが不可能な場合は、データに対する使用可能システム・リソースの比率がテスト・システムと実動システムで同じになるようにしてみてください (例えば、テスト・システムでデータの 10% を使用する場合は、実動システムで使用可能なプロセッサ時間の 10% およびメモリーの 10% を使用するようにします)。
- データベース・オブジェクトを最終的なディスク・ロケーションに置き、ログ・ファイルをサイズ変更し、作業ファイルとバックアップ・イメージのロケーションを判別してバックアップ手順をテストします。
- パッケージを確認して、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実際の限界はベンチマーク・テスト時に分かりますが、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正確に反映する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サーバー・モデル上で実行するのが理想的です。アプリケーションが、最終的に多数のユーザーにサービスを提供し、大量のデータを処理する場合、このことは特に重要になります。ベンチマーク・テスト・プログラムで直接使用するオペレーティング・システムおよび通信機能またはストレージ機能も、既に調整済みでなければなりません。

ベンチマーク・テストの対象の SQL ステートメントは、典型的な SQL が最悪のケースの SQL のいずれかにする必要があります。これらについては以下のリストで説明します。

典型的な SQL

典型的な SQL には、ベンチマーク・テスト中のアプリケーションの一般的な操作で実行されるステートメントが含まれます。どのステートメントを選択するかは、アプリケーションの性質によって異なります。例えば、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 ステートメントをテストします。

最悪のケースの SQL

このカテゴリ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。例えば、顧客が電話口で待っている間に顧客情報の検索と更新を行うために実行するアプリケーションのステートメント。
- 結合の数が多すぎるステートメント、またはアプリケーションの中でも最も複雑なステートメント。例えば、ある顧客が持つ口座すべての月ごとの活動の集計を生成する銀行業務アプリケーションのステートメント。1つの共通表に顧客の住所と口座番号のリストが入っているかもしれませんが、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。
- アクセス・パスが不適切なステートメント。例えば、使用可能な索引によってサポートされないステートメント。
- 実行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、リソース要件がずば抜けて大きいステートメント。例えば、その日のうちに処理する必要がある会計作業のリストを生成するアプリケーションのステートメント。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非常に大きなリストが作成されます。このステートメントは毎日数回しか実行されないかもしれませんが、正しく調整されていないと実行に何分もかかってしまいます。

ベンチマーク・テストの作成

ベンチマーク・テスト・プログラムを設計してインプリメントする際には、さまざまな要因を考慮する必要があります。

このテスト・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ことなので、プログラムの全体的な構造はさまざまに異なります。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・テスト・プログラム

にそれらのステートメントだけを入れ、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。この時間間隔の間に実行されるステートメントの数が、アプリケーションのスループットを示す1つの尺度です。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、経過時間を測定するための信頼性のある適切な方法が必要です。個別の SQL ステートメントが分離して実行されるアプリケーションをシミュレートするには、各ステートメントの PREPARE、EXECUTE、または OPEN、FETCH、または CLOSE の時間を測定するのが最良かもしれません。他のアプリケーションの場合は、最初の SQL ステートメントから COMMIT ステートメントまでのトランザクション時間を測定するとより適切かもしれません。

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、それで必ずしもボトルネックが明らかになるわけではありません。例えば、CPU 使用率、ロッキング、およびバッファ・プール入出力に関する情報は、アプリケーションが CPU の能力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・テスト・プログラムを使用することによって、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。例えば、結果セットが別のアプリケーションへの入力になる場合があります。画面出力のデータの形式制御には、通常は高い CPU コストが必要となり、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・テスト・プログラムがアプリケーションの特定の行処理アクティビティ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 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。

非常に使いやすいとはいえ、DB2 コマンド行プロセッサ (CLP) はベンチマークに適していません。これによる処理オーバーヘッドが加わるからです。ベンチマーク・ツール (**db2batch**) は、インスタンスの `sql1lib` 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、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、それらのステートメントを動的に作成および実行し、結果セットを返すことができます。**db2batch** に返される行数や表示される行数を制御することもできます。さらに、経過時間、プロセッサ時間、バッファ・プールの使用率、ロッキング、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、返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。

Perl または Korn シェル・スクリプトで **db2batch** 呼び出しをラップすることにより、マルチユーザー環境を容易にシミュレートできます。適切な **db2batch** オプションを選択することにより、分離レベルなどの接続属性が同じであることを確認してください。

パーティション・データベース環境では、**db2batch** が適しているのは経過時間の測定のみであることに注意してください。戻されるその他の情報は、コーディネーター・データベース・パーティション上のアクティビティーにのみ関連しています。

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。Linux または UNIX システムでは、シェル・プログラムを使用してドライバー・プログラムを作成できます。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、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復元し、新しいパラメーター値を使って次のテストを設定し、テスト結果を収集および統合することができます。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、所定のテストでの最適なパラメーター値のレポートを作成することができます。

ベンチマーク・テストの実行

最も一般的なタイプのベンチマーク・テストでは、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。

1 回のテストには、同じパラメーター値を使用してアプリケーションを繰り返し実行する (例えば 5 または 10 回) ことが含まれます。これにより、他のパラメーター値による結果と比較して、より信頼できる平均パフォーマンス値を取得できます。

最初の実行 (ウォームアップ実行と呼ばれる) を、後続の実行 (通常実行と呼ばれる) とは分けて考える必要があります。ウォームアップ実行にはバッファ・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、完了までに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、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値を計算するときは、通常実行の結果のみを使用してください。平均を計算する前に、高い値と低い値を除去しておくことは、たいいてい得策です。

実行相互間で最良の整合性を得るには、新しい実行のたびにあらかじめバッファ・プールを既知の状態に戻しておくようにします。テストでは、バッファ・プールにデータがロードされるようにすることができます。こうすると必要なディスク入出力が減るので、後続の実行の速度を上げることができます。バッファ・プールの内容は、無関係な他のデータをバッファ・プールに読み込むか、すべてのデータベース接続が一時的に除去されたときにバッファ・プールの割り振りを解除することにより、強制的に排除できます。

パラメーター値の単一の集合でのテストを完了したら、1 つのパラメーターの値を変更することができます。1 回テストしてから次に繰り返して行うまでの間に、以下に示すタスクを実行して、ベンチマーク環境を元の状態に復元してください。

- 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・データがテスト中に更新される場合、それを一貫性のあるものにする必要があります。そのためには、次のようにします。

- リストア・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、次のテストのために用意が整った状態になります。
- インポートまたはロード・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` ユーティリティーおよび `runstats` ユーティリティーを、このデータが入った表と索引に対して実行する必要があります。

以下に、データベース・アプリケーションのベンチマーク・テストのために行うステップの要約を示します。

ステップ 1

DB2 レジストリー、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、バッファー・プールは、それぞれ標準推奨値のままにしておきます。これに入る値には以下のものが考えられます。

- アプリケーションをエラー・フリーで適切に実行するために必要とされることが分かっている値。
- 前のチューニングでパフォーマンスを改善できた値。
- **AUTOCONFIGURE** コマンドによって提示された値。
- デフォルト値。ただし、以下については該当しない場合があります。
 - ワークロードにとって、およびテストの目標にとって重要なパラメーター。
 - 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るログ・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

この初期ケースに関して一連の繰り返しを実行し、平均経過時間、スループット、またはプロセッサ時間を計算します。できるだけ結果が整合するようにします。実行相互間で違いが数ポイント以下に収まるのが理想です。実行相互間でパフォーマンス測定が大きく変わると、チューニングが非常に難しくなるおそれがあります。

ステップ 2

テストする方法または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

再び一連の繰り返しを実行し、その平均経過時間、またはプロセッサ時間を計算します。

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。

ベンチマーク・テストの分析例

ベンチマーク・テスト・プログラムの出力には、各テストの ID、繰り返し数、ステートメント番号、およびそれぞれの実行の経過時間を含めるようにする必要があります。

これらのサンプル・レポートで示されているデータは、例として示す目的でのみ使用されています。実際に測定された結果を表すものではありません。

ベンチマーク・テストの結果のサマリーは、次のように表示されます。

Test Numb	Iter. Numb	Stmt Numb	Timing (hh:mm:ss.ss)	SQL Statement
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 1. ベンチマーク・テストの結果の例

これを分析すると、CONNECT (ステートメント 01) の完了までに 1.34 秒、OPEN CURSOR (ステートメント 10) に 2 分 8.15 秒かかり、FETCH (ステートメント 15) が 7 行を戻してその最大の遅延が 0.28 秒であり、CLOSE CURSOR (ステートメント 20) に 0.84 秒、および CONNECT RESET (ステートメント 99) の完了までに 0.03 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、後でデータ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・レポートのサマリーは、次のように表示されます。

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 2. ベンチマークのタイミング・レポートの例

第 2 章 パフォーマンス・モニターのツールと方法

システム・パフォーマンスの操作モニター

操作モニターとは、長期に渡り定期的な間隔でキー・システム・パフォーマンス・メトリックを収集することを指します。この情報により、初期構成を自分の要件により適合するように洗練するために重要なデータが得られ、ソフトウェア自体やアップグレード後に生じる新たな問題や、データ・ボリュームまたはユーザー数の増加に対応したり、新しいアプリケーション・デプロイメントに対処したりする準備を整えることにもなります。

操作モニターの考慮事項

操作モニター・ストラテジーは、いくつかの考慮事項に対応している必要があります。

操作モニターは非常に軽量で (測定対象のシステムの多くの部分を消費しない)、かつ汎用である (システム内のいずれかの箇所で生じる可能性のある問題に対する「視野」を広く保てる) ことが必要です。

システムの存続期間中に操作メトリックを定期的に収集することを計画しているので、そうしたデータすべてを管理する手段を備えておくのは大切です。ご使用のデータに関して可能性のある多くの使用法 (例えば、パフォーマンスの長期的傾向など) を考えると、別々のかなり多くの月のデータを任意に収集し、それらを比較できることが望まれます。DB2 製品は、こうした種類のデータ管理をととても容易に行えます。モニター・データの分析と比較はとても簡単になり、長期に渡るデータ・ストレージと編成に関する堅固なインフラストラクチャーが既にきちんと整っています。

DB2 データベース (「DB2」) システムには、データをモニターするための優れたソースがいくつか備わっています。主要なものとしてはスナップショット・モニター、および DB2 バージョン 9.5 以降ではデータ集約のためのワークロード管理 (WLM) 表関数があります。どちらもサマリー・データに焦点を当てており、カウンター、タイマー、およびヒストグラムのようなこれらのツールは、システム内のアクティビティー全体の実行状況を維持します。長期に渡りこうしたモニター・エレメントをサンプリングすると、開始時と終了時の間に生じた平均アクティビティーを導出でき、これはとても参考となる情報となり得ます。

DB2 製品によって提供されるメトリックだけに限定する必要はありません。実際、DB2 ソフトウェアの外部のデータは、単に「あればよい」というだけのものではありません。パフォーマンス上の問題判別では、コンテキスト情報がかぎとなります。ユーザー、アプリケーション、オペレーティング・システム、ストレージ・サブシステム、およびネットワーク、これらすべてはシステム・パフォーマンスに関する有用な情報源となり得ます。システム・パフォーマンスに関する完全な全体像を得るには、DB2 データベース・ソフトウェア以外のメトリックを含めることが重要な部分を成します。

DB2 データベース製品の最近のリリースの傾向では、SQL インターフェースを介してモニター・データをより活用できるようになっています。これにより、DB2 を使用したモニター・データの管理がとても簡単になっています。管理ビューからのデータを、DB2 表などに簡単にリダイレクトできるからです。さらに、アクティビティ・イベント・モニターのデータも DB2 表に書き込むことができ、それにより同じような利点があります。モニター・データの大部分は簡単に DB2 に格納できるので、システム・メトリック (**vmstat** からの CPU 使用率など) を DB2 に格納するための労力もごくわずかで扱いやすくなっています。

操作モニター用に収集するデータのタイプ

継続的な操作モニターのために、収集すると便利なタイプのデータがいくつかあります。

- DB2 システム・パフォーマンス・モニター・メトリックの基本セット
- DB2 構成情報

定期的にデータベース、データベース・マネージャーの構成、DB2 レジストリー変数、およびスキーマ定義のコピーを作成することは、加えた変更の履歴を提供し、モニター・データに現れる変更を説明するのに役立ちます。

- 全体的なシステム負荷

CPU または入出力の使用効率が飽和状態に近づくのを許すと、システム・ボトルネックが生じる可能性があります。DB2 スナップショットだけを使用してそれを検出するのが難しい場合があります。結果として、Linux および UNIX では **vmstat** と **iostat** (ネットワーク問題には、おそらく **netstat** も) を使用し、Windows 上では **perfmon** を使用して、定期的にシステム負荷をモニターするのがベスト・プラクティスです。また **ENV_GET_SYSTEM_RESOURCES** などの管理ビューを使用して、オペレーティング・システム、CPU、メモリー、およびシステムに関連するその他の情報を取得できます。通常、注意を向けるのは、特定の万能値ではなく、ご使用のシステムで標準となる変更値です。

- ビジネス・ロジック・レベルで測定するスループットと応答時間

ビジネス・ロジック・レベルで DB2 について測定する、パフォーマンスのアプリケーション・ビューにはほとんどのエンド・ユーザーに関連する利点があり、通常、これには表示ロジック、アプリケーション・サーバー、Web サーバー、複数のネットワーク・レイヤーなどのボトルネックを生じさせる可能性のあるすべての対象が含まれます。このデータは、サービス・レベルの合意 (SLA) の設定または検証のプロセスに不可欠です。

DB2 システム・パフォーマンス・モニター・エレメントとシステム負荷データは 5 分から 15 分ごとに収集される場合であっても十分にコンパクトなため、長期に渡る合計データ・ボリュームはほとんどのシステムで問題となりません。同様に、このデータの収集によるオーバーヘッドによって CPU 使用量が 1 から 3 パーセントの範囲で通常は増えますが、重要なシステム・メトリックの履歴を継続的に取得するために払うコストとしては小さいと言えます。構成情報に関しては一般的に変更は比較的まれなので、過剰なデータを生成することなく収集するには、1 日に一度が通常は適度な頻度です。

システム・パフォーマンスのモニター・エレメントの基本セット

システム・パフォーマンスの 11 個のメトリックは、継続的な操作モニターの取り組みで使用する良好な基本セットを提供します。

選択肢となるメトリックは数多くありますが、生成されるデータは莫大な量に及ぶので、すべてのデータを収集することは非生産的なことになり得ます。以下に当てはまるメトリックが適しています。

- 収集が簡単 – 毎日モニターするために複雑なツールやコストの高いツールを使用する必要はありませんし、モニターすることによってシステムに多大な負荷をかけたくもありません。
- 簡単に理解できる – 参照するごとにメトリックの意味を検索したくはありません。
- ご使用のシステムに該当する – すべてのメトリックがすべての環境において有意義な情報を提供するわけではありません。
- 感度が良いが、過敏ではない – メトリックの変化は実際のシステム内の変化を示す必要がありますが、メトリック自体が不安定になるべきではありません。

このスターター・セットには、以下に示す 11 個のメトリックが含まれています。

1. 実行されたトランザクションの数:

TOTAL_COMMITS

これにより、システム・アクティビティの基本レベルの優れた測定結果が提供されます。

2. データ、索引、XML ストレージ・オブジェクト、および一時データごとに測定されたバッファー・プール・ヒット率は、以下のようになります。

注: 以下で取り上げられているのは、DB2 pureScale®環境以外の環境におけるバッファー・プールについての説明です。DB2 pureScale環境におけるバッファー・プールの動作は異なります。詳しくは、「データベースのモニタリングガイドおよびリファレンス」の『DB2 pureScale 環境におけるバッファー・プールのモニター』を参照してください。

- データ・ページ: $(\text{pool_data_lbp_pages_found} - \text{pool_async_data_lbp_pages_found} - \text{pool_temp_data_l_reads}) / \text{pool_data_l_reads} \times 100$
- 索引ページ: $((\text{pool_index_lbp_pages_found} - \text{pool_async_index_lbp_pages_found} - \text{pool_temp_index_l_reads}) / \text{pool_index_l_reads}) \times 100$
- XML ストレージ・オブジェクト (XDA) ページ: $((\text{pool_xda_gbp_l_reads} - \text{pool_xda_gbp_p_reads}) / \text{pool_xda_gbp_l_reads}) \times 100$
- 一時データ・ページ: $((\text{pool_temp_data_l_reads} - \text{pool_temp_data_p_reads}) / \text{pool_temp_data_l_reads}) \times 100$
- 一時索引ページ: $((\text{pool_temp_index_l_reads} - \text{pool_temp_index_p_reads}) / \text{pool_temp_index_l_reads}) \times 100$

バッファー・プール・ヒット率は最も基本的なメトリックの 1 つで、ディスク入出力を回避するためにシステムがメモリーをどれほど効率的に活用しているかについての重要な全体的指標です。OLTP 環境では、データの場合はヒット

率が 80 から 85% 以上、索引の場合は 90 から 95% 以上が通常は良好であると見なされます。言うまでもなく、こうした比率は、バッファー・プール・スナップショットからのデータを使用して個別のバッファー・プールに関して計算できます。

注: データおよび索引ページのヒット率を求めるために示される公式では、プリフェッチャーによる読み取りアクティビティは除外されています。

大規模な表スキャンを頻繁に実行するデータウェアハウスなどのシステムではこうしたメトリックは一般的に役立ちますが、データ・ヒット率が著しく低くなることもしばしばあります。その原因は、データがバッファー・プールで読み取られた後、再び使用されないまま、他のデータの余地を生み出すために追い出されるからです。

3. 以下のようにして、バッファー・プールはトランザクションごとに物理的に読み取りと書き込みを行います。

注: 以下で取り上げられているのは、DB2 pureScale環境以外の環境におけるバッファー・プールについての説明です。DB2 pureScale環境におけるバッファー・プールの動作は異なります。詳しくは、「データベースのモニタリングガイドおよびリファレンス」の『DB2 pureScale 環境におけるバッファー・プールのモニター』を参照してください。

$$\frac{(\text{POOL_DATA_P_READS} + \text{POOL_INDEX_P_READS} + \text{POOL_XDA_P_READS} + \text{POOL_TEMP_DATA_P_READS} + \text{POOL_TEMP_INDEX_P_READS})}{\text{TOTAL_COMMITTS}}$$
$$\frac{(\text{POOL_DATA_WRITES} + \text{POOL_INDEX_WRITES} + \text{POOL_XDA_WRITES})}{\text{TOTAL_COMMITTS}}$$

こうしたメトリックはバッファー・プール・ヒット率と密接に関連していますが、若干目的が異なります。ヒット率のターゲット値を考慮できますが、トランザクションごとに読み取りと書き込みのターゲットを定めることはできません。こうした計算が難しいのはなぜでしょうか。データベース・パフォーマンスにおいてディスク入出力は主要な要因ですので、それを調べる手段が複数あると便利です。また、ヒット率は読み取りしか扱いませんが、こうした計算には書き込みも関係します。また最後に別の点として、例えば索引ヒット率が 94% の場合、改善しようとする価値があるのかどうかを把握するのは困難です。1 時間に 100 だけの論理索引読み取りがあり、バッファー・プールでは 94 の場合、チューニングによって最後の 6 を物理読み取りに維持しようとするのは時間の無駄になります。しかし、94% の索引ヒット率が、それぞれが 20 の物理読み取りを行ったトランザクションの統計によるものである場合 (データと索引、正規と一時にさらに分類できます)、バッファー・プール・ヒット率を少し調べてみる価値はあります。

このメトリックは単に物理的な読み取りと書き込みではなく、トランザクションごとに正規化されています。この傾向は、多くのメトリックにも当てはまります。目的は、データが収集された期間の長さによって、またそのときにシステムが非常にビジーな状態だったかあまりビジーではなかったかによって、メトリックを分離することにあります。このようにすると、一般に、モニター・データの収集方法と時期にかかわらず、メトリックに関して同様の値を取得するのに役立ちます。データ収集のタイミングと期間に関してある程度の一貫性

を持たせるのは良いことですが、正規化してしまうと、良いアイデアを得ることはできても重要な考察を得るのが難しくなります。

4. 選択行数に対するデータベース行読み取りの比率は、以下のようになります。

$ROWS_READ / ROWS_RETURNED$

この計算により、適格な行を検出するために、データベース表から読み取られる行の平均数が得られます。この数値が低いと、データの位置指定が効率的で、索引が効率的に使用されていることを一般に示します。例えば、システムで行う表スキャンが多く、大量の行を検査してそれらが結果セットに適格かどうかを判別する必要がある場合には、この数値がとて大きくなる可能性があります。その代わりに、完全修飾されたユニーク索引を使用して表にアクセスする場合には、この統計が非常に小さくなり得ます。索引専用のアクセス・プラン (表から行を読み取る必要がない) では、 $ROWS_READ$ が増加することはありません。

OLTP 環境では、通常このメトリックが 2 または 3 より大きくなることはなく、ほとんどのアクセスでは表スキャンではなく索引スキャンが使用されます。このメトリックは長期に渡りプランの安定度をモニターする簡単な方法の 1 つであり、予期しない増加が生じるときには索引がもう使用されていないので調査が必要なことを多くの場合に示します。

5. トランザクションごとにソートに費やされた時間合計は、以下のようになります。

$TOTAL_SORT_TIME / TOTAL_COMMITTS$

これは、ソート統計を処理する効率的な方法です。書き出されたソートによって生じる余分の時間が自動的に含まれるからです。このため、特にご使用のシステムでソートに関する問題の履歴がある場合には、 $TOTAL_SORTS$ および $SORT_OVERFLOWS$ を収集して分析を簡単にすることもできます。

6. 1000 トランザクションごとに集計されたロック待機時間合計は、以下のようになります。

$1000 * LOCK_WAIT_TIME / TOTAL_COMMITTS$

ロック待機時間が長すぎると応答時間が多くの場合に長くなるので、モニターすることが重要です。1 つのトランザクションにおけるロック待機時間は通常とても短いので、この値は 1000 トランザクションに正規化されます。1000 トランザクションにスケール・アップすると、処理しやすい大きさになります。

7. 1000 トランザクションごとのデッドロックとロック・タイムアウト数は、以下のようになります。

$1000 * (DEADLOCKS + LOCK_TIMEOUTS) / TOTAL_COMMITTS$

ほとんどの実動システムでデッドロックは比較的まれなことです。ロック・タイムアウトはもっと一般的に生じます。通常、アプリケーションではこれらは同様の仕方で扱う必要があります。最初からトランザクションを再実行しなければなりません。これが生じる比率をモニターすると、DBA が認識することなく、数多くのデッドロックまたはロック・タイムアウトがかなり余分の負荷をシステムに生じさせるのを回避するのに役立ちます。

8. 1000 トランザクションごとのダーティ・スチール・トリガー数は、以下のようになります。

$1000 * POOL_DRTY_PG_STEAL_CLNS / TOTAL_COMMITTS$

『ダーティ・スチール』は、バッファ・プール・クリーニングをトリガーするための優先度の最も低い方法です。基本的に、スワップアウトされるページの更新がディスクに書き込まれる際に、新しいバッファ・プール・ページを必要とする SQL ステートメントの処理が中断されます。ダーティ・スチールが頻繁に生じることが許可される場合、スループットと応答時間にかなりの影響を与える可能性があります。

9. 1000 トランザクションごとのパッケージ・キャッシュ挿入数は、以下のようになります。

$1000 * PKG_CACHE_INSERTS / TOTAL_COMMITTS$

パッケージ・キャッシュ挿入は、システムの通常実行の一部ですが、数値が大きい場合には、CPU 時間のかかなりのコンシューマーであることを示す場合があります。適正に設計されている多くのシステムでは、定常状態でシステムが実行された後に生じるパッケージ・キャッシュ挿入はごく少数です。なぜなら、システムで使用または再使用するの静的 SQL ステートメントか、事前に準備された動的 SQL ステートメントだからです。特別な動的 SQL ステートメントのトラフィックが多いシステムでは、SQL コンパイルおよびパッケージ・キャッシュ挿入を回避することは不可能です。ただし、このメトリックは 3 番目のタイプの状態、つまり準備済みのステートメントを再使用していなかったり、頻繁に実行される SQL でパラメーター・マーカーを使用しなかったりする場合に、意図せずにパッケージ・キャッシュの大きな変動がアプリケーションで生じるという事態を監視することも目的としています。

10. ログ・レコードがディスクにフラッシュされるのをエージェントが待機する時間:

$LOG_WRITE_TIME / TOTAL_COMMITTS$

トランザクション・ログがシステム・ボトルネックとなるかなりの可能性があります。その場合の原因はアクティビティーのレベルが高いか、構成が正しくないか、または他の原因が考えられます。ログ・アクティビティーをモニターすると、DB2 側の問題 (アプリケーション主導によるログ要求数の増加を意味する) とシステム側の問題 (多くの場合、ハードウェアまたは構成の問題によって生じるログ・サブシステム・パフォーマンスの低下が原因) を検出できます。

11. パーティション・データベース環境においてパーティション間で送受信される高速コミュニケーション・マネージャー (FCM) バッファ数、以下のようになります。

$FCM_SENDS_TOTAL, FCM_RECVS_TOTAL$

これにより、クラスター内の異なるパーティション間のデータのフローの率、特にフローの均衡が取れているかどうかを示されます。種々のパーティションから受け取るバッファ数がかかなり異なる場合、各パーティションにハッシュされたデータ量に偏りがあることを示すことがあります。

パーティション・データベース環境でのパーティション間モニター

前述のほとんどすべての個々のモニター・エレメント値は、パーティションごとにレポートされます。

一般に、同一の DB2 パーティション・グループ内のすべてのパーティションでは、ほとんどのモニター統計がかなり同様であると予想します。かなり違いがある場合には、データの偏りを示すことがあります。トラッキングするパーティション間比較のサンプルには、以下のものがあります。

- データ、索引、および一時表のための論理および物理バッファ・プールの読み取り
- パーティション・レベルにおける大きな表の行読み取り
- ソート時間とソート・オーバーフロー
- FCM バッファの送受信
- CPU と入出力の使用率

モニター・データ内の異常値

異常値を識別できることは、パフォーマンス上の問題のトラブルシューティング時に、システム・パフォーマンスのモニター・データを解釈するための鍵です。

モニター・エレメントは、その値が通常より悪いとき（つまり、値が異常なとき）、パフォーマンス上の問題の性質を示す手掛かりとなります。一般的に、より悪い値とは、期待された値より高い値（例えば、より高いロック待機時間）のことです。しかし、異常値は、より低いバッファ・プール・ヒット率などのように、期待された値より低くなることもあります。状態に応じて、1 つ以上の方法を使用して、値が正常より悪いかどうかを判別できます。

1 つの手法は、業界の経験法則またはベスト・プラクティスに依存することです。例えば、経験法則では、データの 80 から 85% 以上のバッファ・プール・ヒット率が、一般的に OLTP 環境に良いと考えられています。この経験法則は OLTP 環境に当てはまるものであり、システムの性質上、データ・ヒット率が頻繁にかなり低くなるデータウェアハウスの有用な指針とはならないことに注意してください。

別の手法は、現行値を以前に収集されたベースライン値と比較することです。この手法は、しばしば最も信頼の置けるものであり、正常な状態の間にキー・パフォーマンス・メトリックを収集および保管する十分な操作モニター・ストラテジーがあることに依存しています。例えば、現行のバッファ・プール・ヒット率が 85% であることに気付く場合があります。これは、業界の水準に従えば正常であると考えられますが、パフォーマンス上の問題がレポートされる前に記録された 99% という値と比較するとき、異常であると考えられます。

最後の手法は、現行値を比較可能なシステム上の現行値と比較することです。例えば、比較可能なシステムのバッファ・プール・ヒット率が 99% の場合、85% の現行のバッファ・プール・ヒット率は、異常であると考えられます。

ガバナー・ユーティリティ

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応じてその動作を変更することができます。

重要: DB2 バージョン 9.5 で新しい DB2 ワークロード・マネージャーの戦略的機能が導入されたことで、バージョン 9.7 では DB2 ガバナー・ユーティリティは推奨されなくなりました。今後のリリースでは除去される可能性があります。ガバナー・ユーティリティの非推奨についての詳細は、『DB2 ガバナーおよび Query Patroller の非推奨』を参照してください。DB2 ワークロード・マネージャーについて、およびそれをガバナー・ユーティリティと置き換える方法について詳しくは、『DB2 ワークロード・マネージャーの概念の紹介』および『DB2 ワークロード管理に関してよくある質問』を参照してください。

ガバナー・インスタンスは、フロントエンド・ユーティリティおよび 1 つ以上のデーモンで構成されています。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、デフォルトでは、ガバナーを開始すると、パーティション・データベースの各データベース・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、モニターする単一のデータベース・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。

ガバナーは、ガバナー構成ファイルの規則に従って、アプリケーション・トランザクションを管理します。例えば、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分かったとします。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、取るべき処置も指定します。

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。

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。

ガバナーの開始

ガバナー・ユーティリティは、データベースに接続された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。

始める前に

ガバナーを開始する前に、ガバナー構成ファイルを作成する必要があります。

ガバナーを開始するには、SYSADM または SYSCTRL 権限を持っていないければなりません。

このタスクについて

重要: DB2 バージョン 9.5 で導入されたワークロード管理フィーチャーにより、DB2 ガバナー・ユーティリティーはバージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。これはDB2 pureScale環境ではサポートされていません。詳しくは、『DB2 ガバナーと Query Patroller が推奨されなくなった』(<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0054901.html>) を参照してください。

手順

- ガバナーを開始するには、以下の必須パラメーターを指定して、**db2gov** コマンドを使用します。

START *database-name*

指定するデータベース名は、ガバナー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。

config-file

このデータベースにおけるガバナー構成ファイルの名前。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、ファイル名と共にファイル・パスも含める必要があります。

log-file

このガバナーに対するログ・ファイルの基底名。パーティション・データベースの場合、ガバナーのこのインスタンスのためにデーモンが実行されているデータベース・パーティションごとに、データベース・パーティション番号が付加されます。

- パーティション・データベースにおける単一のデータベース・パーティションでガバナーを開始するには、**dbpartitionnum** オプションを指定してください。

例えば、SALES というデータベースのデータベース・パーティション 3 でガバナーを開始するには、**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、以下のコマンドを入力します。

```
db2gov start sales dbpartitionnum 3 salescfg saleslog
```

- すべてのデータベース・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
```

ガバナー・デーモン

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。

ガバナー・デーモンは、その開始時に必ず以下のタスク・ループを実行します。

- デーモンは、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. デーモンは、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。
3. デーモンは、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の場合、ガバナーは指定された処置を実行します。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規則が、次のガバナーのインターバルの際にアプリケーションに適用されることを意味します。
4. デーモンは、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。

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停止します。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。

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の値がシステム・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。

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。

ガバナー構成ファイル

ガバナー構成ファイルには、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が含まれています。

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。

ガバナー構成ファイルは、モニターするデータベース (必須)、CPU 使用量統計が格納されるアカウント・レコードに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一般節で構成されています。構成ファイルには、オプションでアプリケーション・モニターの規則ステートメントが 1 つ以上含まれる場合もあります。以下の指針は、一般節と規則ステートメントの両方に適用されます。

- 一般的なコメントは中括弧 ({ }) に入れて区切る。

- ほとんどの場合、値は大文字、小文字、または大/小文字混合で指定する。例外はアプリケーション名で (`applname` 節に続いて指定される)、これは大/小文字の区別があります。
- 一般節または規則ステートメントは、それぞれセミコロン (;) で終了する。

規則を更新する必要がある場合には、ガバナーを停止しないで構成ファイルを編集します。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。

パーティション・データベース環境では、ガバナー構成ファイルは、各データベース・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・パーティションにマウントされるディレクトリに作成する必要があります。

一般節

ガバナー構成ファイルには、以下の節は複数回指定できません。

dbname

モニター対象となるデータベースの名前または別名。この節は必須です。

account *n*

各接続ごとの CPU 使用量統計を格納するアカウント・レコードが書き込まれるインターバル (分単位)。このオプションは、Windows オペレーティング・システムでは使用できません。一部のプラットフォームでは、CPU 統計はスナップショット・モニターから使用できません。これに該当する場合、`account` 節は無視されます。

`account` のインターバル未満で完結する短いセッションが発生した場合、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開した場合、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 *n*

デーモンがウェイクアップする時間間隔 (秒単位)。この節を指定しない場合、デフォルト値の 120 秒が使用されます。

規則節

規則ステートメントは、アプリケーションの管理方法を指定します。これらのステートメントは、規則節と呼ばれるより小さな構成要素で構成されます。規則節を使用する場合、次のように、規則ステートメントに特定の順序で出現する必要があります。

1. **desc:** 規則に関するコメント。単一引用符または二重引用符で囲みます。
2. **time:** 規則が評価される時刻。
3. **authid:** アプリケーションがステートメントを実行するときの 1 つ以上の許可 ID。

4. **aplname:** データベースに接続する実行可能ファイル、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、名前は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。例えば、CPU 時間、返される行の数、アイドル時間などがあります。一部のプラットフォームでは、CPU 統計はスナップショット・モニターから使用できません。これに該当する場合、setlimit 節は無視されます。
6. **action:** 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して実行できる処置には、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などが含まれます。

規則節を組み合わせて 1 つの規則ステートメントを作ります。各節は規則ステートメントごとに 1 回までしか使用できません。

```
desc "Allow no UOW to run for more than an hour"  
setlimit uowtime 3600 action force;
```

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。規則節に -1 の値を指定した場合は例外が発生します。この場合、同じ節の後続の指定値は、それより前の指定値のみをオーバーライドできます。前に置かれた規則ステートメントの他の節は、有効なままになります。

例えば、ある規則ステートメントが `rowsel 100000` および `uowtime 3600` 節を使用して、経過時間が 1 時間を超えた場合、または選択された行が 100 000 行を超えた場合には、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` 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、`rowsel 100000` がまだ適用されるために、優先順位が下げられます。

規則適用の順序

ガバナーは、規則を構成ファイル内の最初から最後まで処理します。ただし、ある規則ステートメントの `setlimit` 節がその前にある規則ステートメントにある同じ節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。以下の例では、ADMIN の制限は 5000 行のままです。最初の規則の方がより制限的であるためです。

```
desc "Force anyone who selects 5000 or more rows."  
setlimit rowsel 5000 action force;
```

```
desc "Allow user admin to select more rows."  
authid admin setlimit rowsel 10000 action force;
```

緩やかな規則で、それより前のより制限的な規則をオーバーライドするには、-1 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。例えば、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 who selects 5000 or more rows."  
setlimit rowsel 5000 action force;
```

```
desc "Clear the rowsel limit for admin."  
authid admin setlimit rowsel -1;
```

```
desc "Now set the higher rowsel limit for admin"  
authid admin setlimit rowsel 10000 action force;
```

ガバナー構成ファイルの例

```
{ The database name is SAMPLE; do accounting every 30 minutes;  
  wake up once a second. }  
dbname sample; account 30; interval 1;
```

```
desc "CPU restrictions apply to everyone 24 hours a day."  
setlimit cpu 600 rowsel 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 rowsel 500 rowsread 5000;
```

```
desc "Have the 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 department."  
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 the DB2 CLP by the novice user."  
authid novice  
applname db2bp.exe  
setlimit cpu 5 locks 100 rowsel 250;
```

```
desc "During the day, 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 the lunch hour."  
time 12:00 13:00 authid ming, geoffrey, john, bill  
applname tpcc1, tpcc2, tpcA, tpcG  
setlimit cpu 600 rowsel 120000 action force;
```

```
desc "Increase the priority of an important application so it always  
  completes quickly."  
applname V1app setlimit cpu 1 locks 1 rowsel 1 action priority -20;
```

```
desc "Some people, such as the database administrator (and others),  
  should not be limited. Because this is the last specification  
  in the file, it will override what came before."  
authid gene, hershel, janet setlimit cpu -1 locks -1 rowsel -1 uowtime -1;
```

ガバナーの規則節

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りられる処置を指定する節から構成されています。

規則節は、以下に説明する順序で指定する必要があります。

オプションの先頭節

desc 規則に関する記述を指定します。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 (例えば、time 8:00 18:00) という形式で指定する必要があります。この節が指定されない場合、規則は全日 (24 時間) にわたって評価されます。

authid アプリケーションを実行する許可 ID を 1 つまたは複数指定します。複数の許可 ID を指定する場合は、例えば authid gene, michael, james のように、コンマ(,) で区切る必要があります。この節が指定されない場合、規則はすべての許可 ID 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、例えば applname db2bp, batch, geneprog のように、コンマ(,) で区切る必要があります。この節が指定されない場合、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。管理する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。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。

制限節

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 (例えば、cpu -1 locks 1000 rowsse1 10000)。制限は少なくとも 1 つは指定する必要があります。規則ステートメントに指定されていない制限は、そ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 *n* 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、アプリケーションの CPU 使用は制限されません。

idle *n* 接続において、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。

注: バックアップおよびリストアなどの一部のデータベース・ユーティリティでは、データベースへの接続を確立し、次いでガバナーからは可視でない作業をエンジン・ディスパッチ可能単位 (EDU)

により実行します。これらのデータベース接続はアイドルと表示され、アイドル時間制限を超過する場合があります。ガバナーがこれらのユーティリティに対してアクションを取らないようにするために、呼び出す許可 ID から、それらに -1 を指定します。例えば、ガバナーが許可 ID の DB2SYS で実行しているユーティリティに対してアクションを取らないようにするには、「authid DB2SYS setlimit idle -1」と指定します。

locks *n*

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、アプリケーションが保持するロック数は制限されません。

rowsread *n*

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。指定できる最大値は、4 294 967 298 です。

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` が結果セットを返すために読み取る必要のある行数である点です。この数にはエンジンによるカタログ表の読み取りが含まれます。この数は索引使用時には少なくなる可能性があります。

rowssel *n*

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・データベース・パーティションのみでゼロ以外になります。-1 を指定すると、返すことができる行数は制限されません。指定できる最大値は、4 294 967 298 です。

uowtime *n*

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` API を使用して作業単位モニター・スイッチまたはタイム・スタンプ・モニター・スイッチを非アクティブ化した場合には、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。ガバナーが開始されるより前にアプリケーションの作業単位 (UOW) が開始されると、ガバナーはその UOW を管理しません。

処置節

action 指定された 1 つ以上の制限を超えた場合に取りる処置を指定します。制限を超えたときに **action** 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。

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終了を指定します。(**FORCE APPLICATION** コマンドを使用するとコーディネーター・エージェントが終了します。)

注: パーティション・データベース環境では、`force` アクションは、アプリケーションのコーディネーター・データベース・パーテ

イション上でガバナー・デーモンが実行されている場合のみ実行されます。したがって、データベース・パーティション A 上でガバナー・デーモンが実行しているときに、コーディネーター・データベース・パーティションがデータベース・パーティション B であるアプリケーションの限度を超えた場合には、force アクションはスキップされます。

nice n アプリケーションの処理を行っているエージェントの相対的な優先順位の変更を指定します。Linux および UNIX では、有効値の範囲は -20 から +20 であり、Windows プラットフォームでは、-1 から 6 です。

- Linux および UNIX では、**agentpri** データベース・マネージャー構成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 nice 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・マネージャー構成パラメーターと nice の値を一緒に使用できます。

ガバナーを使用して、デフォルトのユーザー・サービス・スーパークラス、SYSDEFAULTUSERCLASS で実行するアプリケーションの優先順位を制御できます。ガバナーを使用して、このサービス・スーパークラスで実行するアプリケーションの優先順位を低くする場合、エージェントは自分自身をそのアウトバウンド相関関係子から切断し (両者が関連付けられている場合)、ガバナーによって指定されたエージェント優先順位に従ってその相対優先順位を設定します。ガバナーを使用して、ユーザー定義サービスのスーパークラスおよびサブクラスでエージェント優先順位を変更することはできません。その代わりに、スーパークラスまたはサブクラスのエージェント優先順位の設定を使用して、それらのサービス・クラスで実行するアプリケーションを制御する必要があります。一方で、ガバナーを使用してサービス・クラスでの接続を強制することができます。

注: AIX® システムでは、アプリケーションの処理を行っているエージェントの相対的な優先順位を上げるには、インスタンス所有者は CAP_NUMA_ATTACH 機能を持っていないけません。この機能を付与するには、root としてログオンし次のコマンドを実行します。

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE <userid>
```

Solaris 10 以降では、アプリケーションの処理を行っているエージェントの相対的な優先順位を上げることができるようになるには、インスタンス所有者は proc_prioctl 特権を持っていないけません。この特権を付与するには、root としてログオンし、次のコマンドを実行します。

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

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

さらに、DB2 が Solaris の非グローバル・ゾーンで稼働している場合は、そのゾーンの制限特権セットに `proc_priocntl` 特権を追加する必要があります。この特権をゾーンに付与するには、`root` としてログオンし、次のコマンドを実行します。

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

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

Solaris 9 では、DB2 がエージェントの相対優先順位を上げるための機能はありません。DB2 ガバナーの `ACTION NICE` 節を使用するには、Solaris 10 以降にアップグレードしてください。

schedule [class]

スケジューリングによって、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が向上します。その目的は、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平均応答時間を最小化するというものです。

ガバナーは、次の基準に基づいて、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。

- 最も多くのロックを保持しているアプリケーション (ロック待機数を削減するため)
- 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 (できるだけ多くの短期間のステートメントを、このインターバルの間に完了させるため)

各基準で上位の 3 つのアプリケーションには、他のど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、その次の順位にあるアプリケーションに 1 つ高い優先順位が与えられます。例えば、アプリケーション A は最も多くの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 番目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、ガバナーは上記の基準に基づいて、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。`class` 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` API を使用してステートメント・スイッチを非アクティブ化した場合には、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、インスタンス全体でオフになり、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。

- 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。例えば、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) を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、それぞれの応答時間は満足のいくものでは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 (例えば、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。例えば、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 つ目のグループに、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。

`class` オプションが指定されていない場合、ガバナーは、スケジュール処理に該当するアクティブ・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ついての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々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、つまり `applname`、`authid`、または `setlimit` のいずれの節も指定しないことにより、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。

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。

- ガバナーの開始または停止
- ガバナー構成ファイルの読み取り
- アプリケーションの優先順位の変更
- アプリケーションの強制停止
- エラーまたは警告の検出

ガバナー・デーモンには、それぞれに別個のログ・ファイルがあるため、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おうとすると起こる可能性のある、ファイル・ロックのボトルネックを防ぐことができます。ガバナー・ログ・ファイルを照会するには、`db2govlg` コマンドを使用します。

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 オペレーティング・システムの場合、`log` サブディレクトリーは、Windows オペレーティング・システムがアプリケーション・ログ・ファイルをホスティングするために使用する「共通アプリケーション データ」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・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、ガバ

ナーごとにファイル名が固有になるように、ガバナー・デーモンが実行されているデータベース・パーティションの番号が、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 DBPartitionNum RecType Message

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。*DBPartitionNum* 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が含まれます。

RecType フィールドには、ログに書き込まれる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。

- ACCOUNT: アプリケーションの会計統計
- ERROR: エラーが起きた
- 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
- READCFG: ガバナーが構成ファイルの読み取りを行った
- SCHEDGRP: エージェントの優先順位に変更が生じた
- START: ガバナーが開始された
- STOP: ガバナーが停止された
- WARNING: 警告が起きた

これらの値の一部について、以下のリストで詳細に説明します。

ACCOUNT

ACCOUNT レコードは、以下の状況で書き込まれます。

- 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_time** または **agent_sys_cpu_time** モニター・エレメントの値が変更された。
- アプリケーションがもはやアクティブでない。

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>
```

ERROR

ERROR レコードは、ガバナー・デーモンがシャットダウンするときに書き込まれます。

FORCE

FORCE レコードは、ガバナー構成ファイル内の規則に基づいてガバナーがアプリケーションを強制するときに書き込まれます。 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>
```

詳細は次のとおりです。

coord_partition

アプリケーションの調整データベース・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

規則違反の詳細を提供します。有効な値は以下のとおりです。

- 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であった時間の長さ
- 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の規則に基づいて、ガバナーがアプリケーションの優先順位を変更するとき書き込まれます。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>
```

詳細は次のとおりです。

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

規則違反の詳細を提供します。有効な値は以下のとおりです。

- 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であった時間の長さ
- 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

SCHEDGRP

SCHEDGRP レコードは、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、またはアプリケーションが 1 つのスケジューリング・

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に書き込まれます。 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>

説明:

cfg_line

アプリケーションをスケジューリング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

規則違反の詳細を提供します。有効な値は以下のとおりです。

- 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であった時間の長さ
- 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

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。それは、以下のようなフォーマットになります。

Database = <database_name>

WARNING

WARNING レコードは、以下の状況で書き込まれます。

- アプリケーションを強制するために `sqlfrce` 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 (SQL1611W)。
- スナップショットが -1224 (SQL1224N) または -1032 (SQL1032N) 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停止したときに生じます。
- Linux および UNIX で、シグナル・ハンドラーをインストールする試みが失敗した。

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。 *Message* フィールドには、レコード・タイプに応じて異なるその他の非標準情報が入ります。例えば、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。

ガバナー・ログ・ファイルは、次の例のようになります。

```
2007-12-11-14.54.52 0 START Database = TQTEST
2007-12-11-14.54.52 0 READCFG Config = /u/db2instance/sqlllib/tqtest.cfg
2007-12-11-14.54.53 0 ERROR SQLMON Error: SQLCode = -1032
2007-12-11-14.54.54 0 ERROR SQLMONSZ Error: SQLCode = -1032
```

ガバナーの停止

ガバナー・ユーティリティーは、データベースに接続された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。

始める前に

ガバナーを停止するには、SYSADM または SYSCTRL 権限を持っていないければなりません。

このタスクについて

重要: DB2 バージョン 9.5 で導入されたワークロード管理フィーチャーにより、DB2 ガバナー・ユーティリティーはバージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。これはDB2 pureScale環境ではサポートされていません。詳しくは、『DB2 ガバナーと Query Patroller が推奨されなくなった』(<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0054901.html>) を参照してください。

手順

ガバナーを停止するには、**STOP** パラメーターを指定して **db2gov** コマンドを使用します。

例

例えば、SALES データベースのすべてのデータベース・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
```

データベース・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
```


第 3 章 パフォーマンスに影響を及ぼす要因

システム体系

DB2 アーキテクチャーおよびプロセスの概要

クライアント・サイドでは、ローカルまたはリモート・アプリケーションが、DB2 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、共有メモリーおよびセマフォ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) や TCP/IP 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。

図 3 には、DB2 アーキテクチャーとプロセスの一般的な概要が示されています。

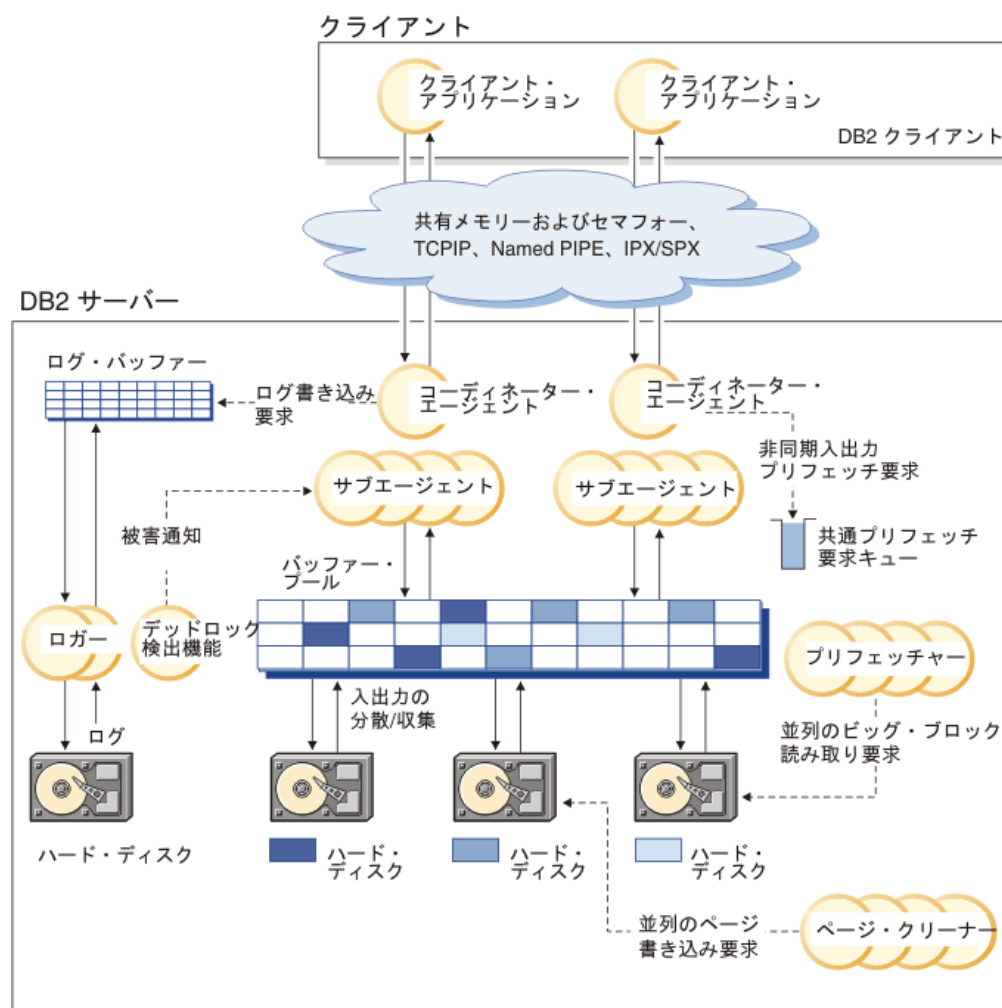


図 3. クライアント接続とデータベース・サーバーのコンポーネント

EDU は、円または円グループとして示されます。

EDU は、すべてのプラットフォーム上でスレッドとしてインプリメントされます。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 および XQuery 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・データベース環境の一部である場合、複数のサブエージェントを割り当てることができます。例えば、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、複数のプロセッサ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・アルゴリズムによって管理されます。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・プールは、データベース・サーバー・メモリのエリアであり、ここで、ユーザー・データ、索引データ、およびカタログ・データのページ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・プールは、データベース・パフォーマンスの主要な決定要素となります。

バッファ・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、アプリケーションがどのようにしてデータに迅速にアクセスするかを制御します。

- 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・プールに移動します。例えば、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・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・プールに要求されたページを移動します。データ・ストレージ用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。ストライピングによって、プリフェッチャーは複数のディスクを使用してデータを同時に取得できます。
- ページ・クリーナー は、データをバッファ・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。これらは変更されたページを検出し、その変更された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・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・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。

DB2 プロセス・モデル

DB2 プロセス・モデルの知識は、データベース・マネージャーとそれに関連するコンポーネントの対話を理解するのに役立ちます。この知識を、問題が発生したときのトラブルシューティングに生かせることもあります。

すべての DB2 データベース・サーバーによって使用されるプロセス・モデルは、データベース・サーバーおよびクライアント間の通信を容易にします。また、データベース・アプリケーションが、データベース制御ブロックおよび重要なデータベース・ファイルなどのリソースから分離されていることも確認します。

DB2 データベース・サーバーは多くの様々なタスクを実行する必要があります。例えば、データベース・アプリケーション要求を処理したり、ログ・レコードがディスクに書き出されたことを確認したりします。それぞれのタスクは通常、個別のエンジン・ディスパッチ可能単位 (EDU) によって実行されます。

DB2 データベース・サーバーでマルチスレッド化アーキテクチャーを使用することには、多くの利点があります。新規スレッドでは、必要なメモリーおよびオペレーティング・システム・リソースがプロセスと比べて少なくなります。なぜなら、一部のオペレーティング・システム・リソースを同じプロセス内のすべてのスレッドの間で共有できるからです。また、一部のプラットフォームでは、スレッド用のコンテキスト切り替え時間がプロセスの場合と比べて短いので、パフォーマンスを改善できます。すべてのプラットフォーム上でスレッド・モデルを使用すると、割り振る EDU を必要に応じて増やすことが簡単になり、複数の EDU によって共有する必要のあるメモリーを動的に割り振ることができるので DB2 データベース・サーバーを構成しやすくなります。

アクセスされるデータベースごとに別々の EDU が開始され、プリフェッチ、通信、およびロギングなどのさまざまなデータベース・タスクを扱います。データベース・エージェントは、データベースのアプリケーション要求を処理するために作成された EDU の特殊なクラスです。

一般には、DB2 データベース・サーバーに依存して、EDU のセットを管理することができます。しかし、EDU を調べる DB2 ツールがあります。例えば、**-edus** オプションを付けて **db2pd** コマンドを使用して、アクティブな EDU スレッドをすべてリストできます。

それぞれのクライアント・アプリケーション接続は、データベースを操作する単一のコーディネーター・エージェントを持ちます。コーディネーター・エージェントは、アプリケーションの代わりに作業を行い、専用メモリー、プロセス間通信 (IPC) または遠隔通信プロトコルを必要に応じて使用して、他のエージェントと通信を行います。

DB2 pureScale インスタンスでは、これらのプロセスは、ホスト上で実行中の DB2 メンバーとクラスター・キャッシング・ファシリティー (CF) のいずれかまたは両方の正常性をモニターするため、ならびにインスタンス内のすべての DB2 メンバーおよび CF にクラスター状態を配布するために使用されます。

DB2 アーキテクチャーは、アプリケーションが DB2 データベース・サーバーとは異なるアドレス・スペースで実行されるようにファイアウォールを提供します (40 ページの図 4)。ファイアウォールは、データベースおよびデータベース・マネージ

チャーを、アプリケーション、ストアド・プロシージャー、およびユーザー定義関数 (UDF) から保護します。ファイアウォールは、アプリケーション・プログラミング・エラーが、内部バッファーまたはデータベース・マネージャー・ファイルを上書きしないようにするため、データベース内のデータの保全性を保守します。ファイアウォールはまた、アプリケーション・エラーがデータベース・マネージャーを破壊しないため、信頼性も向上させます。

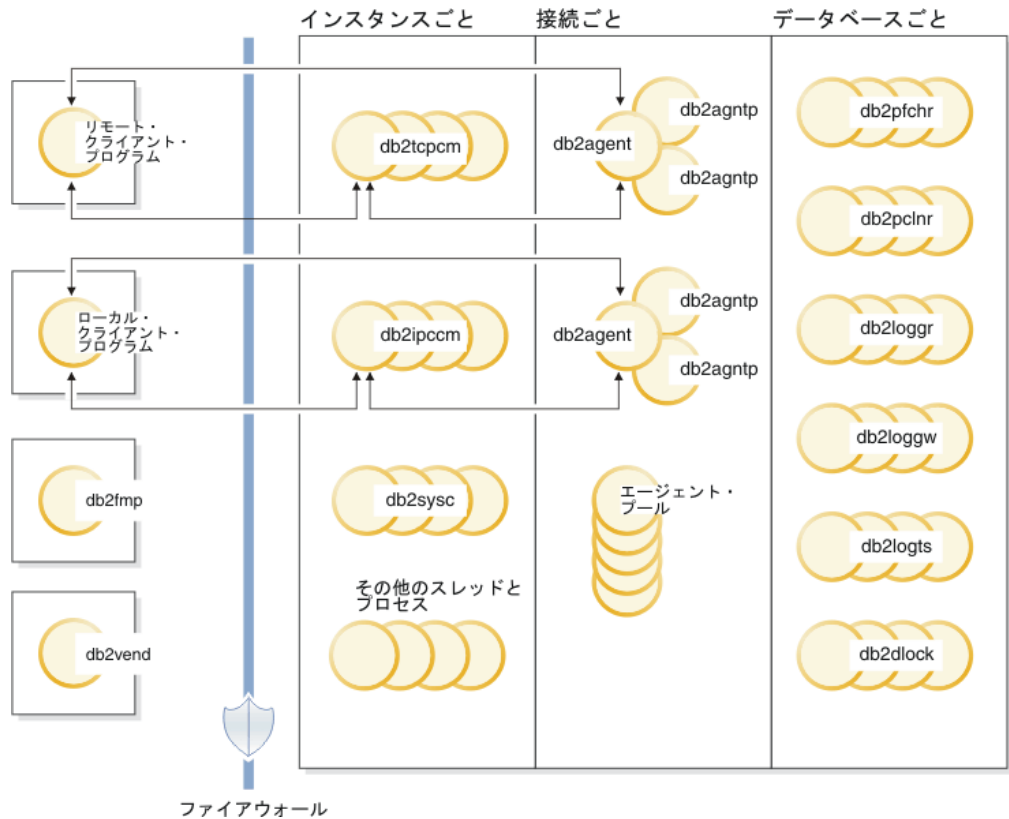


図4. DB2 データベース・システムのプロセス・モデル

クライアント・プログラム

クライアント・プログラムはリモートまたはローカルとすることが可能で、データベース・サーバーと同じマシン上で実行します。クライアント・プログラムは、通信リスナーを介してデータベースとまず連絡を取ります。

リスナー

通信リスナーは、DB2 データベース・サーバーが始動すると開始されます。それぞれの構成された通信プロトコルにはリスナーがあり、ローカル・クライアント・プログラムには、プロセス間通信 (IPC) リスナー (**db2ipccm**) があります。リスナーには以下が含まれます。

- **db2ipccm**、クライアント接続用
- **db2tccm**、TCP/IP 接続用
- **db2tccpdm**、TCP/IP ディスカバリー・ツール要求用

エージェント

ローカルまたはリモートのいずれかのクライアント・プログラム (アプリケーション) からのすべての接続要求は、対応するコーディネーター・エージェント (**db2agent**) に割り振られます。コーディネーター・エージェントが作成されると、これはアプリケーションの代わりにすべてのデータベース要求を実行します。

パーティション・データベース環境、または照会内並列処理 が有効にされているシステムでは、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** および **db2agnts**) に分散します。アプリケーションと関連しているものの、現在アイドル状態であるサブエージェントは、**db2agnta** という名前です。

コーディネーター・エージェントは、以下の可能性があります。

- 別名でデータベースに接続されています。例えば、**db2agent (DATA1)** はデータベース別名 **DATA1** に接続されています。
- インスタンスにアタッチされています。例えば、**db2agent (user1)** はインスタンス **user1** にアタッチされています。

DB2 データベース・サーバーは他のタイプのエージェント (独立コーディネーター・エージェントまたはサブコーディネーター・エージェントなど) をインスタンス化して、特定の操作を実行します。例えば、独立コーディネーター・エージェント **db2agnti** を使用してイベント・モニターを実行したり、サブコーディネーター・エージェント **db2agnsc** を使用して、データベースの異常シャットダウンの後に再始動操作を並列して行います。

ゲートウェイ・エージェント (**db2agentg**) は、リモート・データベースに関連したエージェントです。このエージェントは、クライアントがホスト・データベースにアクセスできるようにする間接接続を提供します。

アイドルのエージェントはエージェント・プールにあります。これらのエージェントは、クライアント・プログラムの代わりとして作動するコーディネーター・エージェント、または既存のコーディネーター・エージェントの代わりとして作動するサブエージェントからの要求に使用可能です。アプリケーション・ワークロードがかなり大きい場合には、適切なサイズのアイドル・エージェント・プールがあるとパフォーマンスが向上します。この場合、アイドル・エージェントは必要なときにすぐに使用でき、(スレッドの作成およびメモリーと他のリソースの割り振りと初期化を含む) アプリケーション接続ごとに新しいエージェントを割り振る必要がありません。DB2 データベース・サーバーはアイドル・エージェント・プールのサイズを自動的に管理します。

プールされたエージェントは、リモート・データベースまたはローカル・データベースに関連付けることができます。リモート・データベース上にプールされたエージェントは、プールされたゲートウェイ・エージェント (**db2agntgp**) として参照されます。ローカル・データベース上にプールされたエージェントは、プールされたデータベース・エージェント (**db2agntdp**) として参照されます。

db2fmp

fenced モード・プロセスは、ファイアウォールの外で fenced ストアード・プロシージャおよびユーザー定義関数を実行する責任があります。 **db2fmp** プロセスは

常に分離されたプロセスですが、実行するルーチンのタイプによっては、マルチスレッドの場合があります。

db2vend

db2vend プロセスは、EDU の代わりにベンダー・コードを実行するプロセスです。例えば、ログ・アーカイブ用のユーザー出口プログラムを実行する場合 (UNIX のみ) などです。

データベース EDU

以下のリストには、各データベースによって使用される、いくつかの重要な EDU が含まれています。

- **db2pfchr**、バッファー・プール・プリフェッチャー用
- **db2pc1nr**、バッファー・プール・ページ・クリーナー用
- **db2loggr**、トランザクション処理およびリカバリーをハンドルするログ・ファイルの取扱用
- **db2logw**、ログ・ファイルへのログ・レコードの書き込み用
- **db2logts**、どのログ・ファイルでどの表スペースがログ・レコードを持つかのトラッキング用。この情報は、データベース・ディレクトリーの DB2TSCHG.HIS ファイルに記録されます。
- **db2dlock**、デッドロック検出用。パーティション・データベース環境では、各パーティションの **db2dlock** EDU によって集められた情報を調整するのに、追加スレッド (**db2glock**) が使用されます。**(db2glock)** はカタログ・パーティションでのみ実行されます。DB2 pureScale環境では、各メンバーについて **db2dlock** EDU が収集した情報を調整するために、**db2glock** EDU が使用されます。メンバーごとに **db2glock** EDU が 1 つずつ開始されますが、アクティブなのは 1 つだけです。
- **db2stmm**、セルフチューニング・メモリー管理フィーチャー用
- **db2taskd**、バックグラウンド・データベース・タスクの分散用。こうしたタスクは **db2taskp** と呼ばれるスレッドによって実行されます。
- **db2fw**、イベント・モニターの高速ライター。ボリュームが多いときに使用し、イベント・モニター・データを表、ファイル、パイプに並行して書き込みます。
- **db2hadrp**、高可用性災害時リカバリー (HADR) 基本サーバー・スレッド
- **db2hadrs**、HADR スタンバイ・サーバー・スレッド
- **db2lfr**、個別のログ・ファイルを処理するログ・ファイル・リーダー用
- **db2loggr**、トランザクション処理およびリカバリーをハンドルするログ・ファイルの取扱用
- **db2logw**、ログ・ファイルへのログ・レコードの書き込み用
- **db2logmgr**、ログ・マネージャー用。リカバリー可能なデータベースのログ・ファイルを管理します。
- **db2logts**、どのログ・ファイルでどの表スペースがログ・レコードを持つかのトラッキング用。この情報は、データベース・ディレクトリーの DB2TSCHG.HIS ファイルに記録されます。
- **db2lused**、オブジェクト使用の更新用

- **db2pcsd**、パッケージ・キャッシュのオートノミック・クリーンアップ用
- **db2pfchr**、バッファ・プール・プリフェッチャー用
- **db2pc1nr**、バッファ・プール・ページ・クリーナー用
- **db2redom**、再実行マスター用。リカバリー中に、再実行ログ・レコードを処理し、ログ・レコードを処理のために再実行作業に割り当てます。
- **db2redow**、再実行作業用。リカバリー中に、再実行マスターの要求で再実行ログ・レコードを処理します。
- **db2shred**、ログ・ページ内での個別のログ・レコード処理用
- **db2stmm**、セルフチューニング・メモリー管理フィーチャー用
- **db2taskd**、バックグラウンド・データベース・タスクの分散用。こうしたタスクは **db2taskp** と呼ばれるスレッドによって実行されます。
- **db2w1md**、ワークロード管理統計の自動収集用
- イベント・モニター・スレッドは、以下のように識別されます。
 - `db2evm%1%2 (%3)`

なお、`%1` の部分は以下ようになります。

- `g` - グローバル・ファイル・イベント・モニター
- `gp` - グローバル・パイプ・イベント・モニター
- `l` - ローカル・ファイル・イベント・モニター
- `lp` - ローカル・パイプ・イベント・モニター
- `t` - 表イベント・モニター

`%2` の部分は以下ようになります。

- `i` - コーディネーター
- `p` - コーディネーターでない

また、`%3` はイベント・モニター名です。

- バックアップおよびリストア・スレッドは、以下のように識別されます。
 - `db2bm.%1.%2` (バックアップおよびリストアのバッファ・マニピュレーター) および `db2med.%1.%2` (バックアップおよびリストアのメディア・コントローラー)。ここで `%` の部分は以下ようになります。
 - `%1` は、バックアップまたはリストアのセッションを制御するエージェントの EDU ID です。
 - `%2` は、特定のバックアップまたはリストアのセッションに属するスレッド (多数の場合もある) の間の違いを識別するために使用される順次値です。
 例えば、**db2bm.13579.2** は EDU ID 13579 の **db2agent** スレッドによって制御される 2 番目の **db2bm** スレッドです。
- 以下のデータベース EDU は、DB2 pureScale環境でのロッキングに使用されません。
 - **db2LLMn1** は、グローバル・ロック・マネージャーにより送信される情報を処理します。これら EDU はメンバーごとに 2 つあります。1 つは 1 次 CF 用、もう 1 つは 2 次 CF 用です。
 - **db2LLMn2** は、データベースの活動化と非アクティブ化中に使用される特殊なタイプのロックを行うためにグローバル・ロック・マネージャーにより送信さ

れる情報を処理します。これら EDU はメンバーごとに 2 つあります。1 つは 1 次 CF 用、もう 1 つは 2 次 CF 用です。

- **db2LLMng** は、他のメンバーがこれらのロックを待って待機しているときに、このメンバーが保持しているロックが適切なタイミングで解放されるようにします。
- **db2LLMr1** は、グローバル・ロック・マネージャーに対するロックの解放を処理します。
- **db2LLMrc** は、データベースのリカバリー操作および CF のリカバリー中に発生する処理を行います。

データベース・サーバー・スレッドとプロセス

データベース・サーバーを機能させるには、システム・コントローラー (UNIX の場合は **db2sysc**、Windows オペレーティング・システムの場合は **db2syscs.exe**) が存在する必要があります。以下のスレッドとプロセスは、様々なタスクを実行します。

- **db2acd**、ヘルス・モニター、自動保守ユーティリティー、および管理用タスク・スケジューラーをホストするオートノミック・コンピューティング・デーモン。このプロセスは以前は **db2hmon** として知られていました。
- **db2aiothr**、データベース・パーティションの非同期入出力要求を管理する (UNIX のみ)
- **db2alarm**、要求されたタイマーの期限が切れたときに EDU に通知する (UNIX のみ)
- **db2disp**、クライアント接続コンセントレーターのディスパッチャー
- **db2fcms**、高速コミュニケーション・マネージャーの送信側デーモン
- **db2fcmr**、高速コミュニケーション・マネージャーの受信側デーモン
- **db2fmd**、障害モニター・デーモン
- **db2licc**、インストール済みの DB2 ライセンスを管理する
- **db2panic**、エージェント限度に達した後の緊急要求をハンドルするパニック・エージェント
- **db2pdbc**、リモート・データベース・パーティションからの並列要求をハンドルする並列システム・コントローラー (パーティション・データベース環境でのみ使用)
- **db2resync**、グローバル再同期リストをスキャンする再同期エージェント
- **db2sysc**、メイン・システム・コントローラー EDU。重要な DB2 サーバー・イベントを処理します。
- **db2sysc** (アイドル) は、DB2 アイドル・プロセスです。ホストでゲスト・メンバーの **restart light** を速やかに可能にし、常駐メンバーとリソースについて競合しません。
- **db2thc1n**、EDU の終了時にリソースをリサイクルする (UNIX のみ)
- **db2wdog**、異常終了をハンドルする UNIX および Linux オペレーティング・システムのウォッチドッグ
- **db2w1mt**、WLM ディスパッチャー・スケジューリング・スレッド
- **db2w1mtm**、WLM ディスパッチャー・タイマー・スレッド

データベース・エージェント

アプリケーションがデータベースにアクセスするとき、複数のプロセスまたはスレッドが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、アプリケーション要求にサービスを提供するために使用される、データベース・マネージャー内のスレッドです。バージョン 9.5 では、エージェントはすべてのプラットフォーム上でスレッドとして実行されます。

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・エージェントは、アプリケーションが切断するまでそのアプリケーションにアタッチしたままなので、アプリケーションとの関連は最も長くなります。この規則の唯一の例外は、エンジン・コンセントレーターが使用可能にされている場合に生じます。この場合、コーディネーター・エージェントはその関連をトランザクション境界 (COMMIT または ROLLBACK) で終了することがあります。

作業エージェントには以下の 3 種類があります。

- 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。

- 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データベース接続にはそれぞれ、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。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有します。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・エージェントになる場合があります。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デタッチされると、そのコーディネーター・エージェントの状態は次のようになります。

- 他の接続が待機状態の場合はアクティブ・コーディネーター・エージェントになります。
- どの接続も待機状態でなく、プール・エージェントの最大数が自動的に管理されているか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。
- どの接続も待機状態でなく、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

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。DB2 バージョン 9.5 では、サブエージェントは非パーティション環境および照会内並列処理が有効にされていない環境でも存在可能です。

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・プログラムの代わりとして作動するコーディネーター・エージェントからの要求時に、または既存のコーディネーター・エージェントの代わりとして作動するサブエージェントのために使用可能です。使用可能なエージェントの数は、データベース・マネージャーの **num_poolagents** 構成パラメーターの値によって異なります。

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量のオーバーヘッドが必要なため、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。

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・サーバー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 の関係を確認します。しかし使用する環境によっては、接続済みアプリケーションの数と、処理が可能なアプリケーション要求の数の間の多対 1 の関係が必要になる場合があります。

次の 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り、これらの要素が別個に制御されます。

- **max_connections** パラメーターは、接続されるアプリケーションの最大数を指定します。
- **max_coordagents** パラメーターは、同時に処理可能なアプリケーション要求の最大数を指定します。

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・データベース・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多いほど、使用できるグローバル・リソースの上限に達

する可能性も高くなります。これを避けるには、**max_connections** の値を **max_coordagents** より高い値に設定するか、または両方のパラメーターを **AUTOMATIC** に設定します。

これらのパラメーターを **AUTOMATIC** に設定することが有利になる 2 つの具体的なシナリオがあります。

- 必要とされるすべての接続をシステムが処理できることははっきりしているが、使用されるグローバル・リソースの量を (コーディネーター・エージェントの数を制限することにより) 制限する場合は、**max_connections** を **AUTOMATIC** に設定します。 **max_connections** が **max_coordagents** より大きいときは、接続コンセントレーターが使用可能になります。
- 接続またはコーディネーター・エージェントの最大数は制限しないが、接続されるアプリケーションの数と処理されるアプリケーション要求の数との多対 1 の関係をシステムが必要としている、またはその関係があることでシステムに利点があることが分かっている場合、両方のパラメーターを **AUTOMATIC** に設定します。両方のパラメーターが **AUTOMATIC** に設定されている場合、データベース・マネージャーは指定したこの値を、接続対コーディネーター・エージェントの理想的な比率として使用します。どちらのパラメーターも、開始値および **AUTOMATIC** 設定を使用して構成できる点に注意してください。例えば、次のコマンドにより、値 200 および **AUTOMATIC** の両方が **max_coordagents** パラメーターに関連付けられます。 `update dbm config using max_coordagents 200 automatic.`

例

次のシナリオを考えます。

- **max_connections** パラメーターが **AUTOMATIC** に設定されていて、現行値は 300 になっている
- **max_coordagents** パラメーターが **AUTOMATIC** に設定されていて、現行値は 100 になっている

max_connections 対 **max_coordagents** の比率は、300:100 です。データベース・マネージャーは接続が行われる時には新しいコーディネーター・エージェントを作成するので、接続集中は必要な場合のみ適用されます。この設定により、以下の処置が取られます。

- 接続 1 から 100 では、新しいコーディネーター・エージェントが作成される。
- 接続 101 から 300 では、新しいコーディネーター・エージェントが作成されない。それらは既に作成済みの 100 個のエージェントを共有する。
- 接続 301 から 400 では、新しいコーディネーター・エージェントが作成される。
- 接続 401 から 600 では、新しいコーディネーター・エージェントが作成されない。それらは既に作成済みの 200 個のエージェントを共有する。
- 以降、同様に続きます。

この例では、接続済みアプリケーションが、各ステップで新規コーディネーター・エージェントを確実に作成できるほど十分に実行されていることを前提としていま

す。一定の期間後に、接続済みアプリケーションが十分な量の作業を駆動しなくなっている場合、コーディネーター・エージェントは非アクティブになり、終了します。

接続の数は減っているが、残りの接続により実行中の作業の量が多い場合は、コーディネーター・エージェントの数が直ちには減らない場合があります。

max_connections および **max_coordagents** パラメーターは、エージェント・プールまたはエージェント終了に直接影響することはありません。通常のエージェント終了規則が引き続き適用されますが、これは、接続対コーディネーター・エージェントの比率が、指定した値に正確に対応していない可能性があることを意味します。エージェントは、終了する前に再利用のためにエージェント・プールに戻される場合があります。

より細分性の高い制御が必要な場合は、より単純化した比率を指定します。例えば、前述の例の 300:100 の比率は、3:1 と表現できます。 **max_connections** を 3 (AUTOMATIC) に設定し、 **max_coordagents** を 1 (AUTOMATIC) に設定すると、3 つの接続ごとに 1 つのコーディネーター・エージェントを作成できます。

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の両方のアプリケーション・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・サーバーがある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。

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の値が **max_coordagents** 構成パラメーターの値より大きい場合には必ず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 と呼ばれる固有のエンジン・ディスパッチ可能単位 (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同時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。

49 ページの図 5 では、DB2 サーバー内の各円は、オペレーティング・システム・スレッドを使用してインプリメントされた EDU を表しています。

サーバー・マシン

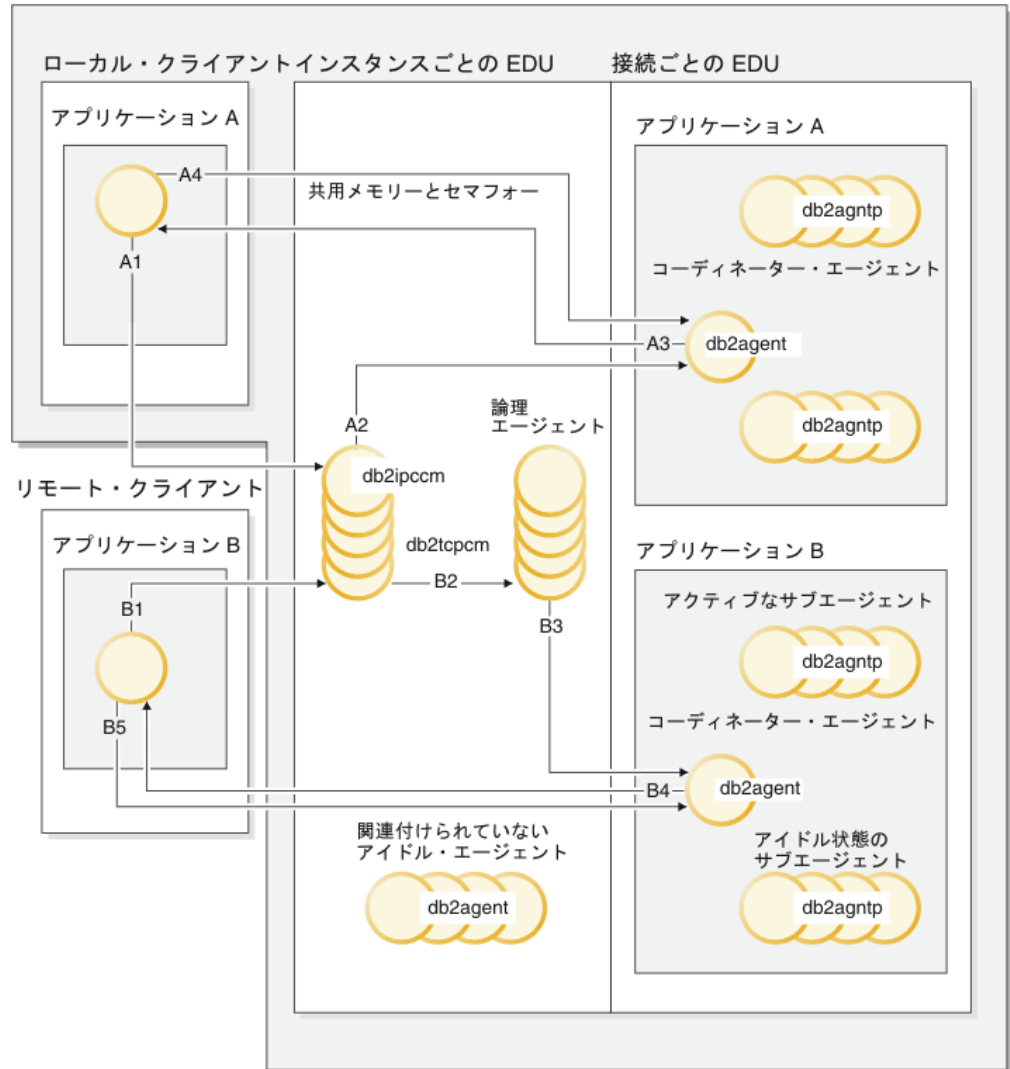


図 5. クライアント/サーバー処理モデルの概要

- A1 では、ローカル・クライアントが 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。
- A3 では、その後、コーディネーター・エージェントはクライアント・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有メモリ通信を確立します。
- 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します。
- B1 では、リモート・クライアントが db2tcpccm を介して通信を確立します。他の通信プロトコルが選択された場合、適切な通信マネージャーが使用されます。
- B2 では、db2tcpccm が db2agent EDU 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。

- B4 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ます。
- B5 では、リモート・クライアント・アプリケーションがデータベースに接続します。

以下の事柄にも注目してください。

- 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、またはパーティション内並列処理が使用可能な環境では、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。
- エージェント・プール (db2agent) では、アイドル・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・コミット操作、バックアップおよびリストア操作、その他のタスクを管理します。

51 ページの図 6 は、サーバー・マシン環境の一部である追加の EDU を示しています。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) の共有プール、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。

サーバー・マシン

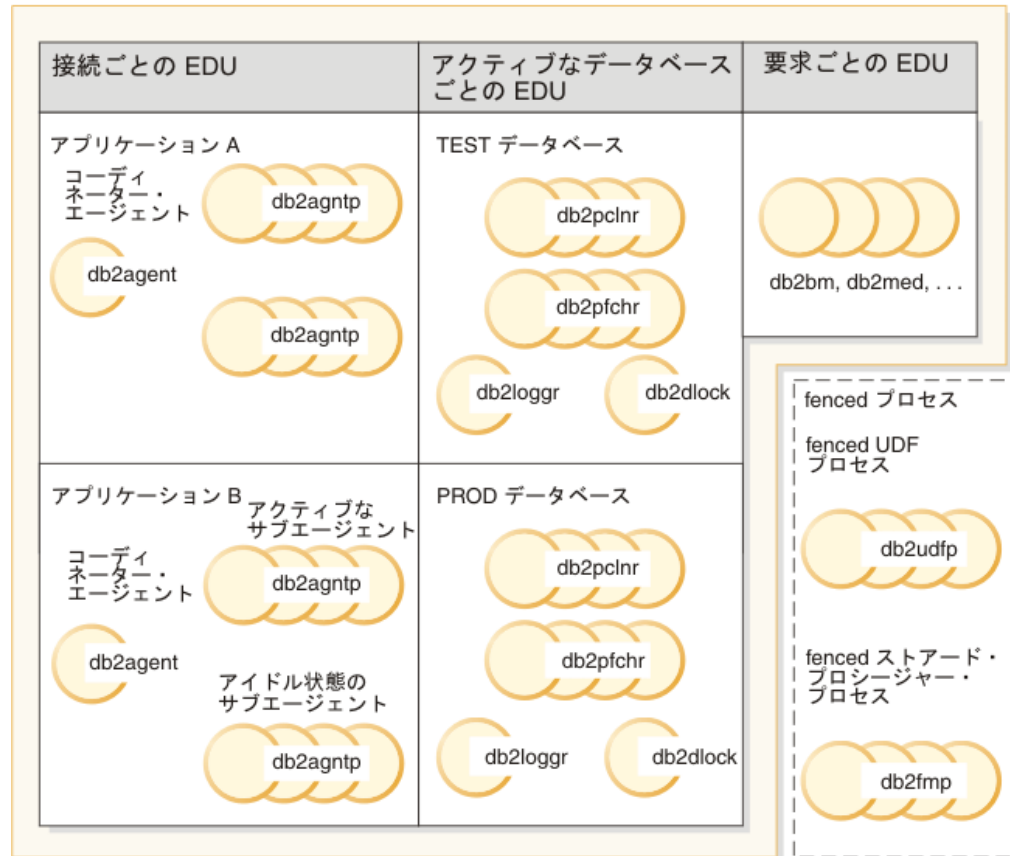


図 6. データベース・サーバーにおける EDU

図には示されていませんが、fenced ユーザー定義関数 (UDF) およびストアド・プロシージャは、その作成と破棄に関連するコストを最小限に抑えるように管理されます。 **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値は「YES」であり、ストアド・プロシージャ・プロセスを、次のプロシージャ呼び出しで再使用できます。

注: unfenced UDF およびストアド・プロシージャは、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。

52 ページの図 7 では、単一データベース・パーティション処理モデルと、複数データベース・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。

パーティションにあります。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。

カタログ・データベース・パーティションに関連した追加の 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データベース・パーティション間のグローバル・デッドロックを制御します。

アプリケーションからの各接続要求は、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・エージェントがあるデータベース・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・データベース・パーティション と呼ばれます。

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・データベース・パーティションにより他のデータベース・パーティションのサブエージェントに送られます。結果すべてがコーディネーター・データベース・パーティションで統合されてから、アプリケーションに戻されます。

複数のデータベース・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 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 (SMP) マシンで大変役立ちます。この環境では、データベース・パーティション間の通信は、共有メモリーおよびセマフォアを使用するように最適化されます。

クライアント接続用の接続コンセントレーターの改善

接続コンセントレーターは、多くの同時クライアント接続を効率的に処理できるので、頻度が高いものの比較的一過性の接続が多いアプリケーションのパフォーマンスが向上します。さらに、それぞれの接続時のメモリー使用量が削減され、コンテキスト切り替えの数が減りました。

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の値が **max_coordagents** 構成パラメーターの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がある環境では、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。このフィーチャーは、これまで DB2 Connect 接続プールでのみ利用できたフィーチャーを取り入れたものです。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に必要な時間を削減します。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。新しい接続要求を受信すると、データベース・マネージャー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。

接続プールまたは接続コンセントレーターを使用するアプリケーションで最良のパフォーマンスを得るため、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。詳しくは、DB2 Connect 製品資料を参照してください。

例

- 平均 1000 人のユーザーが同時に接続する単一パーティション・データベースについて考慮します。時には、接続ユーザーの数が平均より多くなることがあります。並行トランザクション数は 200 に達する可能性があります。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できます。

- **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。
 - **max_connections** は AUTOMATIC (値 1000) として設定し、これにより接続をいくつでもサポートします。この例では、250 より大きい値であれば、接続コンセントレーターが使用可能になります。
 - **num_poolagents** はデフォルト値のままにします。これによって、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、さらに新しいエージェントの作成によるオーバーヘッドを避けます。
- 平均 1000 人のユーザーが同時に接続する単一パーティション・データベースについて考慮します。時には、接続ユーザーの数が 2000 になることがあります。任意の一時点に実行が予想されるのは、平均 500 ユーザーです。並行トランザクションの数は約 250 です。コーディネーター・エージェント数が 500 であると通常は多すぎます。接続ユーザー数が 1000 の場合、250 コーディネーター・エージェントで十分です。

このワークロードを処理するために、データベース・マネージャー構成を以下のように更新できます。

```
update dbm cfg using max_connections 1000 automatic
update dbm cfg using max_coordagents 250 automatic
```

これは、接続の数が 1000 を超える場合、接続の合計数により判別された最大数で、追加のコーディネーター・エージェントが必要に応じて作成されるということです。ワークロードが増えると、データベース・マネージャーはコーディネーター・エージェントに対する接続の割合を相対的に安定させようとしています。

- 接続コンセントレーターを有効にせず、接続するユーザーの数を制限するとします。例えば、同時接続するユーザーの数を 250 に制限するには、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できます。
- **max_coordagents** を 250 に設定します。
- **max_connections** を 250 に設定します。
- 接続コンセントレーターを有効にせず、接続するユーザーの数も制限しないとします。データベース・マネージャー構成を以下のように更新できます。

```
update dbm cfg using max_connections automatic
update dbm cfg using max_coordagents automatic
```

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、または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、各データベース・パーティション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ることができ、それらが関連付けされたアプリケーションから、または新規の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・マネージャーでそれらのサブエージェントを使用できます。

システム内のパフォーマンスとメモリー消費量への影響は、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 つのデータベース・パーティションでアプリケーションとの関連付けを保持できるエージェントとサブエージェントの合計数に影響します。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に再度関連付けするという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。

デフォルトでは、**num_poolagents** は AUTOMATIC (値 100) に設定され、データベース・マネージャーはプールするアイドル・エージェントの数を自動的に管理します。

手動で設定した **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。その後、別のアプリケーションが新しいサブエージェントを必要としているときに、そのエージェント・プール内にサブエージェントがない場合、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・プールにあるアクティブでないサブエージェントをリサイクルします。この動作により、リソースは完全に使用されます。

手動で設定した **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できないデータベース・マネージャー・リソースが使用される可能性があります。

接続コンセントレーターが使用可能である場合、**num_poolagents** の値は、同時にプール内でアイドル状態のままであるエージェントの正確な数を必ずしも反映するわけではありません。さらに多くのワークロード・アクティビティを処理するために、一時的にエージェントが必要になる可能性があります。

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は、データベース・エージェント以外にもあります。例えば、次のようなアクティビティがあります。

- 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
- 通信および IPC listener
- 表スペース・コンテナのリバランサー

良好なパフォーマンスのための構成

InfoSphere® Balanced Warehouse® (BW)、または SAP システム内のものなど、一部のタイプの DB2 デプロイメントには厳密に指定されている構成があります。

BW の場合、CPU 数、CPU に対するメモリーの比率、ディスク数と構成、およびバージョンなどのハードウェア要因については、最適な構成を判別するための徹底的なテストに基づいて事前指定されています。SAP の場合には、ハードウェア構成は明確に指定されていません。ただし、使用可能な非常に多くのサンプル構成があります。加えて、SAP ベスト・プラクティスによって、推奨されている DB2 構成設定が提供されています。十分にテスト済みの構成ガイドラインが提供されているシステムで DB2 デプロイメントを使用している場合、一般的な経験則ではなく、通常はそのガイドラインを活用してください。

詳細なハードウェア構成がまだ決まっていない場合には、提案されているシステムについて考慮してください。ここでの目的は、システムが良好なパフォーマンスを発揮するために鍵となる幾つかの構成上の決定を識別することです。通常このステップはシステムを稼働する前に行うため、実際にどのように動作するかについては十分に理解していない場合があります。そのため、システムで実行予定の事柄についての知識に基づいて、『最良の推測』をある意味では行う必要があります。

ハードウェア構成

システム・パフォーマンスについての構成において、CPU の能力は独立した主要な変数の 1 つとなります。通常その他すべてのハードウェア構成は CPU の能力を考慮に入れて決定されるので、与えられたワークロードに必要な CPU 能力を予測するのは簡単ではありません。ビジネス・インテリジェンス (BI) 環境での道理にかなった見積もりは、プロセッサ・コアごとに 200 から 300 GB のアクティブな生データです。他の環境の場合、既存の 1 つ以上の DB2 システムに基づいて、必要とされる CPU 量を測定するのが健全な方法です。例えば、新しいシステムで処理する必要のあるユーザー数が 50% 増える場合、それぞれのユーザーが少なくとも既存のシステム上における同様の複雑さの SQL を実行すると、CPU 能力を 50% 増やす必要があると考えるのが妥当です。同様に、CPU の使用量に変更が予測される他の要因 (スループット要件が異なったり、トリガーの使用法や参照整合性において変更があったりする場合など) も考慮に入れる必要があります。

(入手可能な情報から導出される) CPU 要件に関する最善の構成を考えると、ハードウェア構成の他の面についても解決できるようになります。必要なシステム・ディスク容量をギガバイトまたはテラバイト単位で考慮する必要がありますが、パフォーマンスで最も重要な要因は 1 秒当たりの入出力 (IOPS) または 1 秒当たりのデータ転送 (メガバイト) に関する能力です。実際上の問題として、これは関係する個々のディスク数によって決定します。

なぜでしょうか? 過去 10 年の間、容量やコストの面ではディスクの進歩の方が優れていますが、速度の面で CPU は驚くほどの進歩を遂げました。ディスク・シーク時間や転送速度の面でも改善が見られますが、CPU 速度に見合うほどではありません。そのため、最新のシステムで必要な集約パフォーマンスを得るには、複数のディスクを使用することがかつてなく重要になっています。とりわけ、かなりの量のランダム・ディスク入出力が予想されるシステムでは大切です。多くの場合、シ

ステム内の合計データ量を含めることが可能な最小数に近いディスクで済ませようとしたくなりますが、そのようにすると通常はパフォーマンスが極端に悪くなります。

RAID ストレージの場合、または個別アドレッシング可能なドライブの場合、経験則上、1 つのプロセッサ・コアに対して少なくとも 10 から 20 のディスクを構成します。ストレージ・サーバーの場合にも、同じような数が推奨されています。ただし、この場合、幾らかの注意が必要です。ストレージ・サーバー上のスペースの割り振りは、多くの場合スループットではなく容量を視野に入れて実行されます。データベース・ストレージの物理的なレイアウトを把握し、論理的に別個なストレージが不注意にもオーバーラップしないようにするのは適切なことです。例えば、4Way システムの妥当な割り振りは、それぞれが 8 ドライブの 8 アレイとなります。しかし、すべての 8 アレイで同じ 8 個の基礎を成す物理ドライブを共有すると、この構成のスループットは 64 個の物理ドライブに分散されている 8 アレイに比べて、大幅に減少します。

DB2 トランザクション・ログ用に幾らかの専用 (非共有) ディスクを取り分けて置くのは良いことです。その理由は、このログの入出力特性は DB2 コンテナーとは大いに異なるからです。例えば、特に書き込みアクティビティの度合いが高いシステムでは、ログ入出力と他のタイプの入出力間の競合によって、ロギングのボトルネックが生じる可能性があります。

一般に、RAID-1 ペアのディスクでは、1 秒あたりに最大 400 の書き込み専用の DB2 トランザクションに対応できる十分なロギング・スループットを提供できます。スループット率が高くなれば、またはロギング量が多くなれば (例えばバルク挿入時など)、ログ・スループットもそれだけ多く必要となります。そのためには、書き込みキャッシュ・ディスク・コントローラーを介してシステムに接続されている RAID-10 構成で追加のディスクを提供できます。

CPU とディスクは実際には異なる時間目盛り (ナノ秒とマイクロ秒) で機能するので、適切な処理パフォーマンスを得るにはそれらを分離する必要があります。これには、メモリーも関係します。データベース・システムでは、メモリーの主要な目的は入出力を回避することにあるので、ある程度まではシステムに多くのメモリーがあればあるほど、より良いパフォーマンスが得られます。幸いなことに、ここ数年の間にメモリーの価格はかなり下落したので、10 から 100 ギガバイト (GB) の RAM を有するシステムも珍しくありません。一般的に、プロセッサ・コアに対して 4 から 8 ギガバイトがほとんどのアプリケーションで必要となります。

AIX の構成

良好なパフォーマンスを得るために変更する必要がある AIX パラメーターは比較的わずかです。ご使用のシステムで既に特定の設定がなされている場合 (例えば BW または SAP 構成)、やはり以下の一般的なガイドラインよりもそれらの構成を優先してください。

- VMO パラメーター **LRU_FILE_REPAGE** は 0 に設定してください。このパラメーターは、AIX が計算ページまたはファイル・システム・キャッシュ・ページを犠牲にするかどうかを制御します。また **minperm** は 3 に設定してください。これは、どちらも AIX 6.1 でのデフォルト値です。

- AIO パラメーター **maxservers** は、初めは CPU ごとにデフォルト値である 10 のままにできます。システムがアクティブになってから、**maxservers** を以下のよう
に調整します。
 1. `ps -elfk | grep aio` コマンドの出力を入手して、すべての非同期入出力 (AIO) カーネル・プロセス (aioserver) が同じ CPU 時間を消費しているかどうかを判別します。
 2. 同じである場合には、**maxservers** の設定が小さすぎる可能性があります。**maxservers** を 10% ずつ増やし、ステップ 1 を繰り返します。
 3. 他の aioserver と比べて使用している CPU 時間が短い aioserver がある場合、システムには少なくとも必要とする数の aioserver があります。使用している CPU 量が少ない aioserver 数が 10% を超える場合、**maxservers** を 10% ずつ減らして、ステップ 1 を繰り返します。
- AIO パラメーター **maxreqs** は $\text{MAX}(\text{NUM_IOCLEANERS} \times 256, 4096)$ に設定してください。このパラメーターは、未確定の AIO 要求の最大数を制御します。
- **hdisk** パラメーター **queue_depth** は、アレイ内の物理ディスク数に基づいて決定します。例えば、IBM ディスクの場合、**queue_depth** のデフォルト値は 3 で、推奨値は $3 \times \text{number-of-devices}$ です。このパラメーターは、キューに入れることのできるディスク要求数を制御します。
- ディスク・アダプター・パラメーター **num_cmd_elems** は、アダプターに接続されているすべてのデバイスの **queue_depth** の合計に設定します。このパラメーターは、キューに入れることのできる、アダプターへの要求数を制御します。

Solaris および HP-UX の構成

Solaris または HP-UX 上で稼働している DB2 の場合、**db2osconf** ユーティリティーを使用してカーネル・パラメーターを確認するとともに、システム・サイズに基づいて推奨します。**db2osconf** ユーティリティーを使用すると、メモリーまたは CPU に基づいて、または現在のシステム構成と想定する今後の構成を比較した汎用の倍率を使用して、カーネル・パラメーターを指定できます。SAP アプリケーションなどの大規模システムを実行している場合、倍率 2 以上を使用するのが適しています。一般に、Solaris および HP-UX を構成する際に **db2osconf** を使用するのが妥当な開始点となりますが、現在および今後のワークロードを考慮に入れられないので最適値ではありません。

Linux の構成

DB2 データベース・マネージャーはかぎとなる Linux カーネル・パラメーターを自動的に更新して、構成に関するさまざまな要件を満たすようにします。

詳しくは、「DB2 サーバー機能 インストール」の『カーネル・パラメーターの要件 (Linux)』を参照してください。

パーティション・データベース環境

パーティション・データベース環境を使用する判断は一般に、純粋にデータ量に基づくものではなく、ワークロードに基づくものです。一般的な指針として、ほとんどのパーティション・データベース環境はデータウェアハウスおよびビジネス・インテリジェンスの領域です。パーティション・データベース環境は大規模で複雑な照会環境で大いに推奨されています。そのシェアード・ナッシング・アーキテクチ

ャーによって、優れた拡張性が可能になるためです。急激に増加する可能性の低い小規模なデータマート (最大約 300 GB) の場合、DB2 Enterprise Server Edition 構成が、多くの場合に適正な選択となります。ただし、大規模な、または急成長する BI 環境の場合、パーティション・データベース環境から大いに益が得られます。

通常、標準的なパーティション・データベース・システムには、データ・パーティションごとに 1 つのプロセッサ・コアがあります。例えば、 n 個のプロセッサ・コアを持つシステムにはパーティション 0 のカタログと、 n 個の追加データ・パーティションがあることがあります。カタログ・パーティションの使用負荷が大きい場合には (例えば、単一パーティション・ディメンション表を保持するなど)、プロセッサ・コアも割り振る可能性があります。システムで非常に多くのアクティブな並行ユーザーをサポートする場合には、パーティションごとに 2 つのコアが必要となることがあります。

一般的な指針としては、1 つのパーティションに対して約 250 GB のアクティブ生データを計画してください。

InfoSphere Balanced Warehouse 資料には、パーティション・データベース構成のベスト・プラクティスに関する詳細情報が記載されています。この資料には、Balanced Warehouse 以外のデプロイメントに関する有用な情報も含まれています。

コード・ページと照合の選択

コード・ページまたはコード・セット、および照合シーケンスの選択は、データベースの動作に影響を与えるだけでなく、パフォーマンスにも多大な影響があります。Unicode の使用がごく一般的になっています。Unicode を使用すると、従来の 1 バイトのコード・ページを使用する場合に比べ、非常に多様な文字ストリングをデータベースで表すことが可能になるためです。DB2 バージョン 9.5 の新規データベースでは、Unicode がデフォルトです。ただし、Unicode コード・セットは一部の個別文字を複数バイトを用いて表すので、ディスクとメモリーの要件が増大する可能性があります。例えば、最も一般的な Unicode コード・セットの 1 つである UTF-8 コード・セットは、1 文字ごとに 1 から 4 バイトを使用します。1 バイトのコード・セットから UTF-8 へのマイグレーションのための平均ストリング拡張係数は、マルチバイト文字が使用される頻度によって異なるため、見積もることがとても困難です。典型的な北米コンテンツの場合、拡張は通常ありません。ほとんどの西ヨーロッパ言語の場合、標準的なアクセント付き文字の使用では 10% 前後の拡張がもたらされます。

これに加え、Unicode を使用すると、1 バイト・コード・ページに比べて余分の CPU 使用量が生じる可能性があります。最初に、拡張が生じる場合、ストリングが長ければそれだけ多くの操作の処理が必要になります。より重要なこととして 2 番目に、ロケールに依存する UCA ベースの照合などの一層高度な Unicode 照合シーケンスが使用するアルゴリズムは、1 バイト・コード・ページで使用される一般的な SYSTEM 照合よりもコストがはるかに高くなる可能性があります。このようにコストが増大するのは、文化的に正しい方法で Unicode ストリングをソートするのが複雑なためです。影響を与える操作としては、ソート、ストリング比較、LIKE 処理、および索引作成があります。

ご使用のデータを正しく表すのに Unicode が必要な場合には、照合シーケンスを注意深く選択してください。

- データベースに複数の言語のデータが含まれていて、そのデータの正しいソート順が最も重要な要素である場合、ロケールに依存する UCA ベースの照合の 1 つを使用してください。データとアプリケーションによっては、IDENTITY シーケンスと比べて、1.5 から 3 倍のパフォーマンス・オーバーヘッドが生じる可能性があります。
- ロケールに依存する UCA ベースの照合には、正規化されたものと正規化されていないものの両方があります。正規化された照合には属性 NO が指定されており、誤った形式の文字を処理するための追加検査を提供します。正規化されていない照合には属性 NX が指定されており、このような検査を提供しません。誤った形式の文字処理の問題がない限りは、非正規化版を使用します。正規化コードを回避するパフォーマンス上の利点があるからです。とは言え、正規化されていない、ロケールに依存する UCA ベースの照合でさえ、コストはとて大きくなります。
- データベースを 1 バイト環境から Unicode 環境に移行している場合、様々な言語のホスティングに関する要件が厳密ではないと (ほとんどのデプロイメントはこのカテゴリーです)、言語対応照合が適切になる可能性があります。言語対応照合 (例えば、SYSTEM_819_BE) は、多くの Unicode データベースには 1 つの言語だけのデータが含まれているという事実の利点を活かします。この照合では、SYSTEM_819 などの 1 バイト照合と同じ参照表ベースの照合アルゴリズムを使用しているので、とても効率的です。一般的な規則として、元の 1 バイト・データベース内の照合動作を受け入れ可能だった場合、Unicode への移行後に言語内容が大幅に変更されない限りは、文化対応照合を考慮してください。これにより、文化的に正しい照合に比べて、パフォーマンス上の非常に大きな利点がもたらされます。

物理的なデータベース設計

- 一般に、ファイル・ベースのデータベース管理ストレージ (DMS) REGULAR 表スペースの方が、システム管理ストレージ (SMS) REGULAR 表スペースよりもパフォーマンスが良くなります。SMS は TEMPORARY 表スペースによく使用されます。特に TEMPORARY 表スペースがとても小さい場合に使用されますが、その場合には時の経過とともに SMS のパフォーマンス上の利点は減少します。
- これまでは、DMS ロー・デバイス表スペースには DMS ファイル表スペースに対して実質的にかなりのパフォーマンス上の利点がありました。しかし直接入出力 (CREATE TABLESPACE ステートメントと ALTER TABLESPACE ステートメントの NO FILE SYSTEM CACHING 文節使用時のデフォルトになりました) が導入されたので、DMS ファイル表スペースは DMS ロー・デバイス表スペースとほとんど同じパフォーマンスを提供します。

重要: システム管理スペース (SMS) を使用する REGULAR 表スペースは推奨されおらず、今後のリリースでは、なくなる可能性があります。システム管理スペースを使用するカタログ表スペースと TEMPORARY 表スペースは非推奨にはなっていませんが、代わりにデータベース管理スペース (DMS) または自動ストレージ表スペース (AMS) を使用することが推奨されています。

初期 DB2 構成設定

DB2 構成アドバイザー (**AUTOCONFIGURE** コマンドとも呼ばれる) は指定した基本システム・ガイドラインを取り込み、DB2 構成値の適正な開始セットを判別します。**AUTOCONFIGURE** コマンドを使用するとデフォルト構成設定値に比べて実際の改善をもたらすので、初期構成値を取得するために推奨されている方法です。

AUTOCONFIGURE コマンドによって生成される推奨値を、システムの特성에基づいてさらに調整して洗練することが多くの場合に必要となります。

以下に、**AUTOCONFIGURE** コマンドの使用に関する提案を幾つか記します。

- DB2 バージョン 9.1 以降、**AUTOCONFIGURE** コマンドはデータベース作成時に自動的に実行されるようになっていますが、**AUTOCONFIGURE** コマンドを明示的に実行することが引き続き推奨されています。これは、システムからの結果をカスタマイズするのに役立つ、キーワード/値のペアを指定できるからです。
- データベースに適切な量のアクティブ・データが移植された後、**AUTOCONFIGURE** コマンドを実行 (または再実行) します。これにより、このツールにデータベースの性質に関する詳細情報が提供されます。データベースにデータを取り込む際のデータ量は重要です。なぜなら、この量はバッファ・プール・サイズの計算などに影響を与えるからです。データの量が多すぎたり少なすぎたりすると、こうした計算の正確さが低下します。
- **mem_percent**、**tpm**、および **num_stmts** などの重要な **AUTOCONFIGURE** コマンド・キーワードに種々の値を試し、こうした変更によって影響を受ける構成値、およびその影響度合いについて把握してください。
- 種々のキーワードと値を試みる場合、**APPLY NONE** オプションを使用します。これにより、推奨値と現在の設定値を比較することができます。
- デフォルト値がご使用のシステムに適さない場合もあるので、すべてのキーワードに値を指定します。例えば、**mem_percent** のデフォルトは 25% ですが専用の DB2 サーバーには低すぎるので、この場合には 85% が推奨値です。

DB2 オートノミックおよび自動パラメーター

DB2 データベース製品の最近のリリースでは、インスタンスまたはデータベースの起動時に自動的に設定されたり、操作時に動的に調整されたりするパラメーターがかなり増えました。ほとんどのシステムでは、細心の注意を払って手動で調整したごく一部のシステムを除いて、より良いパフォーマンスが自動設定によって提供されます。これは、特に DB2 セルフチューニング・メモリー・マネージャー (STMM) によるものです。STMM は、データベース・メモリーの合計割り振りや、DB2 システム内の 4 つの主要なメモリー・コンシューマー (バッファ・プール、ロック・リスト、パッケージ・キャッシュ、およびソート・ヒープ) を動的に調整します。

こうしたパラメーターは各パーティション単位で適用されるので、パーティション・データベース環境で STMM を使用する際には若干の注意を要します。パーティション・データベース・システムでは、STMM は (DB2 システムによって自動的に選択されるものの、選択対象をオーバーライドすることも可能な) 単一パーティション上のメモリー要件を継続的に測定し、STMM が使用可能なすべてのパーティションにヒープ・サイズ更新を「プッシュアウト」します。すべてのパーティションに対して同じ値が使用されるので、データ量、メモリー要件、およびアクティビ

ティーの全般的なレベルがすべてのパーティション間で全く一様なパーティション・データベース環境で STMM は最適に機能します。少数のパーティションに偏ったデータ・ボリュームや異なるメモリー要件がある場合には、こうしたパーティション上では STMM を使用不可にすべきであり、より統一の取れた調整が行えるようにしてください。例えば、一般に STMM はカタログ・パーティションでは使用不可にすべきです。

データ分布が偏っているパーティション・データベース環境の場合、クロス・クラスター・メモリー・チューニングを継続的に行うことは勧められておらず、以下のように、STMM を「チューニング・フェーズ」で選択的かつ一時的に使用し、良好なヒープ設定を手動で判別するのに役立てることができます。

- 「標準的な」1 つのパーティションで STMM を使用可能にします。他のパーティションでは、引き続き STMM を使用不可にします。
- メモリー設定を安定化させてから、STMM を使用不可にして、影響あるパラメーターをチューニング済みの値に手動で「固定」します。
- 同様のデータ・ボリュームとメモリー要件を持つ他のデータベース・パーティション (例えば、同じパーティション・グループ内のパーティション) にチューニング済みの値をデプロイします。
- システム内に類似したデータ・ボリュームとデータ・タイプが含まれ、類似の役割を実行するデータベース・パーティションのセットが別個に幾つもある場合には、この処理を繰り返します。

一般に、構成アドバイザーは該当する場合には、オートノミック設定を使用可能にすることを選択します。これには、(とても有用な) **RUNSTATS** コマンドによる自動統計更新が含まれます。ただし、自動再編成と自動バックアップは除外します。これらは大いに有用ですが、最適な結果を得るためにはご使用の環境とスケジュールに応じて構成する必要があります。自動統計プロファイルは、デフォルトによる使用不可のままにしておきます。このオーバーヘッドはとても大きいので、制御された条件下で、複雑なステートメントを使用する場合に一時的に用いることを意図しています。

明示的な構成設定

一部のパラメーターには自動設定値がなく、構成アドバイザーによって設定されることもありません。こうした場合、明示的に扱う必要があります。パフォーマンスへの影響があるパラメーターだけを、以下で考慮します。

- **logpath** または **newlogpath** は、トランザクション・ログの場所を判別します。構成アドバイザーでさえ、ユーザーのためにログの場所を決定することはできません。前述のように、最も重要な点は表スペースなどの他の DB2 オブジェクトとこのパラメーターはディスク装置を共有できず、データベース・パスの下にあるデフォルトの場所のままにすべきでもありません。理想としては、十分なスループット容量のある専用ストレージ上にトランザクション・ログを配置し、ボトルネックが作成されないようにしてください。
- **logbufsz** は、トランザクション・ロガーの内部バッファー・サイズを 4 KB のページ数で判別します。実稼働環境で良好なパフォーマンスを得るには、8 ページのみのデフォルト値では少なすぎます。構成アドバイザーは入力パラメーターに基づいて常にこのサイズを増やしますが、十分ではない可能性があります。

256 から 1000 ページの値が通常は良好な範囲で、データベース・サーバーの全体的なスキームにおけるメモリー合計にとってはごく小規模です。

- **mincommit** は、グループ・コミット を制御します。グループ・コミットは、DB2 システムによって n 個のコミット・トランザクションをまとめてバッチ化しよう とします。現在のトランザクション・ロガー設計でこれが望ましい動作である ことは、ごくまれです。 **mincommit** をデフォルト値の 1 のままにしてください。
- **buffpage** は、サイズ -1 に定義されている、各バッファー・プールに割り振ら れているページ数を判別します。ベスト・プラクティスは **buffpage** を無視し、 SYSCAT.BUFFERPOOLS 内にエントリーのあるバッファー・プールのサイズを明 示的に設定するか、STMM によってバッファー・プール・サイズを自動的にチュ ーニングするようにします。
- **diagpath** は、多様でかつ有用な DB2 診断ファイルの場所を判別します。これ は、パーティション・データベース環境ではパフォーマンスに影響を与える場合 があるかもしれませんが、通常それ以外ではほとんど影響を及ぼしません。すべ てのパーティションにおける **diagpath** のデフォルト場所は、通常、共有の NFS マウント・パス上にあります。ベスト・プラクティスは、**diagpath** を各パーテ ィションのローカルの非 NFS ディレクトリーにオーバーライドします。これによ り、すべてのパーティションにおいて、診断メッセージによって同じファイルを 更新しようとすることはありません。代わりに、各パーティションでローカルに 保持され、競合が大いに減少します。
- **DB2_PARALLEL_IO** は構成パラメーターではありませんが、DB2 レジストリー変数 の 1 つです。DB2 システムでは、ディスクのアレイで構成されるストレージを 使用するのがごく一般的です。これによりオペレーティング・システムに対して 単一のデバイスが提供されるか、複数のデバイスに渡るファイル・システムを使 用できます。その結果、デフォルトでは DB2 データベース・システムは表ス ペース・コンテナに対して一度に 1 つのプリフェッチ要求のみを行います。これ は、単一のデバイスに対する複数の要求が直列化された仕方で行われるという理 解に基づいています。ただし、コンテナがディスクのアレイ上にある場合、複 数のプリフェッチ要求を直列化せず同時にディスパッチする可能性があります。 これには、**DB2_PARALLEL_IO** が関係します。この変数は、プリフェッチ要求 を単一のコンテナに対して並列に発行できることを DB2 システムに指示しま す。最も簡単な設定は **DB2_PARALLEL_IO**=* (これは、すべてのコンテナが複数 ディスク上にあることを意味し、ここでは 7 つのディスクが想定される) です が、他の設定も並列性の度合い、および影響を受ける表スペースを制御します。 例えば、ご使用のコンテナが 4 つのディスクからなる RAID-5 アレイにあ り、**DB2_PARALLEL_IO** を *:3 に設定するかもしれません。特定の値によってパフ ォーマンス上の利点が生じるかどうかは、エクステンツ・サイズ、RAID セグメ ント・サイズ、および同じディスクのセットを使用しているコンテナ数によっ て異なります。

SAP および他の ISV 環境での考慮事項

DB2 データベース・サーバーを SAP などの ISV アプリケーション用に稼働して いる場合、特定のアプリケーションを考慮に入れたいくつかのベスト・プラクティ スの指針が当てはまることがあります。最も簡単な手段は DB2 レジストリー変数 **DB2_WORKLOAD** です。この変数を適切な値に設定することにより、特定の環境および ワークロードに合わせて集約レジストリー変数を最適化できます。 **DB2_WORKLOAD**

の有効な設定値としては、

1C、CM、COGNOS_CS、FILENET_CM、MAXIMO、MDM、SAP、TPM、WAS、WC、および WP があります。

コード・ページまたはコード・セット、および照合シーケンスなどには、他の推奨値とベスト・プラクティスが当てはまる場合があります。それらは事前判別された値に設定する必要があるからです。詳しくは、アプリケーション・ベンダーの資料を参照してください。

SAP Business One などの多くの ISV アプリケーションでは、初期構成を定義するために **AUTOCONFIGURE** コマンドを正常に使用できます。ただし、SAP NetWeaver インストールでは使用すべきではありません。DB2 構成パラメーターの初期セットが SAP インストール時に適用されるためです。また SAP には代わりとなる有力なベスト・プラクティス方法 (SAP Notes) があり、適した DB2 パラメーター設定が説明されています。一例として、「SAP Note 1086130 - DB2: DB2 9.5 Standard Parameter Settings」があります。

パーティション・データベース環境を使用する場合、SAP アプリケーションに特に注意を払ってください。SAP は、SAP NetWeaver Business Intelligence (Business Warehouse) 製品でパーティション・データベース環境を主に使用します。推奨されているレイアウトでは、パーティション 0 に、DB2 システム・カタログ、ディメンション表とマスター表、および SAP 基本表があります。これにより、このパーティションでのワークロードが、他のパーティション・データベース環境に比べて異なることとなります。SAP アプリケーション・サーバーはこのパーティションで稼働するので、このパーティションに対してだけ、最大 8 個のプロセッサを割り当てることができます。SAP BW ワークロードは並列化の度合いがより高くなり、同時に実行される小規模な照会が多くなるので、通常 SAP BI のためのパーティション数は他のアプリケーションに比べて少なくなります。つまり、データ・パーティションごとに複数の CPU が必要になります。

インスタンス構成

新しい DB2 インスタンスを開始するときには、基本構成を設定するための実行できる数多くのステップが存在します。

- 構成アドバイザーを使用して、バッファ・プール・サイズ、データベース構成パラメーター、およびデータベース・マネージャー構成パラメーターの初期値の推奨値を取得することができます。構成アドバイザーを使用するには、既存のデータベースに **AUTOCONFIGURE** コマンドを指定するか、または **CREATE DATABASE** コマンドに **AUTOCONFIGURE** パラメーターを指定します。推奨値は表示することができ、また **CREATE DATABASE** コマンドの **APPLY** パラメーターを使用して適用することもできます。推奨値は、ユーザーが提供する入力と、アドバイザーが収集するシステム情報に基づいています。
- データベース・マネージャーまたは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 (『構成パラメーター・サマリー』を参照) をご覧ください。これらのサマリー表には、特定の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、または小さく向上するか、または変化しないかを示す列が含まれています。こうした表を使用して、ご使用の環境でパフォーマンスを最も向上させるのに役立つ可能性のあるパラメーターを見つけてください。

- **ACTIVATE DATABASE** コマンドを使用して、データベースを活動化し、すべての必要なデータベース・サービスを始動します。そのようすると、データベースが接続でき、任意のアプリケーションで使用できるようになります。パーティション・データベース環境では、このコマンドはすべてのデータベース・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。

表スペースの設計

ディスク・ストレージのパフォーマンス要因

ディスク・ストレージ構成などのハードウェア特性が、システムのパフォーマンスに強い影響を与える場合があります。

パフォーマンスは、ディスク・ストレージ構成の以下の 1 つ以上の側面によって影響を受ける可能性があります。

- 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どのように適正に分割するかによって、さまざまな状況でシステムがどのように機能するかが大部分決まります。

- ディスク入出力の分散

複数の装置とコントローラーにディスク入出力要求をどれほどバランスよく分散するかによって、データベース・マネージャーがディスクからデータを取り出す速度に影響を与えます。

- ディスク・サブシステムのコア・パフォーマンス・メトリック

秒単位のディスク操作の数または秒単位で転送される容量 (MB 単位) は、システム全体のパフォーマンスに非常に強い影響を与えます。

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、照会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。

これらの特性には、次のものが含まれます。

- コンテナー特性

コンテナー特性は、照会の実行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、照会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** カタログ・ビューの 2 つの列が使用されます。

- **OVERHEAD** では、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) が示されます。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。

$$\text{OVERHEAD} = \text{ミリ秒単位の平均シーク時間} \\ + (0.5 * \text{回転待ち時間})$$

詳細は次のとおりです。

- 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 (ミリ秒単位) は以下のように計算されます。

$$(1 / \text{RPM}) * 60 * 1000$$

詳細は次のとおりです。

- 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。

例えば、ディスクの 1 分当たりの回転数が 7200 であるとし、回転待ち時間の公式を使用すると、次のようになります。

$$(1 / 7200) * 60 * 1000 = 8.328 \text{ ミリ秒}$$

この値は、平均シーク時間が 11 ミリ秒であると想定すると、次のようにオーバーヘッドの見積りに使用できます。

$$\text{OVERHEAD} = 11 + (0.5 * 8.328) \\ = 15.164$$

- TRANSFERRATE では、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) が示されます。

それぞれの表スペース・コンテナ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。

$$\text{TRANSFERRATE} = (1 / \text{spec_rate}) * 1000 / 1024000 * \text{page_size}$$

詳細は次のとおりです。

- 転送速度に関するディスクの規格 (1 秒当たりの MB 数) を示す *spec_rate* で除算し、1 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 (例えば、4 KB ページの場合は 4096 バイト)。

例えば、ディスクの規格速度が 1 秒当たり 3 MB であるとし、この場合は以下のようになります。

$$\text{TRANSFERRATE} = (1 / 3) * 1000 / 1024000 * 4096 \\ = 1.333248$$

つまり、ページ当たり約 1.3 ミリ秒になります。

表スペース・コンテナが単一の物理ディスクではなく、ディスク・アレイ (RAID など) である場合、TRANSFERRATE を見積も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、ボトルネックがディスク・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。しかしアレイが大きければ、ボトルネックがディスク・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。そのため、順次スキャン中に、実際の入出力速度 (MB 単位) を測定する必要があります。例えば、`select count(*) from big_table` の順次スキャンの結果は数メガバイトのサイズになる可能性があります。この場合、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。例えば、4 つのコンテナがある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 の順次入出力速度は、コンテナ当たり 25 MB となります。
TRANSFERRATE はページ当たり $(1 / 25) * 1000 / 1\,024\,000 * 4096 = 0.16$ 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は、それぞれ異なる物理ディスク上に存在する可能性があります。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 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。

- 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。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を減らすことができます。

オプティマイザーは、SYSCAT.TABLESPACES カタログ・ビューの PREFETCHSIZE 列および EXTENTSIZE 列の情報を使用して、発生するプリフェッチの量を見積もります。

- EXTENTSIZE を設定できるのは、表スペースを作成するときだけです。4 または 8 ページのエクステント・サイズで普通は十分です。
- PREFETCHSIZE は、表スペースを作成または変更するときを設定できます。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。このパラメーターのサイズ変更に関する推奨を検討して、必要な変更を行うか、AUTOMATIC に設定してください。

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前に、`runstats` ユーティリティーを実行することを考慮してください。これによって索引に関する最新の統計を収集し、可能な限り最適なデータ・アクセス・プランが照会オプティマイザーで選択されるようにします。

表

標準の表における表および索引の管理

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。

例えば、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。ほとんどのページにはユーザー・レコードだけが含まれています。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データ・サーバーによって使用されます。例えば、標準の表の場合、データ・ページには 500 ページごとにフリー・スペース制御レコード (FSCR) があります (69 ページの図 8)。これらのレコードは、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で使用可能なフリー・スペースの量をマップします。

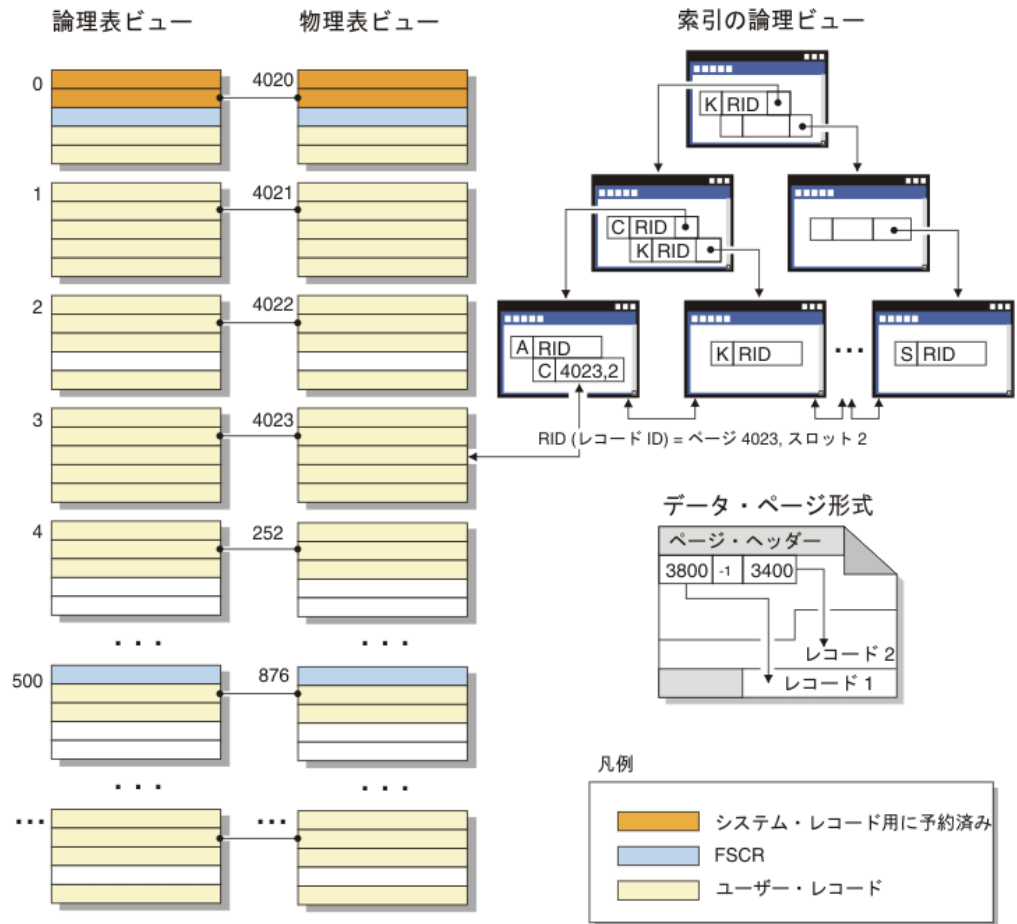


図8. 標準の表の論理表、レコード、および索引構造

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。データベース管理スペース (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。

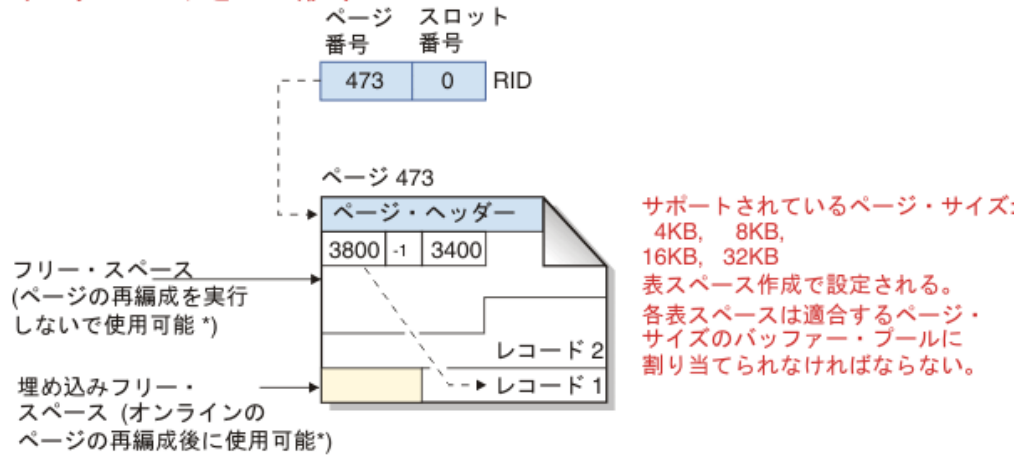
各データ・ページのフォーマットは同じです。ページはページ・ヘッダーで始まり、その後にスロット・ディレクトリーが続きます。スロット・ディレクトリーの各項目は、ページの異なるレコードに対応します。スロット・ディレクトリー内の項目は、レコードが始まるデータ・ページ上のバイト・オフセットを表します。-1の項目は、削除されたレコードに対応します。

レコード ID とページ

レコード ID は、ページ番号とそれに続くスロット番号で構成されます (70 ページの図9)。索引レコードには、ridFlag という追加のフィールドが含まれます。ridFlag は、キーに削除済みのマークが付けられているかなど、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識別するのに使用されま

す。レコードに割り当てられた RID は、表が再編成されるまで変わりません。

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9. 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時、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。

DB2 データ・サーバーでは、さまざまなページ・サイズがサポートされます。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。例えば、順次アクセスが意思決定支援アプリケーションに多用される場合や、一時表が集中的に使用される場合などです。行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。例えば、ランダム・アクセスはオンライン・トランザクション処理 (OLTP) 環境でよく使用されます。

標準の表における索引管理

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。B ツリー索引はページの均衡の取れた階層として調整され、項目が挿入または削除されるとデータ・キーを再調整することによって、アクセス時間を最短にします。

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。これは、索引キーの上位 10 パーセントが新しいページに配置されることを意味します。このタイプの索引ページの分割は、新しい上位キー値を使用して挿入操作が頻繁に実行されるワークロードの場合に役立ちます。

表に X ロックがある場合のみ、索引ページから削除済み索引キーが除去されます。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。

索引作成時に、MINPCTUSED に正の値を指定して、オンライン索引デフラグを使用可能にした場合、索引リーフ・ページはオンラインでマージできます。

MINPCTUSED 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トを表します。キーを除去した後に索引ページでスペースの使用量がこの値を下回ると、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・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、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いければ、それは行われません。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。MINPCTUSED の推奨値は、50 % 以下です。

CREATE INDEX ステートメントの INCLUDE 節を使用すると、索引リーフ・ページに (キー列より多い) 1 つ以上の列を指定できます。索引 B ツリーに対する配列操作が関係しないこれらの組み込み列では、索引のみのアクセスで適格である照会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、キー列の更新のコストよりも低いですが、索引の一部では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。

MDC 表および ITC 表のための表と索引の管理

マルチディメンション・クラスタリング (MDC) 表と挿入時クラスタリング (ITC) 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きます。

標準の表と同様に、MDC 表と ITC 表は列に分割されたデータの行を含むページに編成されます。各ページの行は、レコード ID (RID) で識別されます。ただし、MDC 表と ITC 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。例えば、72 ページの図 10 はエクステント・サイズが 4 の表を示しています。0 から 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 から 7 の番号が付けられた次の 4 ページは、表の 2 番目のブロックとなります。

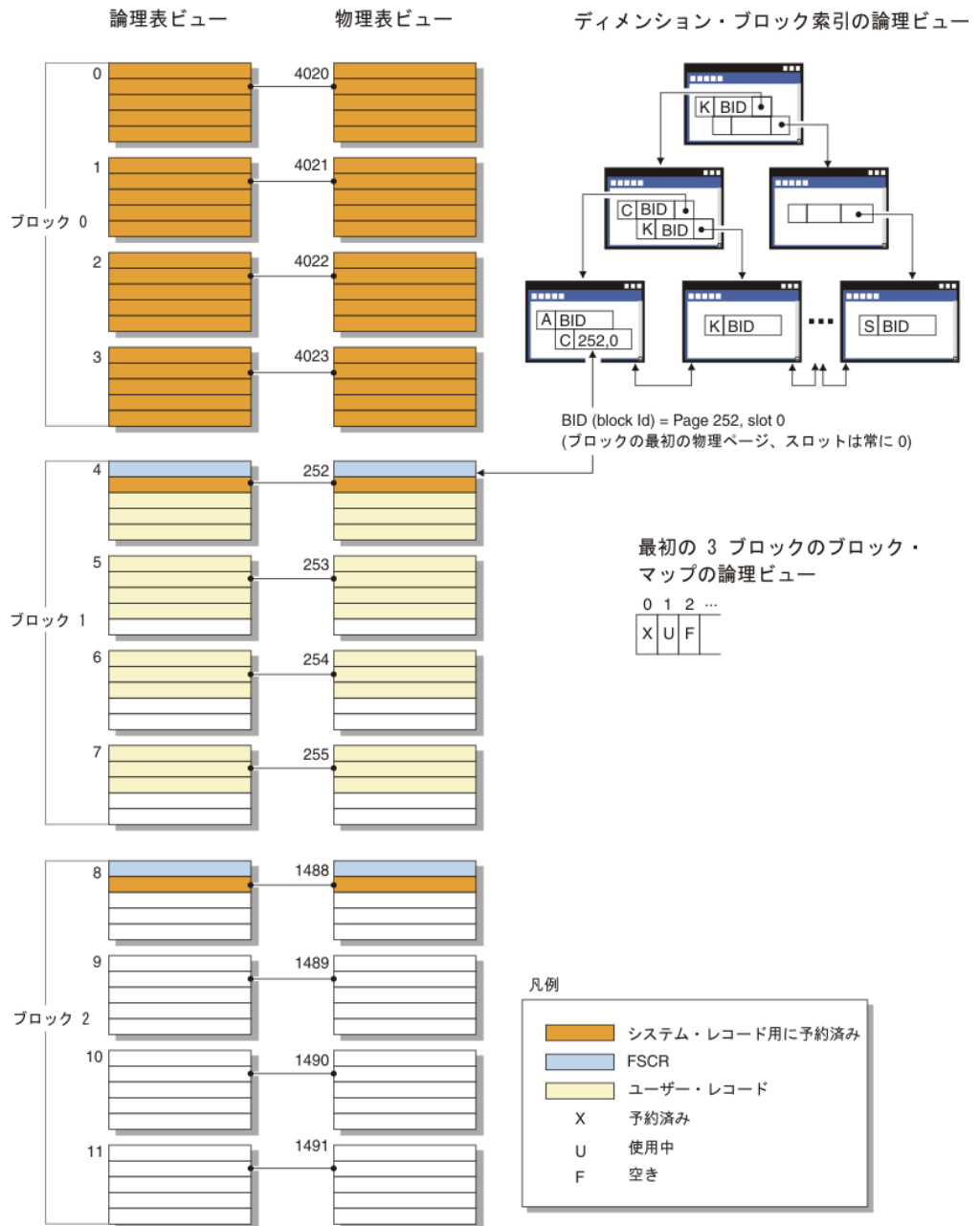


図 10. MDC および ITC 表の論理表、レコード、および索引構造

最初のブロックには、DB2 サーバー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 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 つ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。
- 複合ブロック索引。これには、すべてのディメンションのキー列が含まれ、挿入と更新アクティビティの際にクラスタリングを保守するために使用されます。

オブティマイザーは、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・ブロック索引を使用するアクセス・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、オブ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、これにより入出力が最小になり、パフォーマンスが向上します。

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。

その名前が示すとおり、ITC 表は、データを行の挿入時刻に基づいてクラスタ化します。MDC 表と ITC 表の違いは次のとおりです。

- どのようなデータ・アクセスでも、ブロック索引は使用されない。
- 表に対して単一の複合ブロック索引しか作成されず、その索引は 1 つの仮想ディメンションから構成される。
- 索引がオブティマイザーによってプランに選択されることはない。これは、索引に含まれる列を SQL ステートメントによって参照することができないためです。

MDC 表と ITC 表は、表スペースに空のブロックを解放できます。

索引

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、74 ページの図 11 に示されているように、いくつかのレベルがあります。ここで「rid」はレコード ID (RID) を意味します。

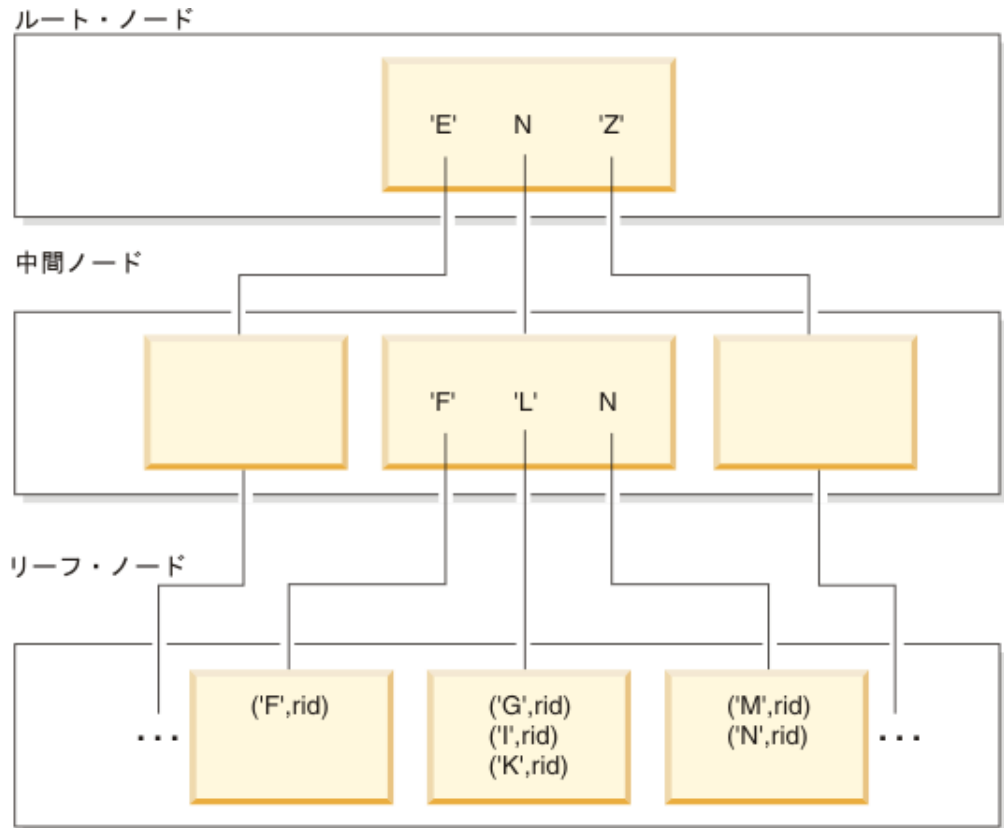


図 11. B+ ツリー索引の構造

トップレベルはルート・ノードと呼ばれます。最下位レベルはリーフ・ノードで構成され、ここに索引キー値と、対応するデータが入っている表の行へのポインタが保管されます。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・ノードには、その次のレベルにある(中間)ノードごとに1つのキーがあります。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。例えば、図に示すように、索引に3つのレベルがあるとします。特定の索引キー値を検出するために、索引マネージャーは、ルート・ノードから、検索キー値以上のキー値のうちの最初のものを探します。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、各中間ノードでこの手順を行います。

図 11 で検索しているキーは「I」であるとします。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 RID が見つかる特定のリーフ・ノードを指しています。その RID 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が入っている場合もあります。こうしたポインタによって、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** オプション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) 表または挿入時クラスタリング (ITC) 表の場合、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。複合ブロック索引も 1 つ作成されます。この索引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。それらの索引には 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。

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* は、論理的に削除されたとして RID にマークを付け、後で物理的に除去できるようにする目的で使用されます。更新または削除操作のコミット後、削除済みとマークされたキーは除去できます。索引内のそれぞれの可変長列について、追加の 2 バイトで列の値の実際の長さを保管します。

索引のクリーンアップおよび保守

索引の作成後、索引をコンパクトかつ整理された状態に保たないと、時間とともにパフォーマンスが低下する恐れがあります。

以下の推奨事項は、索引を可能な限り小さく、効率的に維持するのに役立ちます。

- オンライン索引デフラグを使用可能にする。

MINPCTUSED 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。

- コミットを頻繁に実行します。コミット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レベルの 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、コミット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、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** コマンドを使用すると、索引や表をいつ再編成すべきか、また **CLEANUP** パラメーターを使っていつ **REORG INDEXES** コマンド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** オプションを指定して **REORG INDEXES** コマンドを使用します。

DB2 バージョン 9.7 フィックスパック 1 以降のリリースの場合、**REORG INDEXES** コマンドを **ON DATA PARTITION** パラメーターを使用してデータ・パーティション表で発行し、指定のパーティションのパーティション索引を再編成します。索引の再編成の際、影響を受けないパーティションは引き続き読み取れますが、書き込み可能なアクセスは影響を受けるパーティションにのみ限定されます。

- 目的がスペースの再利用である場合は、**REORG INDEXES** コマンドに **CLEANUP** および **RECLAIM EXTENTS** パラメーターを指定して使用してください。**CLEANUP** パラメーターは、再利用可能なスペースの量を最大化します。

ADMIN_GET_INDEX_INFO 関数の RECLAIMABLE_SPACE 出力に、再利用可能なスペース量 (キロバイト単位) が示されます。

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、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。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** パラメーターを使って **REORG INDEXES** コマンドを実行するとき

CLEANUP 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 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、未削除のレコード ID (RID) を含むページから、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、最低でも PCTFREE のフリー・スペースがマージ後のリーフ・ページに確保されるような場合、隣接するリーフ・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ます。デフォルトの PCTFREE 値は 10 パーセントです。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。

データ・パーティション表の場合、非同期索引クリーンアップが完了した後に **RUNSTATS** コマンドを呼び出すことが推奨されています。表にデータタッチされたデータ・パーティションが存在しているかどうかを判別するには、

SYSCAT.DATAPARTITIONS カタログ・ビューで **STATUS** フィールドを照会し、値「L」(論理的データタッチ済み)、「D」(マテリアライズ照会表のようなデータタッチ従属表を持つデータタッチされたパーティション)、または「I」(索引クリーンアップ) を見つけます。

- 索引が再作成される時 (または、データ・パーティション索引の場合、索引パーティションが再作成される時)

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。

- デフォルトの **REBUILD** パラメーターを使用する **REORG INDEXES**
- **REORG TABLE (INPLACE** または **RECLAIM EXTENTS** パラメーターを使用しない)
- **IMPORT (REPLACE** パラメーターを使用する)
- **LOAD (INDEXING MODE REBUILD** パラメーターを使用する)

非同期索引クリーンアップ

非同期索引クリーンアップ (AIC) は、索引項目を無効にする操作の後に行われる、索引の据え置きクリーンアップです。索引のタイプに応じて、項目はレコード ID (RID) またはブロック ID (BID) 別にできます。バックグラウンドで非同期的に操作される索引クリーナーによって、無効な索引項目は除去されます。

AIC は、データ・パーティションをパーティション表からデタッチするプロセスを促進し、パーティション表に 1 つ以上の非パーティション索引があると開始されます。この場合、AIC はデタッチされたデータ・パーティションおよび疑似削除された項目を参照するすべての非パーティション索引項目を除去します。すべての索引が消去された後に、デタッチされたデータ・パーティションに関連する ID がシステム・カタログから除去されます。DB2 バージョン 9.7 フィックスパック 1 以降のリリースでは、非同期パーティションのデタッチ・タスクによって AIC が開始されます。

DB2 バージョン 9.7 フィックスパック 1 より前のリリースでは、パーティション表に従属マテリアライズ照会表 (MQT) がある場合、SET INTEGRITY ステートメントが実行されるまで AIC は開始されません。

通常の表アクセスは、AIC の進行中に維持されます。索引にアクセスする照会は、まだ消去されていない無効な項目を無視します。

多くの場合、パーティション表に関連付けられた各非パーティション索引に対して 1 つのクリーナーが開始されます。AIC タスクを適切な表パーティションに分散し、データベース・エージェントを割り当てる責任を担っているのは、内部タスク分散デーモンです。分散デーモンとクリーナー・エージェントは内部システム・アプリケーションで、LIST APPLICATIONS コマンド出力にそれぞれ db2taskd と db2aic というアプリケーション名で表示されます。偶然の中断を防ぐため、システム・アプリケーションを強制終了することはできません。データベースがアクティブな限り、分散デーモンはオンラインのままです。クリーニングが完了するまでは、クリーナーもアクティブなままです。クリーニングの進行中にデータベースを非アクティブにすると、データベースの再活動時に AIC が再開します。

パフォーマンスへの AIC の影響

AIC がパフォーマンスに与える影響はごくわずかです。

疑似削除された項目がコミット済みかどうかを判別するには、瞬時の行ロック・テストが必要です。しかし、ロックは決して獲得されないため、並行性には影響ありません。

各クリーナーは最小の表スペース・ロック (IX) および表ロック (IS) を獲得します。それらのロックは、他のアプリケーションがロックを待機中であるとクリーナーが判別した際に解放されます。このことが生じると、クリーナーは一時的に 5 分間処理を中断します。

クリーナーは、ユーティリティー・スロットル機能とも統合されています。デフォルトでは、各クリーナーには 50 のユーティリティー影響優先度があります。この優先度は、SET UTIL_IMPACT_PRIORITY コマンドまたは db2UtilityControl API を使用して変更することができます。

AIC のモニター

AIC は、**LIST UTILITIES** コマンドでモニターできます。それぞれの索引クリーナーは、出力に別個のユーティリティとして表示されます。以下は、**LIST UTILITIES SHOW DETAIL** コマンドによる出力の例です。

```
ID = 2
Type = ASYNCHRONOUS INDEX CLEANUP
Database Name = WSDB
Partition Number = 0
Description = Table: USER1.SALES, Index: USER1.I2
Start Time = 12/15/2005 11:15:01.967939
State = Executing
Invocation Type = Automatic
Throttling:
  Priority = 50
Progress Monitoring:
  Total Work = 5 pages
  Completed Work = 0 pages
  Start Time = 12/15/2005 11:15:01.979033

ID = 1
Type = ASYNCHRONOUS INDEX CLEANUP
Database Name = WSDB
Partition Number = 0
Description = Table: USER1.SALES, Index: USER1.I1
Start Time = 12/15/2005 11:15:01.978554
State = Executing
Invocation Type = Automatic
Throttling:
  Priority = 50
Progress Monitoring:
  Total Work = 5 pages
  Completed Work = 0 pages
  Start Time = 12/15/2005 11:15:01.980524
```

この場合、**USERS1.SALES** 表で作動する 2 つのクリーナーがあります。1 つのクリーナーは索引 **I1** を処理し、もう 1 つは索引 **I2** を処理します。「進捗モニター」セクションには、クリーニングが必要な索引ページの見積もり合計数と、クリーンな索引ページの現行数が示されます。

State フィールドは、クリーナーの現行状態を示します。通常の状態は「実行中」ですが、使用可能なデータベース・エージェントにクリーナーが割り当てられるのを待機している場合、またロック競合のためにクリーナーが一時的に中断している場合には、「待機中」状態になる場合があります。

それぞれのデータベース・パーティションは、各データベース・パーティション上で実行されているタスクに限り **ID** を割り当てるので、異なるデータベース・パーティションの異なるタスクであっても、ユーティリティ **ID** が同じになる場合があります。ことに注意してください。

MDC 表の非同期索引クリーンアップ

非同期索引クリーンアップ (AIC) を使用すると、マルチディメンション・クラスタリング (MDC) 表から条件を満たすデータのブロックを削除するのに効果的な方法であるロールアウト削除のパフォーマンスを向上させることができます。AIC は、索引項目を無効にする操作の後に行われる、索引の据え置きクリーンアップです。

標準のロールアウト削除の際に、索引は同期的にクリーンアップされます。レコード ID (RID) 索引が表に数多く含まれている場合、かなりの時間が、削除している表の行を参照している索引キーを除去するために費やされます。削除操作のコミット後にそのような索引をクリーンアップするように指定すると、ロールアウトの速度を速めることが可能です。

MDC 表での AIC の利点を生かすには、**据え置き索引表クリーンアップ・ロールアウト** というメカニズムを明示的に有効にする必要があります。据え置きロールアウトを指定するには 2 つの方法があり、**DB2_MDC_ROLLOUT** レジストリー変数を DEFER に設定する方法や、**SET CURRENT MDC ROLLOUT MODE** ステートメントを発行する方法です。据え置き索引クリーンアップ・ロールアウト操作中は、トランザクションがコミットされるまでは RID 索引に対する更新は行われず、ブロックにはロールアウトというマークが付けられます。行レベルの処理は必要ないため、削除操作中にブロック ID (BID) 索引がクリーンアップされます。

データベースがシャットダウンされるなどしてロールアウト削除がコミットされるか、データベースの再始動後に表に最初にアクセスすると、AIC ロールアウトが起動されます。AIC が進行中でも、クリーンアップ中の索引へのアクセスを含め、索引に対する照会は成功します。

1 つの MDC 表に対して 1 つの調整クリーナーが存在します。複数のロールアウトが関係する索引クリーンアップはその 1 つのクリーナーに統合され、クリーナーはそれぞれの RID 索引用のクリーンアップ・エージェントを作成します。クリーンアップ・エージェントは、同時に複数の RID 索引を更新します。またクリーナーは、ユーティリティー・スロットル機能とも統合されています。デフォルトでは、各クリーナーには 50 のユーティリティー影響優先度があります (許容される値は 1 から 100 までで、0 はスロットルなしを示します)。この優先度は、**SET UTIL_IMPACT_PRIORITY** コマンドまたは db2UtilityControl API を使用して変更することができます。

注: DB2 バージョン 9.7 以降のリリースでは、パーティション化された RID 索引を含むデータ・パーティション MDC 表において据え置きクリーンアップ・ロールアウトはサポートされません。サポートされるのは、NONE モードと IMMEDIATE モードだけです。 **DB2_MDC_ROLLOUT** レジストリー変数が DEFER に設定されている場合、または **DB2_MDC_ROLLOUT** の設定をオーバーライドするために **CURRENT MDC ROLLOUT MODE** 特殊レジスターが DEFERRED に設定されている場合、クリーンアップ・ロールアウト・タイプは IMMEDIATE になります。

MDC 表に対して非パーティション化された RID 索引のみ存在する場合には、据え置き索引クリーンアップ・ロールアウトがサポートされます。MDC ブロック索引は、パーティション化または非パーティション化することができます。

据え置き索引クリーンアップ・ロールアウト操作の進行状況のモニター

MDC 表でロールアウトされたブロックはクリーンアップが完了するまでは再利用できないので、据え置き索引クリーンアップ・ロールアウト操作の進行をモニターするのは有用です。 **LIST UTILITIES** コマンドを使用して、クリーンアップ中の各索引のユーティリティー・モニター項目を表示します。また、**ADMIN_GET_TAB_INFO** 表関数または **GET SNAPSHOT** コマンドを使用すると、ロー

ルアウト削除後に非同期クリーンアップが保留されている (BLOCKS_PENDING_CLEANUP)、データベース内の MDC 表ブロックの合計数を取得できます。

以下の **LIST UTILITIES SHOW DETAIL** コマンドの出力例では、クリーンアップされた各索引のページ数の進行状況が示されています。それぞれのフェーズが 1 つの RID 索引を表します。

```
ID = 2
Type = MDC ROLLOUT INDEX CLEANUP
Database Name = WSDB
Partition Number = 0
Description = TABLE.<schema_name>.<table_name>
Start Time = 06/12/2006 08:56:33.390158
State = Executing
Invocation Type = Automatic
Throttling:
  Priority = 50
Progress Monitoring:
  Estimated Percentage Complete = 83
  Phase Number = 1
    Description = <schema_name>.<index_name>
    Total Work = 13 pages
    Completed Work = 13 pages
    Start Time = 06/12/2006 08:56:33.391566
  Phase Number = 2
    Description = <schema_name>.<index_name>
    Total Work = 13 pages
    Completed Work = 13 pages
    Start Time = 06/12/2006 08:56:33.391577
  Phase Number = 3
    Description = <schema_name>.<index_name>
    Total Work = 9 pages
    Completed Work = 3 pages
    Start Time = 06/12/2006 08:56:33.391587
```

オンライン索引のデフラグ

索引のリーフ・ページにおける使用済みスペースの最小量を表すユーザー定義可能なしきい値 **MINPCTUSED** 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。ページ上に十分なスペースが存在し、2 つの隣接するページをマージできる場合は、即時にマージが行われます。その結果として生じた空の索引リーフ・ページは、その後削除されます。

MINPCTUSED 文節は、**ALTER INDEX** ステートメントでは変更できません。既存の索引に、オンライン索引デフラグによるマージ機能が必要となった場合は、その索引をいったんドロップしてから **MINPCTUSED** 節を指定した **CREATE INDEX** ステートメントを使用して再作成する必要があります。オンライン索引デフラグを使用可能にする場合は、隣接するページが検査されるときにそれらのページをマージできる可能性を高めるため、**MINPCTUSED** に 50 より小さい値を設定する必要があります。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。 **MINPCTFREE** の設定の有無によって、その索引に対し **REORG CLEANUP** を実行できるかどうか左右されるわけではありません。

MINPCTFREE に 1 から 50 の低い値を設定すると、実行時に自動的に実行されるページのマージが多くなるため、REORG CLEANUP に残される作業は削減される可能性があります。

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。削除済みのページをデータベース管理スペース (DMS) ストレージ・モデル内の他のオブジェクト用に解放するために、REBUILD または RECLAIM EXTENTS という 2 つの再編成オプションがあります。システム管理スペース (SMS) ストレージ・モデルの場合は、REORG REBUILD のみ許可されます。RECLAIM EXTENTS は、ページを移動して、削除済みのページから成るフル・エクステントを作成し、その後それらを解放します。REBUILD は、PCTFREE を考慮しながら索引を可能な限り小さく最初から再作成します。

索引のレベル数に対処できるのは、REBUILD のみです。索引のレベル数を減らすことが目的である場合は、REBUILD オプションを指定して再編成を実行してください。

表に X ロックがある場合、キー削除中にキーはページから物理的に除去されます。この場合、オンライン索引デフラグは有効です。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ず、索引デフラグは試行されません。

MINPCTUSED の値にかかわらず索引をデフラグするには、REORG INDEXES コマンドを CLEANUP ALL オプションを指定して呼び出します。索引全体が検査され、可能な場合はいつでも、隣接する 2 つのリーフ・ページがマージされます。このマージが可能となるのは、マージされるページに少なくとも PCTFREE のフリー・スペースが残っている場合です。PCTFREE は索引作成時に指定できます。デフォルト値は 10 (パーセント) です。

パフォーマンスを向上させるためのリレーショナル索引の使用

索引を使うと、表データへアクセスするときのパフォーマンスを向上させることができます。リレーショナル・データのアクセスにはリレーショナル索引が使用され、XML データのアクセスには XML データに対する索引が使用されます。

照会最適化はリレーショナル表データにアクセスするためにリレーショナル索引を使用するかどうかを決定しますが、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するのはユーザーに任されています。この唯一の例外は、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。

リレーショナル索引を作成した後、またはプリフェッチ・サイズを変更した後は、runstats ユーティリティを実行して、新しい索引統計を収集してください。runstats ユーティリティ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、最適化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。

特定のパッケージでリレーショナル索引を使用するかどうかを決めるには、Explain 機能を使用します。1 つ以上の SQL ステートメントによって活用される可能性の

あるリレーショナル索引についてのアドバイスを入手するには、**db2advise** コマンドを使用して設計アドバイザを起動します。

IBM® InfoSphere Optim™ Query Workload Tunerは、単一の SQL ステートメントのパフォーマンスと SQL ステートメント・グループのパフォーマンス (照会ワークロードと呼ばれる) を向上するツールを提供しています。この製品について詳しくは、<http://www.ibm.com/software/data/optim/query-workload-tuner-db2-luw/index.html>の製品概要のページを参照してください。また、バージョン 3.1.1 以降では、ワークロード設計アドバイザを使用して、DB2 設計アドバイザ・ウィザードで利用できた多くの操作を実行できます。詳しくは、<http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.datatools.qrytune.workloadtunedb2luw.doc/topics/genrecsdsgn.html>のワークロード設計アドバイザの資料を参照してください。

リレーショナル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、SQL 照会において参照される表ごとに、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、表が大きいほどそのようなスキャンにかかる時間も長くなります。表スキャンは、表の中のほとんどの行を必要とする複雑な照会に効果的です。他方、索引スキャンは、表の行の一部だけを戻す照会で、表の行に効果的にアクセスすることができます。

オプティマイザは、リレーショナル索引列が 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。索引ファイルは一般的に表全体よりも小さく、読み取りに必要な時間も少なく済みます。特に表が大きい場合はその違いが影響します。さらに、索引全体をスキャンする必要がない場合もあります。索引に述部を適用すると、データ・ページから読み取らなければならない行数が減ります。

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、ソート操作をしなくても正しい順序で行を検索できます。照会されている表にリレーショナル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らない点に注意してください。ORDER BY 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になります。

リレーショナル索引に include 列を含めることもできます。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、オプティマイザは表そのものにアクセスする必要なく、索引のみから必要な情報を取得することができます。

リレーショナル索引を使用しない場合と比べての索引の欠点

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。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。

- 各索引には、ストレージ・スペースが必要になります。正確な量は、表のサイズとリレーショナル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挿入または削除の各操作では、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それぞれの更新操作についても同じことが言えます。
- 各リレーショナル索引は、 옵ティマイザーが考慮するアクセス・プランの別の候補を示すため、照会のコンパイル時間が長くなります。

リレーショナル索引の計画のヒント

効果的な索引を設計すると、照会においてリレーショナル・データに容易にアクセスできるようになります。

設計アドバイザー (**db2adv** コマンド) を使用して、特定の照会またはワークロードを定義する一連の照会に最適な索引を調べてください。このツールは、**include** 列や、逆方向スキャンに使用できる索引など、パフォーマンスを向上させる機能を推奨します。

有用なリレーショナル索引を作成する上で、以下の指針も役立つでしょう。

- データの効率的な検索
 - 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検索効率が向上する場合
 - 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

例えば、

```
create unique index idx on employee (workdept) include (lastname)
```

LASTNAME を索引キーの一部としてではなく、**include** 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の意味します。

- 頻繁に実行される照会の **WHERE** 節の中で使用される列には、リレーショナル索引を作成します。

以下の例では、**WORKDEPT** 列が多く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 **WHERE** 節の効率が良くなると考えられます。

```
where workdept='A01' or workdept='E21'
```

- 照会で参照されるそれぞれの列の名前を指定する複合キーを持つリレーショナル索引を作成します。この方法で索引を作成すると、リレーショナル・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。

例えば次の照会を考えてみましょう。

```
select lastname
from employee
where workdept in ('A00','D11','D21')
```

EMPLOYEE 表の **WORKDEPT** 列と **LASTNAME** 列にリレーショナル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、

照会をより効率的に処理できる可能性があります。述部は `WORKDEPT` を参照するため、この列はリレーショナル索引の最初のキー列でなければなりません。

- 表の効率的な検索

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` ステートメントで `ALLOW REVERSE SCANS` オプション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ンのほうが、逆方向のスキンよりも少し効率的です。

- より大きな表への効率的なアクセス

リレーショナル索引を使用して、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。このページ数は、`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。

- 更新または削除操作のパフォーマンスの向上

- 親表に対するそのような操作のパフォーマンスを向上させるには、外部キーについてリレーショナル索引を作成します。
- `REFRESH IMMEDIATE` と `INCREMENTAL` マテリアライズ照会表 (MQT) に対するそのような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 定義の `GROUP BY` 節内の列で構成される) について固有のリレーショナル索引を作成します。

- 結合のパフォーマンスの向上

複数列リレーショナル索引の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 (`expression1 = expression2`) で指定されることの最も多い列か、明確な値の数が最も多い列を最初のキー列として使用します。

- ソート

- ソート操作を速くするには、リレーショナル・データのソート時に頻繁に使用される列についてリレーショナル索引を作成します。
- 一部のソートを回避するには、可能であれば `CREATE INDEX` ステートメントを使用して、主キーおよびユニーク・キーを定義します。
- 頻繁に実行される照会で必要とされる順序に行を配列するため、リレーショナル索引を作成します。順序付けは、`DISTINCT`、`GROUP BY`、および `ORDER BY` 節で必要になります。

以下の例では、`DISTINCT` 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` 列に対して定義された索引を使用して重複値を除去できます。以下の `GROUP BY` 節を使用する例のようにして、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
```


- 新しく挿入された行のクラスタリングの保持およびページ分割の回避

クラスタリング索引を定義すると、表を再編成する必要性が大幅に少なくなります。CREATE TABLE ステートメントで PCTFREE オプションを使用し、行を適切に挿入できるようにどれだけのフリー・スペースを各ページに残すかを指定します。LOAD コマンドに pagefreespace ファイル・タイプ修飾子を指定することもできます。

- 索引保守コストとストレージ・スペースの節約

- 既存の他の索引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。例えば、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 についての別の索引は通常有用ではありません。
- 根拠もなく多くの列に索引を作成しないでください。必要のない索引はスペースを浪費するだけでなく、準備時間を増やす原因ともなります。
- オンライン・トランザクション処理 (OLTP) 環境では、1 つの表に作成する索引は 1 つまたは 2 つにします。
- 読み取り専用照会環境では、1 つの表に 5 つを超える索引を作成することも可能です。

注: さまざまな述部を使用した多くの随時照会が行われるワークロードでは、索引ギャップ・カーディナリティーが小さく、索引ギャップの後の非索引ギャップの選択度が小さいため、表に小さい索引を多数作成するのではなく、大きな複合索引を少数作成するほうが適しているでしょう。

- 混合照会および OLTP 環境では、1 つの表に 2 つから 5 つの索引が妥当でしょう。
- オンライン索引デフラグの使用可能化

リレーショナル索引の作成時に MINPCTUSED オプションを使用します。MINPCTUSED は、オンライン索引デフラグを使用可能にし、索引リーフ・ページ上で使用できなければならないスペースの最小量を指定します。

リレーショナル索引のパフォーマンスのヒント

リレーショナル索引を効率的に活用するために取ることのできる処置がいくつかあります。

- 大規模なユーティリティー・ヒープを指定する

リレーショナル索引を作成中または再編成中の表に対して多くの更新アクティビティーが実行されると予想される場合には、これらの操作を高速化する上で役立つ大きなユーティリティー・ヒープ (util_heap_sz データベース構成パラメーター) を構成することを検討してください。

- 対称型マルチプロセッサ (SMP) 環境でのソート・オーバーフローを回避するには、sheapthres データベース・マネージャー構成パラメーターの値を増やします。
- リレーショナル索引には別々の表スペースを作成する

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し、索引表スペースを別のバッファ・プールに割り当てることもできます。このようにすると、索引ページはデータ・ページと競合することがないため、バッファに長く保持される可能性があります。

索引用に別の表スペースを使用する場合は、索引用のその表スペースの構成を最適化できます。索引は一般に表より小さく、分散しているコンテナも少ないため、通常は索引のエクステント・サイズの方が小さくなります。照会オプティマイザーは、アクセス・プランを選択するときに、表スペースが入っている装置の速度を考慮します。

- クラスタリングの度合いが高くなるようにする

SQL ステートメントが結果の順序付けを必要としているとき (例えばそこに ORDER BY、GROUP BY、または DISTINCT 節が含まれるとき)、以下の場合にはオプティマイザーが選択可能な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。特定の索引におけるクラスタリングの度合い情報については、SYSCAT.INDEXES カタログ・ビューの CLUSTERRATIO 列および CLUSTERFACTOR 列を照会してください。
- 表が小さいので、表のスキャンや結果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は、runstats ユーティリティーによって収集される CLUSTERRATIO または CLUSTERFACTOR 統計を向上させて、データの特定順序を維持しようと試みます。クラスタリング索引を作成した後、オフラインで表の REORG 操作を実行します。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。

表の PCTFREE 値は、ページ上に将来データが挿入される場合のために空にしておくスペースの量を指定します。これは、その挿入されたデータを適切にクラスタ化できるようにするために使用されます。表に PCTFREE 値を指定しない場合、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。

データ・クラスタリングは、範囲がクラスタ化された表の場合を除き、更新操作中に維持されません。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、更新操作を使用する代わりに、レコードを削除してからレコードの更新バージョンを挿入します。

- 表および索引の統計を最新のものにしておく

新規リレーショナル索引を作成した後は、runstats ユーティリティーを実行して索引統計を収集してください。それらの統計は、索引の使用によってデータ・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ときに役立ちます。

- 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、リレーショナル索引の MINPCTUSED がゼロより大きい値に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・スペースの大きさが指定された MINPCTUSED 値未満になった場合に索引リーフ・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。

- 必要に応じてリレーショナル索引を再編成する

索引から最高のパフォーマンスを得るには、それら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、索引をドロップして再作成するか、REORG ユーティリティを使用します。

頻繁に再編成を行う必要を少なくするためには、CREATE INDEX ステートメントに適切な PCTFREE 値を指定して、各索引のリーフ・ページ作成時に十分のフリー・スペースを残しておくようにします。将来の活動で、索引ページ分割の可能性が低くなるようにレコードを索引に挿入できます。ページ分割が生じるとページの隣接性が悪くなるため、索引ページ・プリフェッチの効率が悪くなります。リレーショナル索引の作成時に指定する PCTFREE 値は、索引が再編成されるときにも保持されます。

- リレーショナル索引使用に関する 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 ステートメントを発行して、それぞれのリレーショナル索引が最低でも 1 回は使用されているかどうかを検査します。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、スキャン中の大きな表がネスト・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を「VOLATILE (揮発性)」として宣言する

VOLATILE 表 とは、実行時のカーディナリティーが大きく異なる場合がある表のことを言います。この種類の表においては、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。

そのような表を揮発性として宣言するには、ALTER TABLE ステートメントで **VOLATILE** 節を使用します。以下の状況では、オプティマイザーは、統計に関係なく、そのような表に対する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の一部である場合
- 索引が索引スキャン中に述部を適用できる場合

型付き表の場合、ALTER TABLE...VOLATILE ステートメントの使用は、型付き表階層のルート表のみでサポートされます。

競合問題の改善のために DB2 pureScale環境で CURRENT MEMBER デフォルト値を使用する

DB2 pureScale環境では、列のデフォルト値を CURRENT MEMBER 特殊レジスターに設定することができます。これを設定すると、そのメンバー情報を使用して表または索引を範囲パーティション化することができるため、データベースの競合が削減されます。

以下の各シナリオは、CURRENT MEMBER 列を使用する新規索引を作成することによってデータベースの競合問題が改善される状況の概要を示すものです。このような新規索引を作成すれば、DB2 pureScaleクラスターは、表の索引を参照するときメンバー番号情報を利用できるため、メンバー間でのアクティブな共有量を削減することができます。このリソースの削減により、DB2 pureScale環境の速度および全体的なパフォーマンスを改善できます。

ユニークな増加シーケンス値またはタイム・スタンプの挿入

増え続ける一続きの値シーケンス（購入注文番号など）を格納するために、表の 1 行が使用されることがあります。これらの値は、ID 列として自動的に生成される場合もありますし、外部アプリケーションを使用してプログラムで生成される場合もあります。しかし、このような挿入がかなりの頻度で発生すると、競合の問題が生じてユーザー照会の速度が低下する可能性があります。

照会のパフォーマンスを改善するために、新しい CURRENT MEMBER 列を作成してからそのデータに対して索引を作成して、シーケンスを格納する列と結合させることができます。

1. 以下のようにメンバー番号を格納する新規列を追加します。

```
ALTER TABLE ordernumber
  ADD COLUMN curmem SMALLINT DEFAULT CURRENT MEMBER IMPLICITLY HIDDEN
```

2. 以下のようにシーケンス列（この例では seqnumber）に索引を作成（またはドロップして再作成）します。このとき、新しい列を索引に追加します。

```
CREATE INDEX seqindex
  ON ordernumber (curmem, seqnumber)
```

シーケンスが一連のタイム・スタンプ値である場合、データベース設計で同様のアプローチを取ることができます。タイム・スタンプ列の索引は PAGE SPLIT HIGH オプションを使用し、新しい CURRENT MEMBER 列も含めます。

固有値が少数である場合の索引付け

表によっては、特定の列の中で異なる値が含まれる割合が、表の行数に比べてわずかである場合があります。この場合にも、CURRENT MEMBER 列を追加することに利点があります。

この場合、データベースのパフォーマンスを改善するには、以下のようにします。

1. 以下のようにメンバー番号を格納する新規列を追加します。

```
ALTER TABLE customerinfo
  ADD COLUMN curmem SMALLINT DEFAULT CURRENT MEMBER IMPLICITLY HIDDEN
```

2. 以下のように、異なる値がわずかしか含まれない 1 つ以上の列（zipcode や country など）に対して、索引を作成（またはドロップして再作成）します。

```
CREATE INDEX lowvaridx
ON customerinfo (zipcode, country, curmem)
```

これらすべてのケースで、索引の圧縮により、新しい CURRENT MEMBER 値に対する索引のサイズが削減される可能性があります。

既存の SQL ステートメントへの影響を抑えるには、CREATE TABLE コマンドまたは ALTER TABLE コマンドで IMPLICITLY HIDDEN オプションを使用して、新しい列を隠します。この隠し列のオプションにより、メンバー情報は、明示的な SQL ステートメント参照でのみ利用できるようになります。

パーティションとクラスタリング

パーティション表での索引の動作

パーティション表の索引は、非パーティション表の索引と同様に機能します。ただし、パーティション表の索引は、索引をパーティション化するかどうかによって、異なるストレージ・モデルを使用して格納されます。

通常の非パーティション表の索引はすべて、共有索引オブジェクト内にありますが、パーティション表の非パーティション索引は、データ・パーティションが複数の表スペースにまたがっている場合であっても、単一の表スペースの独自の索引オブジェクト内に作成されます。データベース管理スペース (DMS) とシステム管理スペース (SMS) 表スペースは両方とも、表データとは別の場所にある索引の使用をサポートしています。各非パーティション索引は、LARGE 表スペースを含めて独自の表スペースに配置できます。各索引表スペースは、データ・パーティションとして DMS か SMS のどちらかの同一のストレージ・メカニズムを使用しなければなりません。LARGE 表スペース内の索引は、最大 2^{29} ページまで含めることが可能です。すべての表スペースは、同一のデータベース・パーティション・グループになければなりません。

パーティション索引は、複数の索引パーティションに索引データを分割するという索引編成スキームを使用します。分割は、表のパーティション・スキームに従って行われます。各索引パーティションは、対応するデータ・パーティション内の表の行のみを参照します。特定のデータ・パーティション用の索引パーティションはすべて、同じ索引オブジェクト内にあります。

DB2 バージョン 9.7 フィックスパック 1 以降では、パーティション表内の XML 列の XML データに対するユーザー作成の索引は、パーティション索引と非パーティション索引のどちらにもすることができます。デフォルトはパーティション索引です。システム生成の XML 領域索引は必ずパーティション化され、システム生成の列パス索引は常に非パーティションです。DB2 V9.7 では、XML データに対する索引はすべて非パーティションです。

非パーティション索引の利点には、以下のものがあります。

- 各索引に異なる表スペース特性を定義できます (例えば、スペースの使用効率をより良いものにするには、ページ・サイズを異ならせると役立つ場合があります)。
- 索引は、相互に独立して再編成できます。
- 索引のドロップ操作に関するパフォーマンスが向上します。

- 入出力競合を削減して、索引データにより効率的な同時アクセスを提供するのに役立ちます。
- 個々の索引をドロップすると、索引再編成を行わなくてもシステムがスペースをすぐに使用できます。

パーティション索引の利点には、以下のものがあります。

- データのロールインおよびロールアウトのパフォーマンスが向上します。
- 索引がパーティション化されるため、索引ページ上の競合が少なくなります。
- 各索引パーティションの索引 B ツリー構造により、以下の利点が得られる場合があります。
 - 挿入、更新、削除、およびスキャンのパフォーマンスが向上する。これは、表のすべてのデータを参照する索引に比べて、通常、索引パーティションの B ツリーの中に含まれるレベルの数が少ないためです。
 - パーティションの除去が有効である場合、スキャンのパフォーマンスおよび並行性が向上する。パーティションの除去は、パーティション索引および非パーティション索引の両方のスキャンで使用可能ですが、より効果的なのはパーティション索引のスキャンの場合です。それは、各索引パーティションには、対応するデータ・パーティション以外のキーは含まれないためです。この構成により、非パーティション索引での同様の照会に比べて、スキャンするキーと索引ページの数が少なく済む場合があります。

非パーティション索引は索引列上の順序を常に保持しますが、パーティション索引は、一部のシナリオ (例えば、パーティション列が索引列と一致しない場合、および複数のパーティションがアクセスされる場合) において、複数のパーティションにまたがる順序を失うことがあります。

オンラインの索引作成中は、表への同時読み取り/書き込みアクセスが許可されます。オンライン索引が作成された後、索引作成中に表に行われた変更が、新しい索引に適用されます。表への書き込みアクセスは、索引の作成が完了し、トランザクションがコミットするまで、ブロックされます。パーティション索引の場合、各データ・パーティションは、そのデータ・パーティションに対して (索引パーティションの作成中に) 加えられた変更を適用する期間中に限り、静止されて読み取り専用アクセスを受け入れます。

パーティション索引サポートは、ALTER TABLE...ATTACH PARTITION ステートメントを使用してデータをロールインするときに、特に役立ちます。非パーティション索引が存在している場合 (表に XML データがある場合、XML 列パスの索引は含まない)、パーティションのアタッチの後に、SET INTEGRITY ステートメントを発行します。このステートメントは、非パーティション索引の保守、範囲妥当性検査、制約チェック、およびマテリアライズ照会表 (MQT) の保守に必要です。非パーティション索引の保守は、時間がかかり、大容量のログ・スペースを必要とする可能性があります。パーティション索引を使用して、この保守のコストを回避してください。

アタッチ操作の後に、保守対象の表の非パーティション索引 (XML 列パス索引を除く) が存在する場合、SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントは SET INTEGRITY...IMMEDIATE CHECKED ステートメントのように機能

します。すべての整合性処理、非パーティション索引の保守、および表の状態遷移は、あたかも SET INTEGRITY...IMMEDIATE CHECKED ステートメントが発行されたかのように実行されます。

図 12 のダイアグラムは、1 つのパーティション表に対する 2 つの非パーティション索引を示しており、それぞれの索引は別々の表スペースにあります。

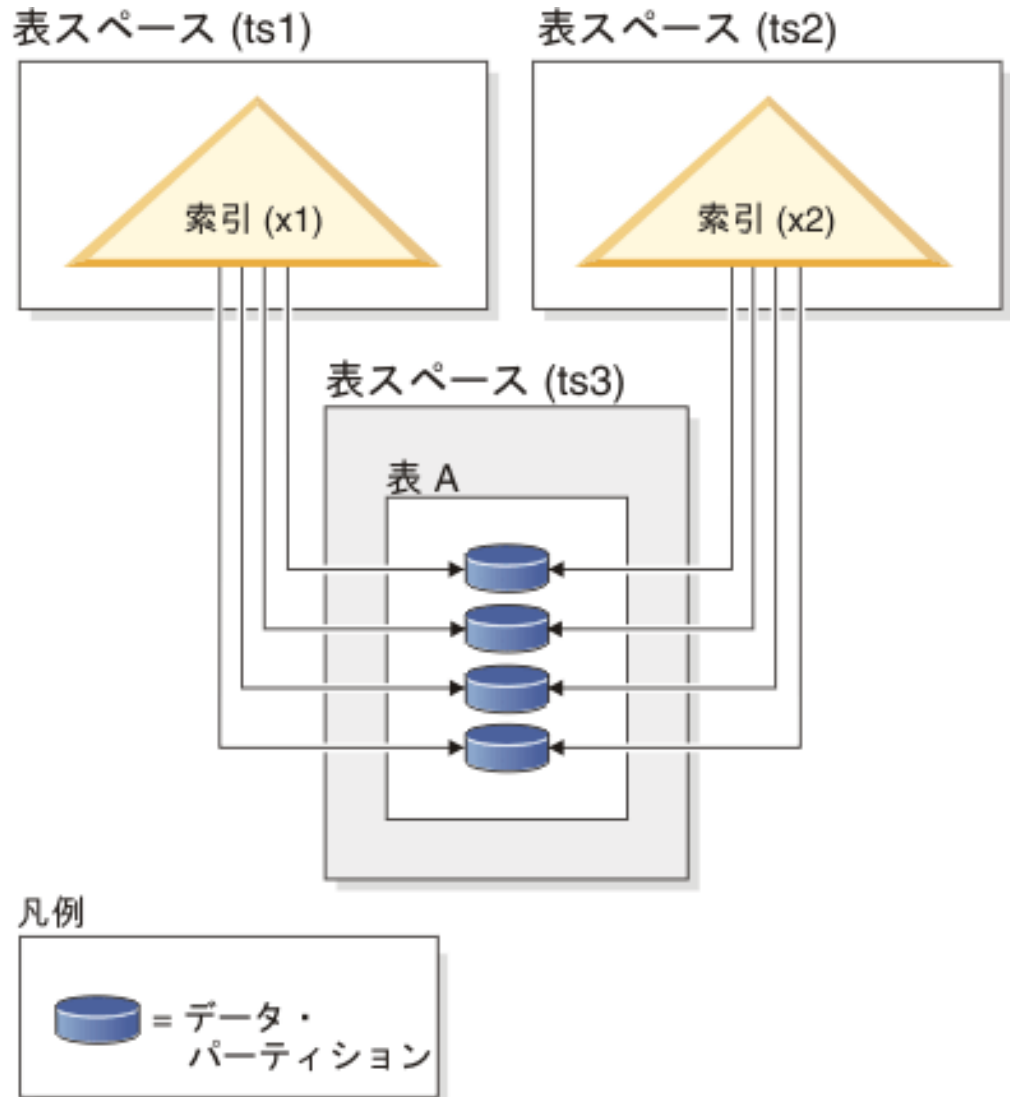
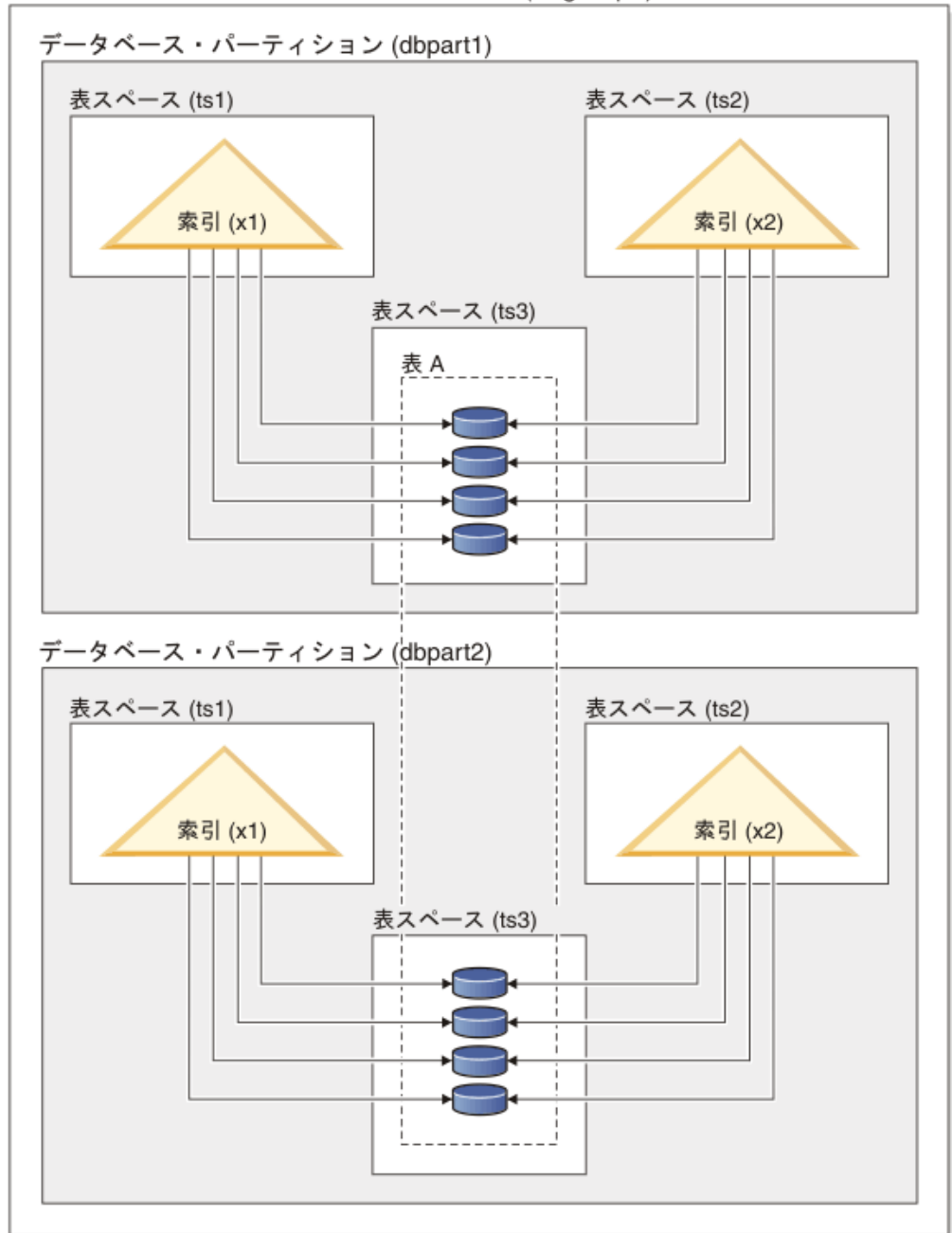


図 12. パーティション表上の非パーティション索引

92 ページの図 13 のダイアグラムは、2 つのデータベース・パーティションにわたり、なおかつ 1 つの表スペースに内在するパーティション表の、非パーティション索引を示しています。

データベース・パーティション・グループ (dbgroup1)



凡例

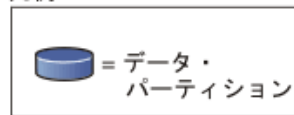


図 13. 分散とパーティション化の両方が行われている表における非パーティション索引

93 ページの図 14 のダイアグラムは、1 つのパーティション表に対してパーティション索引と非パーティション索引が混在する状態を示しています。

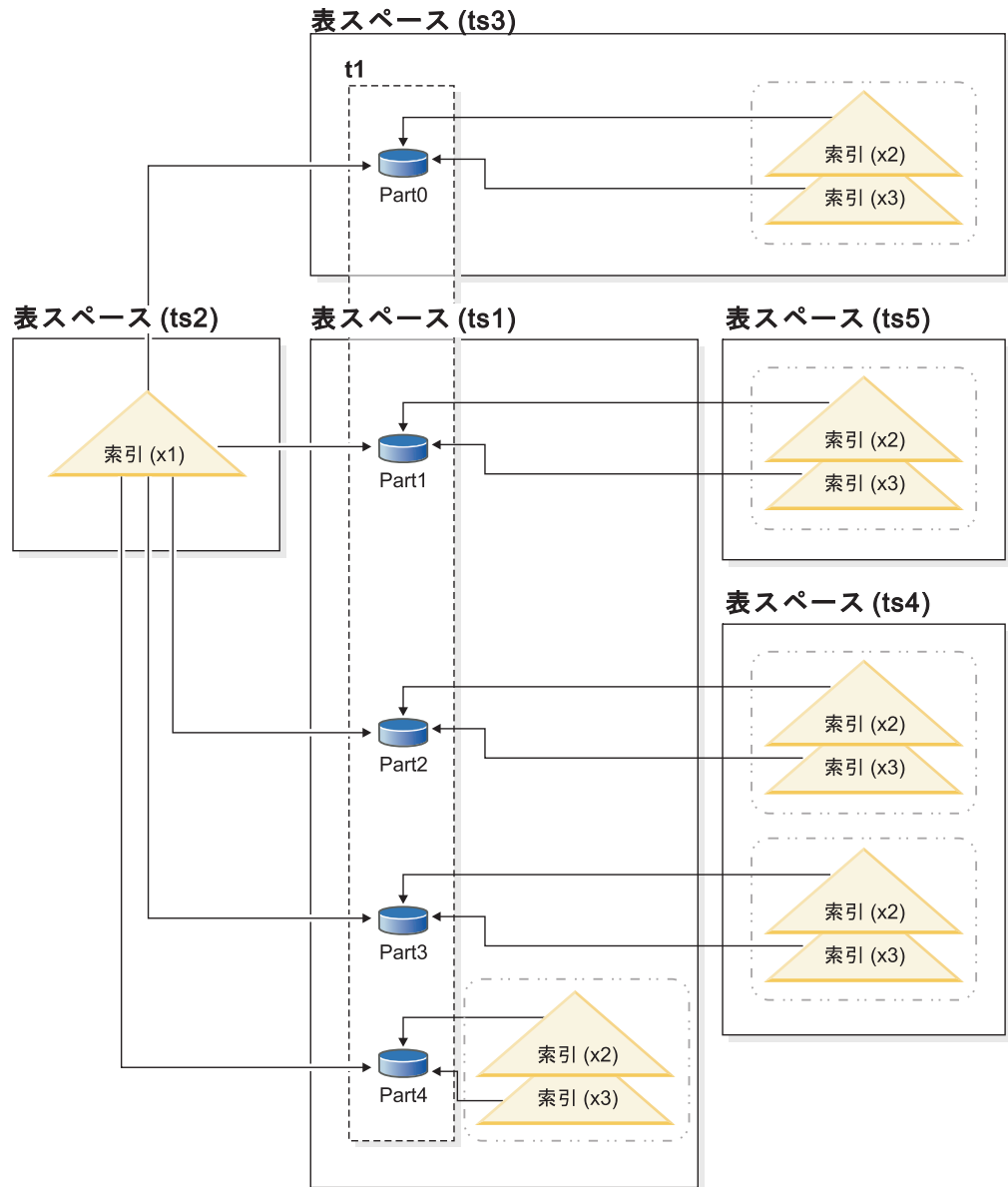


図 14. パーティション表上のパーティション索引と非パーティション索引

非パーティション索引 X1 は、すべてのデータ・パーティションの行を参照します。対照的に、パーティション索引 X2 および X3 は、関連するデータ・パーティションの行のみを参照します。表スペース TS3 は、関連するデータ・パーティションの表スペースを共有する索引パーティションも示しています。この構成は、パーティション索引のデフォルトです。

方法はそれぞれ異なりますが、非パーティション索引およびパーティション索引のデフォルトの位置をオーバーライドすることができます。非パーティション索引の場合、索引を作成するときに表スペースを指定できます。パーティション索引の場合は、表を作成するときに、索引パーティションを格納する表スペースを決めておく必要があります。

非パーティション索引

非パーティション索引の索引位置をオーバーライドするには、`CREATE INDEX` ステートメントで `IN` 節を使用して、代わりに表スペース・ロケーションを索引用に指定できます。必要に応じて、別々の索引を別々の表スペースに配置することができます。パーティション化されていない索引を配置する場所を指定しないでパーティション表を作成し、表スペースを指定しない `CREATE INDEX` ステートメントを使用して索引を作成する場合、最初のアタッチされたデータ・パーティションまたは表示可能なデータ・パーティションの表スペースに索引が作成されます。次の 3 つの考え得る各ケースをケース 1 から順番に評価して、索引が作成される場所を判別します。この評価により、一致したケースが見つかりと停止して、索引の表スペースの配置場所を判別します。

ケース 1:

索引表スペースが `CREATE INDEX...IN tblspace` ステートメントで指定される場合、この索引用に指定された表スペースを使用します。

ケース 2:

索引表スペースが `CREATE TABLE...INDEX IN tblspace` ステートメントで指定される場合、この索引用に指定された表スペースを使用します。

ケース 3:

表スペースが指定されない場合、最初のアタッチされたデータ・パーティションまたは可視のデータ・パーティションによって使用される表スペースを選択します。

パーティション索引

デフォルトでは、索引パーティションは、参照するデータ・パーティションと同じ表スペースに配置されます。このデフォルト動作をオーバーライドするには、`CREATE TABLE` ステートメントを使用して定義するデータ・パーティションごとに `INDEX IN` 節を使用する必要があります。つまり、パーティション表でのパーティション索引の使用を計画している場合、表を作成するときに、索引パーティションを保管する場所をあらかじめ決めておく必要があります。パーティション索引を作成するときに `INDEX IN` 節を使用しようとする、エラー・メッセージを受け取ります。

例 1: パーティション表 `SALES (a int, b int, c int)` を想定し、ユニーク索引 `A_IDX` を作成します。

```
create unique index a_idx on sales (a)
```

表 `SALES` がパーティション化されるため、索引 `a_idx` もパーティション索引として作成されます。

例 2: 索引 `B_IDX` を作成します。

```
create index b_idx on sales (b)
```

例 3: パーティション索引の索引パーティションのデフォルトの位置をオーバーライドするには、パーティション表を作成するときに定義するパーティションごとに `INDEX IN` 節を使用します。以下の例では、表 `Z` の索引が表スペース `TS3` に作成されます。

```
create table z (a int, b int)
  partition by range (a) (starting from (1)
    ending at (100) index in ts3)

create index c_idx on z (a) partitioned
```

パーティション表上の非パーティション索引のクラスタリング

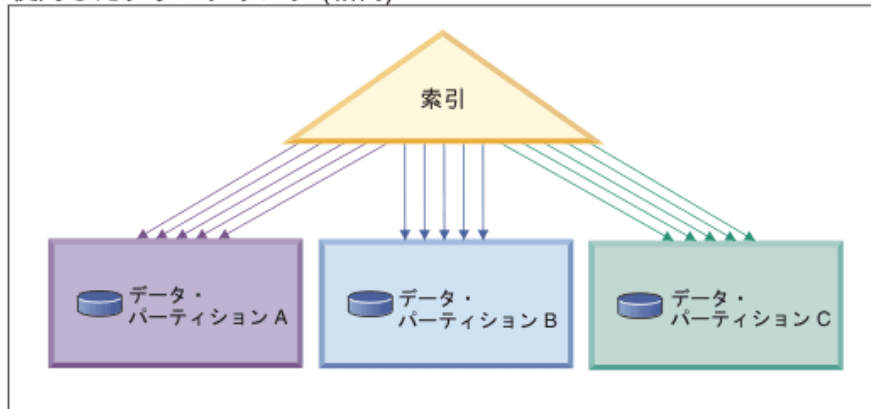
クラスタリング索引には、REGULAR 表に対するのと同じ利点がパーティション表に関するもあります。しかし、クラスタリング索引を選択する際には表パーティション・キー定義に関して注意が必要です。

任意のクラスタリング・キーを使用して、パーティション表にクラスタリング索引を作成できます。データベース・サーバーは、クラスタリング索引を使用して、各データ・パーティション内でデータをローカルにクラスタ化しようとします。クラスタ化された挿入操作の際、適切なレコード ID (RID) を検出するために索引検索が実行されます。この RID は、表内でレコードを挿入するスペースを検索する際の開始点として使用されます。効率が良く、最適化されたクラスタリングを実行するには、索引列と表パーティション・キー列間に相関がなければなりません。そのような相関を確保する 1 つの方法は、以下の例に示されているように、表パーティション・キー列を使用して索引列を先頭に付けることです。

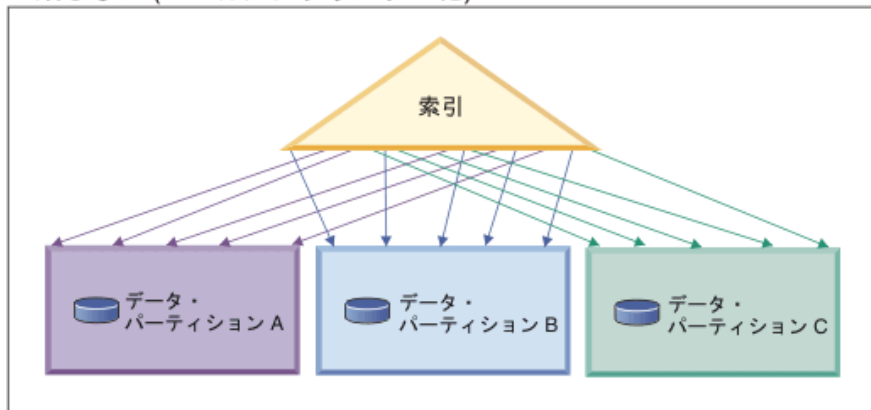
```
partition by range (month, region)
create index...(month, region, department) cluster
```

データベース・サーバーがこの相関を強制することはありませんが、適切なクラスタリングを行うために索引中のすべてのキーがパーティション ID ごとに互いにグループ化されていることが期待されています。例えば、QUARTER 上にパーティションされた表が 1 つあり、クラスタリング索引が DATE で定義されているとします。QUARTER と DATE の間には関連があり、データ・パーティションのすべてのキーが索引内で相互にグループ化されているので、効率の良い最適なデータのクラスタリングを行うことが可能です。96 ページの図 15 は、クラスタリングが表パーティション・キーと相互に関連がある場合にのみ、最適なスキャン・パフォーマンスが得られることを示しています。

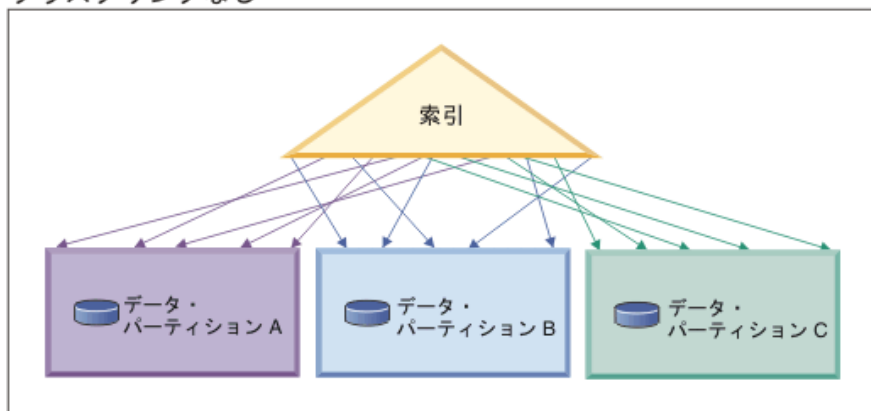
接頭部としてパーティション・キーを使用したクラスタリング (関連)



クラスタリングがパーティション・キーに一致しない (ローカルにクラスター化)



クラスタリングなし



凡例

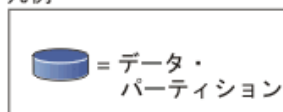


図 15. パーティション表でのクラスター化された索引の見込まれる効果。

クラスタリングの利点には、以下のものがあります。

- 各データ・パーティション内で、行はキーの順序になります。

- クラスタリング索引は、キーの順序で表内をトラバースするため、スキャンのパフォーマンスが向上します。なぜなら、スキャナーは最初のページの最初の行を取り出し、その後その同じページの各行を取り出してから次のページに移動するからです。つまり、どの時点でも表の 1 ページのみがバッファ・プール内になければならないという意味です。対照的に、表がクラスタ化されていない場合、異なるページから各行が取り出される確率が高くなります。バッファ・プールが表全体を保持できる場合を除いて、ほとんどのページが何度も取り出される可能性が高くなり、スキャンの速度がとて遅くなります。

クラスタリング・キーが表パーティション・キーと関連していないものの、データがローカルにクラスタ化される場合、バッファ・プールに各データ・パーティションの 1 ページを保持できるスペースが十分にあれば、クラスタ化された索引の利点を十分に生かすことが依然として可能です。これは、特定のデータ・パーティションから取り出される各行が、同じパーティションから以前に取り出された行の近くにあるためです (96 ページの図 15 の 2 番目の例を参照)。

フェデレーテッド・データベース

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・データベース・システムは、DB2 データ・サーバー (フェデレーテッド・データベース) と 1 つ以上のデータ・ソースで構成されています。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。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および制御するサーバー・オプションを指定することもできます。

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・マネージャーの **federated** 構成パラメーターを **YES** に設定する必要があります。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。

`CREATE SERVER` ステートメントに指定するサーバー・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。例えば、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できます。`cpu_ratio` オプションは、データ・ソースおよびフェデレーテッド・サーバーにおけるプロセッサの相対速度を指定し、`io_ratio` オプションは、データ・ソースおよびフェデレーテッド・サーバーにおけるデータ入出力装置の相対速度を指定します。

サーバー・オプション値はシステム・カタログ (`SYSCAT.SERVEROPTIONS`) に書き込まれ、オプティマイザーは、データ・ソースのアクセス・プランを作成するときこの情報を使用します。統計情報が変更された場合 (例えばデータ・ソース・プロセッサ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、カタログを新しい値で更新してください。

リソースの利用

メモリーの割り振り

メモリーの割り振りと割り振り解除はさまざまな時点で発生します。メモリーは、指定のイベントが発生する際（アプリケーションの接続時など）に特定のメモリー領域に割り振られる場合もあり、構成の変更に対する対応として再割り振りが行われる場合もあります。

図 16 は、データベース・マネージャーによって各種用途のために割り振られるさまざまなメモリー領域、およびユーザーがこれらのメモリー領域のサイズを制御することを可能にする構成パラメーターを示しています。パーティション・データベース環境では、各データベース・パーティションには独自のデータベース・マネージャー共有メモリー・セットがあることを覚えておいてください。

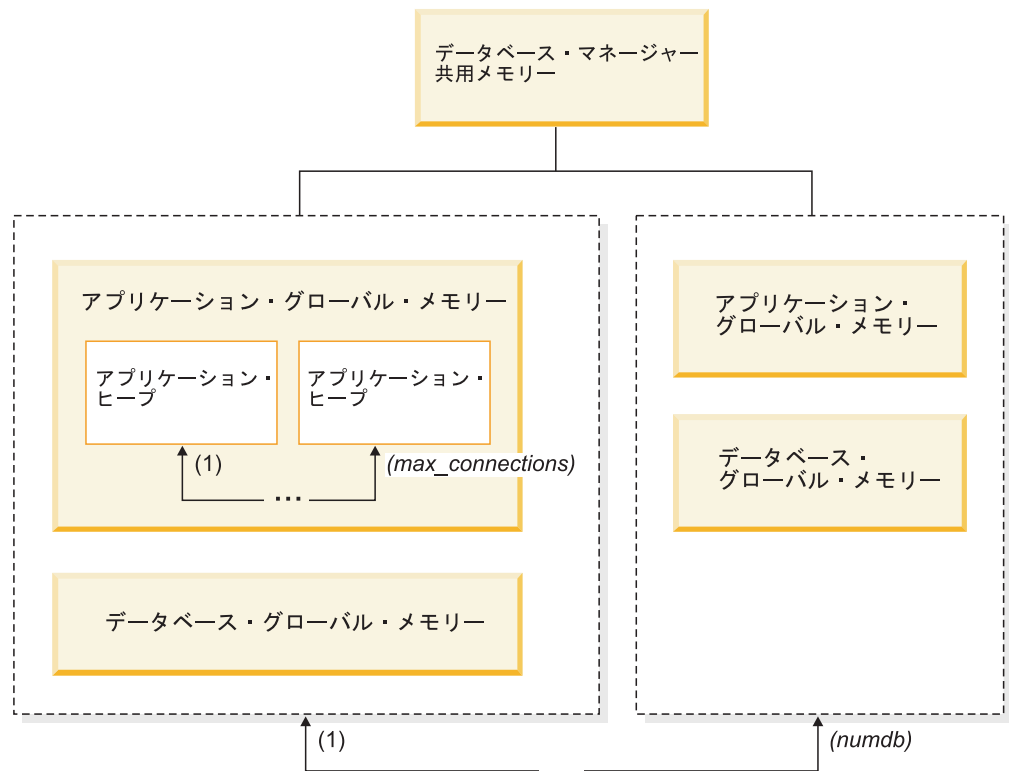


図 16. データベース・マネージャーによって割り振られるメモリーのタイプ

メモリーは、以下のいずれかのイベントが生じるとデータベース・マネージャーによって割り振られます。

データベース・マネージャーが開始したとき (db2start)

データベース・マネージャーの共有メモリー（インスタンス共有メモリーとも呼ばれる）は、データベース・マネージャーが停止する (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。DB2 は、データベース・マネージャーの共有メモリーのサイズを自動的に制御します。

データベースが初めて活動化または接続される時

データベースに接続するすべてのアプリケーションでデータベース・グローバル・メモリーが使用されます。データベース・グローバル・メモリーのサイズは、**database_memory** データベース構成パラメーターで指定されます。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る初期メモリー量を計算し、データベースの必要に基づいて実行時にデータベースのメモリー・サイズを自動的に構成できます。

以下のメモリー領域は動的に調整できます。

- バッファ・プール (ALTER BUFFERPOOL ステートメントを使用)
- データベース・ヒープ (ログ・バッファを含む)
- ユーティリティ・ヒープ
- パッケージ・キャッシュ
- カタログ・キャッシュ
- ロック・リスト

sortheap、**sheapthres_shr**、および **sheapthres** 構成パラメーターも、動的に更新可能です。唯一の制約事項は、**sheapthres** は、0 から 0 より大きい値への動的な変更ができず、その逆の変更もできません。

共有ソート操作がデフォルトで実行され、任意の一時点にソート・メモリー・コンシューマーが使用できるデータベース共有メモリーの量は、**sheapthres_shr** データベース構成パラメーターの値で決まります。専用ソート操作は、パーティション内並列処理、データベース・パーティション、および接続コンセントレーターがすべて使用不可になっていて、データベース・マネージャーの **sheapthres** 構成パラメーターが非ゼロの値に設定されている場合にのみ、実行されます。

アプリケーションがデータベースに接続するとき

それぞれのアプリケーションには、固有のアプリケーション・ヒープがあります。これは、アプリケーション・グローバル・メモリーの一部です。

applheapsz データベース構成パラメーターを使用してアプリケーションが割り振ることができるメモリーの量を制限することもできますし、

appl_memory データベース構成パラメーターを使用してアプリケーション・メモリー消費量全体を制限することもできます。

エージェントが作成される時

パーティション・データベース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、そのエージェントのためにエージェント専用メモリーが割り振られます。エージェント専用メモリーには、この特定のエージェントによってのみ使用されるメモリーが含まれます。専用ソート操作が使用可能になっている場合、専用ソート・ヒープはエージェント専用メモリーの中から割り振られます。

それぞれのタイプのメモリー領域に割り振られるメモリーの量は、以下の構成パラメーターによって制限されます。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 このデータベース・マネージャー構成パラメーターは、さまざまなアプリケ

ーションが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、割り振ることができるメモリーの量も大きくなります。

maxappls

このデータベース構成パラメーターは、特定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。このパラメーター値は、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・メモリーの両方に割り振ることができるメモリーの量に影響します。

max_connections

このデータベース・マネージャー構成パラメーターは、データ・サーバーに同時にアクセスできるデータベース接続またはインスタンスのアタッチの数を制限します。

max_coordagents

このデータベース・マネージャー構成パラメーターは、インスタンス内で(さらにパーティション・データベース環境においてはデータベース・パーティションごとに) 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・コーディネーター・エージェントの数を制限します。 **maxappls** と **max_connections** とを合わせて使用することにより、こ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せん。

db2mtrk コマンドで起動されるメモリー・トラッカーを使用して、インスタンス内の現行のメモリーの割り振りを表示することができます。また、**ADMIN_GET_MEM_USAGE** 表関数を使用して、インスタンス全体または単一のデータベース・パーティションだけの合計メモリー消費量を判別することもできます。**MON_GET_MEMORY_SET** 表関数および **MON_GET_MEMORY_POOL** 表関数を使用すると、現行メモリー使用量を、インスタンス、データベース、またはアプリケーションの各レベルで調べることができます。

UNIX および Linux オペレーティング・システムでは、**ipcs** コマンドを使用してすべての共有メモリー・セグメントをリスト表示できますが、消費されたリソース量が正確に反映されているわけではありません。**db2mtrk** コマンドを **ipcs** の代わりに使用できます。

メモリー・セットの概要

DB2 のメモリー・マネージャーは、オペレーティング・システムからのメモリー割り振りをメモリー・セットにグループ化します。

特定のメモリー・セット内のメモリーは、共通属性を共有します。その属性には、メモリーを使用する一般的な目的、予想される揮発性、メモリーの増加時の制約などが含まれます。例えば、バッファ・プールはデータベース・メモリー・セットから割り振られ、データベースがアクティブである限り割り振られます。ステートメント・ヒープは、アプリケーションからの特定の SQL 準備要求のためにアプリケーション・メモリー・セットから割り振られ、ステートメント・コンパイル操作の間のみ持続します。

各メモリー・セット内で、特定のメモリー域は、一般に特定のメモリー・セットのタイプに関連した目的のために割り当てられます。例えば、あるタイプのデータベース・レベルの処理では、データベース・メモリー・セットからのメモリーが使用されます。高速コミュニケーション・マネージャーで必要なメモリーは FCM メモリー・セットから割り振られます。

表 1 は、さまざまなタイプのメモリー・セットをリストしています。

表 1. メモリー・セット

メモリー・セットのタイプ*	説明	このセットから割り振られるメモリーの有効範囲
DBMS	データベース・メモリー・マネージャー・セット。このセット内の大部分のメモリーは、データベースに特定されない通信サービスなどの、基本インフラストラクチャーの目的で使用されます。このメモリー・セットからのユーザー構成可能メモリーにはモニター・ヒープ・サイズ (mon_heap_sz) と監査バッファ・サイズ (audit_buf_sz) が含まれていますが、このセットには構成は必要ありません。	インスタンス
FMP	fenced モード・プロセスのメモリー・セット。このセットから割り振られるメモリーは、エージェントと fenced モード・プロセスの間の通信に使用されます。このセットからのメモリー割り振りは、 DB2_FMP_COMM_HEAPSZ レジストリー変数と as1heapsz 構成パラメーターを使用して構成できます。	インスタンス
PRIVATE	このセットから割り振られるメモリーは、基本インフラストラクチャーや診断サポートなどの一般的な目的で使用されます。専用メモリー・セットには構成は必要ありませんが、専用ソート・モデルを使用するシステムは例外で、 sheapthres 構成パラメーターを 0 以外の値に設定します。	インスタンス

表1. メモリー・セット (続き)

メモリー・セットのタイプ*	説明	このセットから割り振られるメモリーの有効範囲
DATABASE	データベース・メモリー・セット。このセットから割り振られるメモリーは、一般に単一のデータベース固有の処理に使用され、アプリケーション固有の処理には使用されません。このセットから割り振られるメモリーの例としては、バッファー・プール、データベース・ヒープ、ロック・リスト、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、および共有ソート・ヒープが含まれます。このセットは database_memory データベース構成パラメーターを使用して構成できます。セルフチューニング・メモリー・マネージャー (STMM) を使用してこのメモリー域をチューニングすることもできます。	データベース
APPLICATION	アプリケーション・メモリー・セット。このセットから割り振られるメモリーは、一般にアプリケーション固有の処理に使用されます。このセットから割り振られるメモリーには、アプリケーション、統計ヒープ、ステートメント・ヒープ、および構成不要な共有作業域が含まれます。このセットは appl_memory データベース構成パラメーターを使用して構成できます。	データベース
FCM	高速コミュニケーション・マネージャーのメモリー・セット。このセットから割り振られるメモリーは、高速コミュニケーション・マネージャーによって排他的に使用されます。このセットからのメモリーは、 fcm_num_buffers および fcm_num_channels データベース・マネージャー構成パラメーターを使用して構成できます。	ホスト
* 1 列目に示されている名前は、 memory_set_type モニター・エレメントに応じて戻される名前です。		

データベース・マネージャー共有メモリー

データベース・マネージャーの共有メモリーは、複数の異なるメモリー領域に編成されています。構成パラメーターを使用すると、これらの領域のサイズを制御できます。

103 ページの図 17 は、データベース・マネージャーの共有メモリーがどのように割り振られるかを示します。

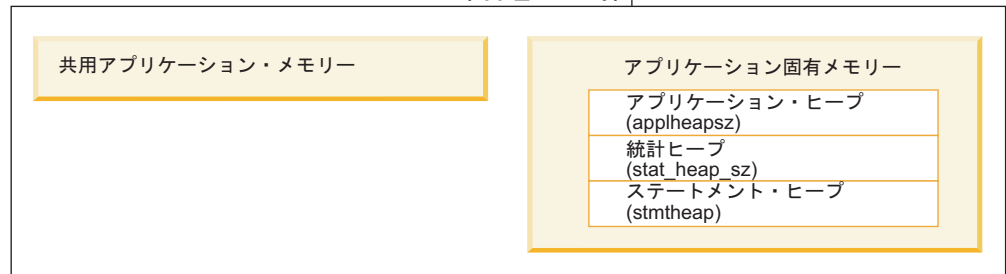
データベース・マネージャーの共用メモリ(FCM を含む)



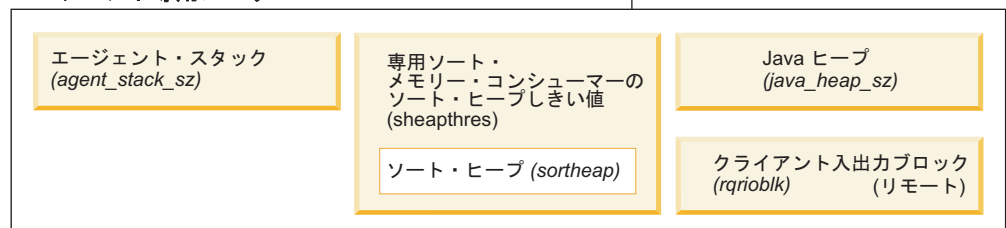
データベース・グローバル・メモリ (database_memory)



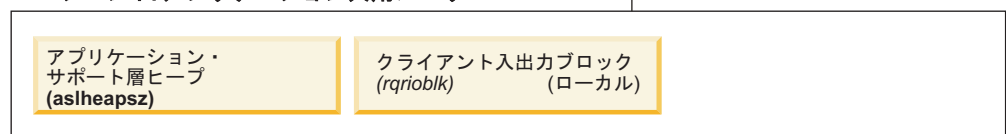
アプリケーション・グローバル・メモリ (appl_memory)



エージェント専用メモリ



エージェント/アプリケーション共用メモリ



注: この図のボックスの大きさがメモリの相対的なサイズを表すわけではありません。

図 17. データベース・マネージャーでのメモリの使用方法

データベース・マネージャー共有メモリ

データベース・マネージャー共有メモリは、以下の構成パラメーターの影響を受けます。

- **audit_buf_sz** 構成パラメーターは、データベース監査アクティビティに使用されるバッファのサイズを決定します。

- **mon_heap_sz** 構成パラメーターは、データベース・システム・モニター・データに使用されるメモリー領域のサイズを決定します。
- 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。FCM メモリー所要量は FCM バッファ・プールから割り振られます。

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、バッファ・プールのサイズと、以下のデータベース構成パラメーターの影響を受けます。

- **catalogcache_sz**
- **database_memory**
- **dbheap**
- **locklist**
- **pckcachesz**
- **sheapthres_shr**
- **util_heap_sz**

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、**appl_memory** 構成パラメーターによって制御できます。以下のデータベース構成パラメーターを使用して、いずれか 1 つのアプリケーションが消費できるメモリーの量を制限できます。

- **applheapsz**
- **stat_heap_sz**
- **stmheap**

エージェント専用メモリー

各エージェントには、それぞれの専用メモリー領域が必要です。データ・サーバーは、構成済みメモリー・リソースに合わせて必要な数のエージェントを作成します。コーディネーター・エージェントの最大数は、データベース・マネージャーの **max_coordagents** 構成パラメーターを使用して制御できます。各エージェントの専用メモリー領域の最大サイズは、次の構成パラメーターの値によって決まります。

- **agent_stack_sz**
- **sheapthres** および **sortheap**

エージェント/アプリケーション共有メモリー

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有メモリー・セグメントの合計数は、次の 2 つの値のうちの小さい方の値によって制限されます。

- アクティブなすべてのデータベースの **maxappls** データベース構成パラメーターの合計値
- **max_coordagents** データベース構成パラメーターの値

注: エンジン集中機能が有効にされている構成 (**max_connections** > **max_coordagents**) では、アプリケーションのメモリー消費量は **max_connections** によって制限されます。

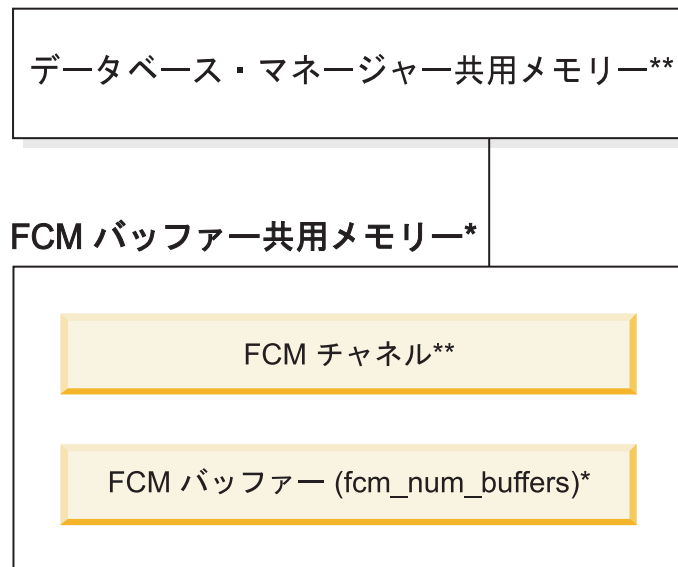
エージェント/アプリケーション共有メモリーは以下のデータベース構成パラメーターの影響も受けます。

- `aslheapsz`
- `rqrioblk`

FCM バッファ・プールとメモリーの要件

パーティション・データベース・システムでは、高速コミュニケーション・マネージャー (FCM) バッファ共有メモリーはデータベース・マネージャーの共有メモリーの中から割り振られます。

図 18 にそのことが示されています。



凡例

- * すべての論理パーティションによって共用されるもの
- ** 論理パーティションごとのもの

図 18. 複数の論理パーティションが使用されている場合の FCM バッファ・プール

各データベース・パーティションの FCM バッファの数は、データベース・マネージャーの `fcm_num_buffers` 構成パラメーターにより制御されます。デフォルトでは、このパラメーターは `automatic` に設定されます。このパラメーターを手動で調整するには、`buff_free` および `buff_free_bottom` システム・モニター・エレメントのデータを使用します。

各データベース・パーティションの FCM チャンネルの数は、データベース・マネージャーの `fcm_num_channels` 構成パラメーターにより制御されます。デフォルトでは、このパラメーターは `automatic` に設定されます。このパラメーターを手動で調整するには、`ch_free` および `ch_free_bottom` システム・モニター・エレメントのデータを使用します。

DB2 データベース・マネージャーは、必要に応じてより多くの FCM バッファーおよびチャンネルを割り振ることにより、FCM メモリー・リソースを自動的に管理できます。これにより、パフォーマンスが改善され、「FCM リソース不足」ランタイム・エラーを防ぎます。Linux オペレーティング・システムでは、データベース・マネージャーは、より大容量 (最大デフォルト量の 4 GB まで) のシステム・メモリーを FCM バッファーおよびチャンネルに事前割り振りすることができます。追加の FCM バッファーまたはチャンネルが必要なときにのみ、メモリー・スペースは影響を受けます。この動作を使用可能にするには、**DB2_FCM_SETTINGS** レジストリー変数の **FCM_MAXIMIZE_SET_SIZE** オプションを YES (または TRUE) に設定します。YES はデフォルト値です。

メモリー使用量に影響を与えるチューニング・パラメーターのガイドライン

メモリーを手動でチューニングするとき (つまり、セルフチューニング・メモリー・マネージャーを使用しないとき)、ベンチマーク・テストを行うと、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が得られます。

ベンチマーク・テストでは、代表的な SQL ステートメントおよび最悪ケースの SQL ステートメントがサーバーに対して実行され、パフォーマンスがこれ以上向上しないポイントが見つかるまでメモリー・パラメーターの値が変更されます。これは、メモリーの割り振りを増やしてもアプリケーションにとってパフォーマンス上の利益がこれ以上得られないポイントです。

パラメーターによっては、メモリー割り振りの上限が、既存のハードウェアおよびオペレーティング・システムの有効範囲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。メモリー・パラメーターの値は、正当な理由があるのではない限り、最高値に設定しないようにすると良いでしょう。これは、使用可能メモリーが豊富にあるシステムにも当てはまります。それは、データベース・マネージャーがシステム上の使用可能メモリーすべてをすぐに使い果たすことがないようにするためです。また、大量のメモリーを管理すると、オーバーヘッドが増大します。

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にコミットされ、パラメーター設定によって特定のメモリー・ヒープの最大サイズが決まります。ただし、バッファー・プールおよび以下の構成パラメーターの場合、指定されたすべてのメモリーが割り振られます。

- **aslheapsz**
- **fcm_num_buffers**
- **fcm_num_channels**
- **locklist**

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。

セルフチューニング・メモリーの概要

セルフチューニング・メモリーは、メモリー構成パラメーターの値の設定とバッファー・プールのサイズ変更を自動的に実行することにより、メモリー構成のタスクを単純化します。メモリー・チューナーを有効にすると、使用可能メモリー・リソ

ースがバッファー・プール、ロック・メモリー、パッケージ・キャッシュ、およびソート・メモリーなどのメモリー・コンシューマーの間で動的に分配されます。

セルフチューニング・メモリーは、**self_tuning_mem** データベース構成パラメーターによって使用可能にします。

以下のメモリー関連のデータベース構成パラメーターを自動的に調整できます。

- **database_memory** - データベース共有メモリー・サイズ
- **locklist** - ロック・リスト用最大ストレージ
- **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
- **pckcachesz** - パッケージ・キャッシュ・サイズ
- **sheapthres_shr** - 共有ソートのソート・ヒープのしきい値
- **sortheap** - ソート・ヒープ・サイズ

セルフチューニング・メモリー

DB2 バージョン 9 からは、メモリー・チューニング・フィーチャーにより、いくつかのメモリー構成パラメーターの値が自動的に設定されることによって、メモリー構成のタスクを単純化しています。メモリー・チューナーを有効にすると、使用可能メモリー・リソースがバッファー・プール、ロック・メモリー、パッケージ・キャッシュ、およびソート・メモリーなどのメモリー・コンシューマーの間で動的に分配されます。

チューナーは、**database_memory** 構成パラメーターによって定義されたメモリー制限内で作動します。このパラメーターの値も自動的に調整することができます。セルフチューニングが有効な場合 (**database_memory** の値が **AUTOMATIC** に設定されている場合)、チューナーはデータベースの全体的なメモリー要件を判別し、現在のデータベース要件に応じてデータベース共有メモリーに割り振られるメモリーの量を増減します。例えば、現行データベース要件が高く、システムに十分な空きメモリーがある場合、さらに多くのメモリーがデータベース共有メモリーに割り振られます。データベース・メモリー所要量が下げられるか、またはシステムの空きメモリー量が過度に減ると、データベース共有メモリーの一部が解放されます。

database_memory 構成パラメーターが **AUTOMATIC** に設定されていない場合、データベースはこのパラメーターに指定されているメモリー量を使用し、必要に応じてメモリー・コンシューマー間でそれを分配します。メモリーの量は、**database_memory** を数値に設定するか、またはそれを **COMPUTED** に設定するという 2 とおりの方法で指定できます。後者のケースでは、メモリーの合計量は、データベース起動時のデータベース・メモリー・ヒープの初期値の合計に基づきます。

また以下のようにして、メモリー・コンシューマーはセルフチューニングを有効にできます。

- バッファー・プールの場合は、**ALTER BUFFERPOOL** または **CREATE BUFFERPOOL** ステートメントを使用します (**AUTOMATIC** キーワードを指定します)。
- ロッキング・メモリーの場合は、**locklist** または **maxlocks** データベース構成パラメーターを使用します (値 **AUTOMATIC** を指定します)。
- パッケージ・キャッシュの場合は、**pckcachesz** データベース構成パラメーターを使用します (値 **AUTOMATIC** を指定します)。

- ソート・メモリーの場合、**sheapthres_shr** または **sorheap** データベース構成パラメーターを使用します (値 **AUTOMATIC** を指定します)。

セルフチューニング操作による変更内容は、**stmmlog** サブディレクトリーにある、メモリー・チューニング・ログ・ファイル内に記録されます。これらのログ・ファイルには、特定のチューニング・インターバルにおける各メモリー・コンシューマーからのリソース要求の要約が含まれていて、ログ項目内のタイム・スタンプによって判別されます。

使用できるメモリーがわずかしかない場合、セルフチューニングによって得られるパフォーマンス上の利点には限度があります。チューニングの決定はデータベース・ワークロードに基づいているので、メモリー要件が急激に変化するワークロードではセルフチューニング・メモリー・マネージャー (**STMM**) の実効性が限定されるためです。ご使用のワークロードのメモリー特性が絶えず変化する場合には、**STMM** によるチューニングの頻度が低く、変化するターゲット状態の影響を受けます。このシナリオでは、**STMM** は完全な収束には至りませんが、その代わりに現行のワークロードに対してチューニングされたメモリー構成を保持しようとしています。

セルフチューニング・メモリーを有効にする

セルフチューニング・メモリーは、メモリー構成パラメーターの値の設定とバッファ・プールのサイズ変更を自動的に実行することにより、メモリー構成のタスクを単純化します。

このタスクについて

メモリー・チューナーを有効にすると、使用可能メモリー・リソースが、バッファ・プール、ロック・メモリー、パッケージ・キャッシュ、およびソート・メモリーといった複数のメモリー・コンシューマーの間で動的に分配されます。

手順

1. データベースのセルフチューニング・メモリーを有効にするには、**UPDATE DATABASE CONFIGURATION** コマンドまたは **db2CfgSet** API を使用して、**self_tuning_mem** データベース構成パラメーターを **ON** に設定します。
2. メモリー構成パラメーターによって制御されるメモリー領域のセルフチューニングを有効にするには、**UPDATE DATABASE CONFIGURATION** コマンドまたは **db2CfgSet** API を使用して、該当する構成パラメーターを **AUTOMATIC** に設定します。
3. バッファ・プールのセルフチューニングを有効にするには、**CREATE BUFFERPOOL** ステートメントまたは **ALTER BUFFERPOOL** ステートメントを使用して、バッファ・プール・サイズを **AUTOMATIC** に設定します。パーティション・データベース環境の場合、**SYSCAT.BUFFERPOOLDBPARTITIONS** には、そのバッファ・プールのいかなる項目も存在するべきではありません。

タスクの結果

注:

1. セルフチューニング対象のメモリーは、種々のメモリー・コンシューマー間で分散されます。したがって、どの時点においても、少なくとも 2 つのメモリー領域 (例: ロック・メモリーおよびデータベース共有メモリー) が、同時にセルフ

チューニングのために使用可能でなければなりません。以下の条件のいずれかが成立する場合に、メモリー・チューナーは、システム上のメモリーをアクティブにチューニングします (`self_tuning_mem` データベース構成パラメーターの値は ON)。

- 1 つの構成パラメーターまたはバッファー・プール・サイズが `AUTOMATIC` に設定されており、`database_memory` データベース構成パラメーターが数値または `AUTOMATIC` のいずれかに設定されている。
 - `locklist`、`sheapthres_shr`、`pckcachesz`、またはバッファー・プール・サイズのいずれか 2 つが、`AUTOMATIC` に設定されている。
 - `sortheap` データベース構成パラメーターが `AUTOMATIC` に設定されている。
2. `locklist` データベース構成パラメーターの値は、`maxlocks` データベース構成パラメーターと共にチューニングされます。`locklist` パラメーターのセルフチューニングを無効にすると、`maxlocks` パラメーターのセルフチューニングも自動的に無効になります。また、`locklist` パラメーターのセルフチューニングを有効にすると、`maxlocks` パラメーターのセルフチューニングも自動的に有効になります。
 3. `sortheap` または `sheapthres_shr` データベース構成パラメーターの自動チューニングが可能となるのは、データベース・マネージャー構成パラメーター `sheapthres` が 0 に設定されている場合のみです。
 4. `sortheap` の値は、`sheapthres_shr` と共にチューニングされます。`sortheap` パラメーターのセルフチューニングを無効にすると、`sheapthres_shr` パラメーターのセルフチューニングも自動的に無効になります。また、`sheapthres_shr` パラメーターのセルフチューニングを有効にすると、`sortheap` パラメーターのセルフチューニングも自動的に有効になります。
 5. セルフチューニング・メモリーは、高可用性災害時リカバリー (HADR) 1 次サーバー上でのみ実行されます。HADR システムでセルフチューニング・メモリーを活動化すると、構成が正しく設定されていれば、2 次サーバーでは実行されず 1 次サーバーでのみ実行されます。HADR データベースの役割を切り替えると、セルフチューニング・メモリー操作も新しい 1 次サーバーで実行されるように切り替わります。1 次データベースが開始した後、またはテークオーバーによってスタンバイ・データベースが 1 次データベースに変換された後、セルフチューニング・メモリー・マネージャー (STMM) のエンジン・ディスパッチ可能単位 (EDU) は、最初のクライアントが接続されるまで開始しない場合があります。

セルフチューニング・メモリーを無効にする

データベース全体、または 1 つ以上の構成パラメーターまたはバッファー・プールにおいて、セルフチューニング・メモリーを無効にすることができます。

このタスクについて

データベース全体のセルフチューニング・メモリーを無効にした場合、`AUTOMATIC` に設定されているメモリー構成パラメーターおよびバッファー・プールでは、自動チューニングが引き続き有効です。しかし、メモリー領域はその現行サイズのままとなります。

手順

1. データベースのセルフチューニング・メモリーを無効にするには、**UPDATE DATABASE CONFIGURATION** コマンドまたは `db2CfgSet` API を使用して、**self_tuning_mem** データベース構成パラメーターを **OFF** に設定します。
2. メモリー構成パラメーターによって制御されるメモリー領域のセルフチューニングを無効にするには、**UPDATE DATABASE CONFIGURATION** コマンドまたは `db2CfgSet` API を使用して、該当する構成パラメーターを **MANUAL** に設定するか、パラメーターの数値を指定します。
3. バッファー・プールのセルフチューニングを無効にするには、**ALTER BUFFERPOOL** ステートメントを使用して、バッファー・プール・サイズを特定の値に設定します。

タスクの結果

注:

- メモリー構成パラメーターのセルフチューニングの中には、関連する別のメモリー構成パラメーターも共に有効でなければ、有効にできないものもあります。このため、例えば、**locklist** または **sortheap** データベース構成パラメーターのセルフチューニング・メモリーを無効にすると、**maxlocks** または **sheapthres_shr** データベース構成パラメーターのセルフチューニング・メモリーも、それぞれ無効となります。

セルフチューニングが有効になっているメモリー・コンシューマーの判別

構成パラメーターによって制御されるセルフチューニング・メモリー設定、またはバッファー・プールに適用されるセルフチューニング・メモリー設定を表示できます。

このタスクについて

メモリー・チューナーの反応は、メモリー・コンシューマーのサイズ変更に必要な時間によって制限されることに注意する必要があります。例えば、バッファー・プール・サイズを縮小するプロセスには長い時間がかかることがあるため、バッファー・プール・メモリーをソート・メモリーと交換することによって得られるパフォーマンス上のメリットを、すぐには実現できない場合もあります。

手順

- 構成パラメーターの設定を表示するには、以下の方法のいずれかを使用します。
 - **SHOW DETAIL** パラメーターを指定して **GET DATABASE CONFIGURATION** コマンドを使用します。

セルフチューニングを使用可能にできるメモリー・コンシューマーは、出力で次のようにグループ化されます。

Description	Parameter	Current Value	Delayed Value
Self tuning memory	(SELF_TUNING_MEM)	= ON (Active)	ON
Size of database shared memory (4KB)	(DATABASE_MEMORY)	= AUTOMATIC(37200)	AUTOMATIC(37200)
Max storage for lock list (4KB)	(LOCKLIST)	= AUTOMATIC(7456)	AUTOMATIC(7456)
Percent. of lock lists per application	(MAXLOCKS)	= AUTOMATIC(98)	AUTOMATIC(98)
Package cache size (4KB)	(PCKCACHESZ)	= AUTOMATIC(5600)	AUTOMATIC(5600)
Sort heap thres for shared sorts (4KB)	(SHEAPTHRES_SHR)	= AUTOMATIC(5000)	AUTOMATIC(5000)
Sort list heap (4KB)	(SORTHEAP)	= AUTOMATIC(256)	AUTOMATIC(256)

- db2CfgGet API を使用します。

次の値が戻されます。

SQLF_OFF	0
SQLF_ON_ACTIVE	2
SQLF_ON_INACTIVE	3

SQLF_ON_ACTIVE は、セルフチューニングが使用可能かつアクティブであることを示します。一方、SQLF_ON_INACTIVE は、セルフチューニングが使用可能であるものの、現在は非アクティブであることを示します。

- バッファ・プールのセルフチューニング設定を表示するには、以下の方法のいずれかを使用します。

- セルフチューニングが使用可能なバッファ・プールのリストをコマンド行から取得するには、以下の照会を使用します。

```
SELECT BPNAME, NPAGES FROM SYSCAT.BUFFERPOOLS
```

バッファ・プールのセルフチューニングが使用可能になると、その特定のバッファ・プールについて、SYSCAT.BUFFERPOOLS ビューの NPAGES フィールドが -2 に設定されます。セルフチューニングが使用不可になると、NPAGES フィールドはバッファ・プールの現行サイズに設定されます。

- セルフチューニングが使用可能なバッファ・プールの現行サイズを判別するには、次のように **GET SNAPSHOT** コマンドを使用し、バッファ・プールの現行サイズ (**bp_cur_buffsz** モニター・エレメントの値) を調べます。

```
GET SNAPSHOT FOR BUFFERPOOLS ON database-alias
```

特定のデータベース・パーティション上のバッファ・プールのサイズを指定する ALTER BUFFERPOOL ステートメントは、そのバッファ・プールに関する例外項目を SYSCAT.BUFFERPOOLDBPARTITIONS カタログ・ビューに作成します (または既存の項目を更新します)。バッファ・プールの例外項目が存在する場合に、デフォルトのバッファ・プール・サイズが AUTOMATIC に設定されていると、そのバッファ・プールはセルフチューニング操作に関与しません。

パーティション・データベース環境でのセルフチューニング・メモリー

パーティション・データベース環境でセルフチューニング・メモリー・フィーチャーを使用する場合、このフィーチャーがシステムを適切に調整するかどうかを決定するいくつかの要因があります。

パーティション・データベースでセルフチューニング・メモリーが使用可能にされると、単一のデータベース・パーティションがチューニング・パーティションとして指定され、メモリーのチューニングに関するすべての決定は、そのデータベース・パーティションのメモリーおよびワークロード特性に基づいて行われます。チューニングに関する決定がそのパーティションで行われると、メモリーの調整が他のデータベース・パーティションに分散され、すべてのデータベース・パーティションが同様の構成を保守することが保証されます。

この単一チューニング・パーティション・モデルでは、すべてのデータベース・パーティションでメモリー要件が類似している場合のみ、このフィーチャーが使用さ

れると想定します。以下に示すガイドラインは、パーティション・データベースでセルフチューニング・メモリーを使用可能にするかどうかを判別する際に使用します。

パーティション・データベースでセルフチューニング・メモリーが推奨される場合

すべてのデータベース・パーティションでメモリー要件が類似していて、それらのパーティションが類似のハードウェア上で稼働している場合、変更を加えずにセルフチューニング・メモリーを使用可能にすることができます。これらのタイプの環境では、以下の特性が共通しています。

- すべてのデータベース・パーティションが同一のハードウェア上に存在し、複数の論理データベース・パーティションが複数の物理データベース・パーティションに均一に分散されている
- 完璧かほぼ完璧に分散されているデータがある
- 複数のデータベース・パーティションでワークロードが均一に分散されている。つまり、他のデータベース・パーティションと比べて、1 つ以上のヒープに関するメモリー要件が高いパーティションは存在しません。

そのような環境では、すべてのデータベース・パーティションが同等に構成されていると、セルフチューニング・メモリーはシステムを適切に構成します。

パーティション・データベースでセルフチューニング・メモリーが制限付きで推奨される場合

環境内のほとんどのデータベース・パーティションでメモリー要件が類似していて、それらのパーティションが類似のハードウェア上で稼働している場合は、初期構成にいくらかの注意を払うかぎり、セルフチューニング・メモリーを使用することができます。これらのシステムには、データ用のデータベース・パーティションのセット 1 つと、少数のコーディネーター・パーティションおよびカタログ・パーティションのセットが存在する場合があります。このような環境では、コーディネーター・パーティションとカタログ・パーティションを、データを含むデータベース・パーティションとは異なった構成にすると便利です。

データを含むすべてのデータベース・パーティションでセルフチューニング・メモリーを使用可能にし、これらのデータベース・パーティションの 1 つをチューニング・パーティションとして指定する必要があります。加えて、コーディネーター・パーティションとカタログ・パーティションが異なる構成になっている可能性があるため、セルフチューニング・メモリーをこれらのパーティションで使用不可にする必要があります。コーディネーター・パーティションおよびカタログ・パーティションでセルフチューニング・メモリーを使用不可にするには、これらのパーティションで `self_tuning_mem` データベース構成パラメーターを OFF に設定してください。

パーティション・データベースでセルフチューニング・メモリーが推奨されない場合

各データベース・パーティションのメモリー要件が異なっている場合や、さまざまなデータベース・パーティションが大幅に異なるハードウェア上で稼働している場合は、セルフチューニング・メモリー・フィーチャーを使用不可にすることをお

勧めします。このフィーチャーを使用不可にするには、すべてのパーティションで `self_tuning_mem` データベース構成パラメーターを OFF にします。

異なるデータベース・パーティションのメモリー要件の比較

複数の異なるデータベース・パーティションのメモリー要件が十分に類似しているかどうかを判別する最良の方法は、スナップショット・モニターを参照することです。以下のスナップショット・エレメントがすべてのデータベース・パーティションで類似していれば (差が 20% 以下)、それらのデータベース・パーティションのメモリー要件はかなり類似していると見なせます。

以下のデータを収集するには、コマンド `get snapshot for database on <dbname>` を発行します。

```
Locks held currently           = 0
Lock waits                     = 0
Time database waited on locks (ms) = 0
Lock list memory in use (Bytes) = 4968
Lock escalations              = 0
Exclusive lock escalations     = 0

Total Shared Sort heap allocated = 0
Shared Sort heap high water mark = 0
Post threshold sorts (shared memory) = 0
Sort overflows                = 0

Package cache lookups         = 13
Package cache inserts         = 1
Package cache overflows       = 0
Package cache high water mark (Bytes) = 655360

Number of hash joins          = 0
Number of hash loops          = 0
Number of hash join overflows = 0
Number of small hash join overflows = 0
Post threshold hash joins (shared memory) = 0

Number of OLAP functions      = 0
Number of OLAP function overflows = 0
Active OLAP functions         = 0
```

以下のデータを収集するには、コマンド `get snapshot for bufferpools on <dbname>` を発行します。

```
Buffer pool data logical reads = 0
Buffer pool data physical reads = 0
Buffer pool index logical reads = 0
Buffer pool index physical reads = 0
Total buffer pool read time (milliseconds) = 0
Total buffer pool write time (milliseconds) = 0
```

パーティション・データベース環境でのセルフチューニング・メモリーの使用

パーティション・データベース環境でセルフチューニング・メモリーが有効な場合、データベース・パーティションが 1 つ (チューニング・パーティション と呼ばれる) 存在します。これは、メモリー構成をモニターして、構成変更があればそれを他のすべてのデータベース・パーティションに伝搬し、すべての関連するデータベース・パーティション間で構成の整合性を保持します。

チューニング・パーティションは、パーティション・グループのデータベース・パーティションの数およびバッファ・プールの数などの、幾つかの特性に基づいて選択されます。

- チューニング・パーティションとして現在指定されているデータベース・パーティションを判別するには、**ADMIN_CMD** プロシージャを以下のように呼び出します。

```
CALL SYSPROC.ADMIN_CMD('get stmm tuning dbpartitionnum')
```

- チューニング・パーティションを変更するには、**ADMIN_CMD** プロシージャを以下のように呼び出します。

```
CALL SYSPROC.ADMIN_CMD('update stmm tuning dbpartitionnum <partitionnum>')
```

チューニング・パーティションは、非同期で更新されるか、次のデータベース始動時に更新されます。メモリー・チューナーが自動的にチューニング・パーティションを選択するようにするには、*partitionnum* 値に -1 を入力します。

パーティション・データベース環境でのメモリー・チューナーの開始

パーティション・データベース環境では、メモリー・チューナーが開始されるのは **ACTIVATE DATABASE** コマンドによって明示的にデータベースがアクティブにされる場合に限ります。セルフチューニング・メモリーでは、すべてのパーティションがアクティブである必要があるからです。

指定のデータベース・パーティションでセルフチューニング・メモリーを無効にする

- データベース・パーティションのサブセットでセルフチューニング・メモリーを無効にするには、こうしたデータベース・パーティションに対して **self_tuning_mem** データベース構成パラメーターを OFF に設定します。
- 特定のデータベース・パーティションの構成パラメーターによって制御された、メモリー・コンシューマーのサブセットに対してセルフチューニング・メモリーを無効にするには、関連する構成パラメーターの値またはバッファ・プール・サイズの値を **MANUAL** またはそのデータベース・パーティションの特定の値に設定します。セルフチューニング・メモリーの構成パラメーターの値は、すべての稼働パーティション間で整合させることをお勧めします。
- 特定のデータベース・パーティションの特定のバッファ・プールでセルフチューニング・メモリーを無効にするには、**ALTER BUFFERPOOL** ステートメントを発行し、サイズ値と、セルフチューニング・メモリーを無効にするパーティションを指定します。

特定のデータベース・パーティション上のバッファ・プールのサイズを指定する **ALTER BUFFERPOOL** ステートメントは、そのバッファ・プールに関する例外項目を **SYSCAT.BUFFERPOOLDBPARTITIONS** カタログ・ビューに作成します (または既存の項目を更新します)。バッファ・プールの例外項目が存在する場合に、デフォルトのバッファ・プール・サイズが **AUTOMATIC** に設定されていると、そのバッファ・プールはセルフチューニング操作に関与しません。例外項目を除去して、バッファ・プールをセルフチューニング用に有効にするには、以下のようにします。

1. ALTER BUFFERPOOL ステートメントを発行して、バッファー・プール・サイズを特定の値に設定することにより、このバッファー・プールのセルフチューニングを使用不可にします。
2. 別の ALTER BUFFERPOOL ステートメントを発行し、このデータベース・パーティション上のバッファー・プールのサイズをデフォルトに設定します。
3. 別の ALTER BUFFERPOOL ステートメントを発行して、バッファー・プール・サイズを AUTOMATIC に設定することにより、このバッファー・プールのセルフチューニングを使用可能にします。

非均一環境でセルフチューニング・メモリーを有効にする

データをすべてのデータベース・パーティションに均一に配分し、各パーティションで実行されるワークロードが同様のメモリー要求量を持つのが理想的です。データ配分に偏りがあり、1 つ以上のデータベース・パーティションに他のデータベース・パーティションよりもかなり多い（または非常に少ない）データが含まれる場合、これらの変則的なデータベース・パーティションをセルフチューニング用に有効にするべきではありません。メモリー所要量がデータベース・パーティション間で偏っている場合にも同じことが当てはまります。これが生じる可能性があるのは、例えば、リソースを多く使用するソートが 1 つのパーティションでのみ実行される場合、または一部のデータベース・パーティションが別のハードウェアと関連付けられており、他のデータベース・パーティションよりも使用可能メモリーが多い場合です。このタイプの環境にある一部のデータベース・パーティションでは、セルフチューニング・メモリーを有効にすることができます。偏りがある環境でメモリーのセルフチューニングを活用するには、同様のデータおよびメモリー所要量のデータベース・パーティションのセットを識別し、セルフチューニング用にそれらを有効にします。残りのパーティションのメモリーは手動で構成してください。

バッファー・プール管理

バッファー・プールは、データベース・ページ用の作業メモリーおよびキャッシュを提供します。

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。ほとんどのページ・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。

アプリケーションが表の行にアクセスするときに、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・プールで探します。そのページがそこで見つからない場合、データベース・マネージャーはディスクからページを読み取り、それをバッファー・プールに入れます。その後、そのデータを使用して照会を処理できます。

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に行われます。データベースの活動化は、接続する最初のアプリケーションによって暗黙的に行われる場合もあります。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、サイズ変更、およびドロップすることができます。ALTER BUFFERPOOL ステートメントを使用すると、バッファー・プールのサイズを増やすことができます。デフォルトでは、十分なメモリーを使用できれば、ステートメントの実行直後にバッファー・プールのサイズが変更されます。ス

テートメントの実行時に十分なメモリーを使用できない場合は、データベースが再活動化されたときにメモリーが割り振られます。バッファー・プールのサイズを小さくする場合は、トランザクションが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。バッファー・プール・メモリーは、データベースが非アクティブ化されたときに解放されます。

あらゆる状況において適切なバッファー・プールを使用できるようにするため、DB2 は、それぞれ 4 KB、8 KB、16 KB、および 32 KB のページ・サイズを持つ小さなシステム・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・ファイルにはありません。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。

- 指定されたバッファー・プールが DEFERRED キーワードを使用して作成されたためにそれが開始されない場合、または必要なページ・サイズのバッファー・プールの作成に使用できるメモリーが不足しているためにそれが非アクティブである場合。

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、システム・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。

- データベース接続の試行中にバッファー・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。システム・バッファー・プールがあるため、DB2 は引き続き十分に機能しますが、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。

バッファー・プールを作成するときには、別のページ・サイズを明示的に指定していない限り、データベースの作成時に指定されたページ・サイズが指定されます。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。

db2mtrk コマンドを発行することにより起動できるメモリー・トラッカーによって、バッファー・プールに割り振られたデータベース・メモリーの量を表示できます。さらに、**GET SNAPSHOT** コマンドを使用してバッファー・プールの現行サイズ (**bp_cur_buffsz** モニター・エレメントの値) を調べることができます。

アクティビティのバッファー・プールの優先順位は、DB2 ワークロード・マネージャーによって提供されるワークロード管理機能のより大きなセットの一部として制御できます。詳細については、『DB2 ワークロード・マネージャーの概念の紹介』および『サービス・クラスのバッファー・プールの優先順位』を参照してください。

データ・ページのバッファ・プール管理

バッファ・プール・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。ページが更新中の場合、更新者以外はアクセスできません。しかし、ページが更新中でない場合、多数の同時リーダーがいてもかまいません。
- **ダーティー・ページ** には、変更されているがまだディスクに書き込まれていないデータが含まれています。

ページは、データベースがシャットダウンされるまで、またはあるページが占有しているスペースが別のページで必要になるまで、あるいはページがバッファ・プールから明示的にページされる (例えばオブジェクトのドロップ時のページ) まで、バッファ・プール内に残ります。別のページがスペースを必要とするときにどのページが除去されるかは、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
- 再びページが参照される可能性
- ページに含まれるデータのタイプ
- 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ていないか

クラッシュの発生時、またはオンライン・バックアップなどのデータベース操作の前には、`FLUSH BUFFERPOOLS` ステートメントを使用してデータベースのリカバリ・ウィンドウを縮小します。詳しくは、`FLUSH BUFFERPOOLS` ステートメントを参照してください。

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出されます。変更されてディスクに書き出されたページは、スペースが必要になるのでない限りバッファ・プールからは自動的に除去されません。

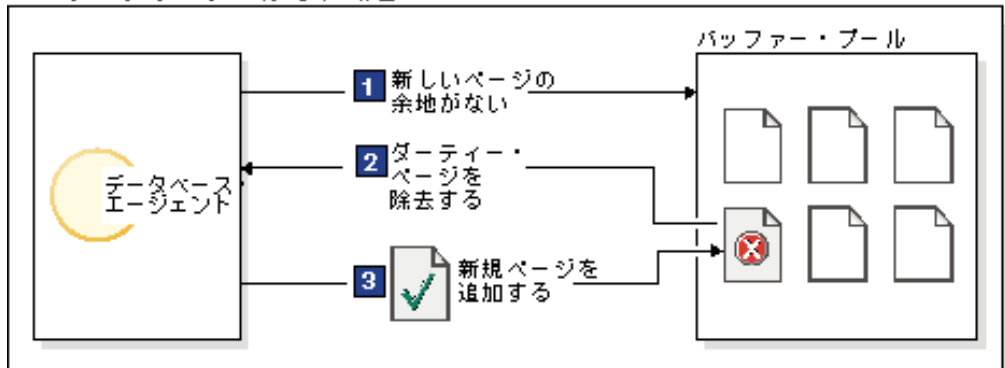
ページ・クリーナー・エージェント

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・ページをディスクに書き込むのはページ・クリーナー・エージェントです。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関係しておらず、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 **非同期バッファ書き込み機能** と呼ばれることもあります。

更新が頻繁に行われるワークロードでパフォーマンスを改善するために、先行ページ・クリーニングを使用可能にすることもできます。この方法では、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います。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。

図 19 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・プールの管理作業の分担方法を示しています。

ページ・クリーナーがない場合



ページ・クリーナーを使った場合

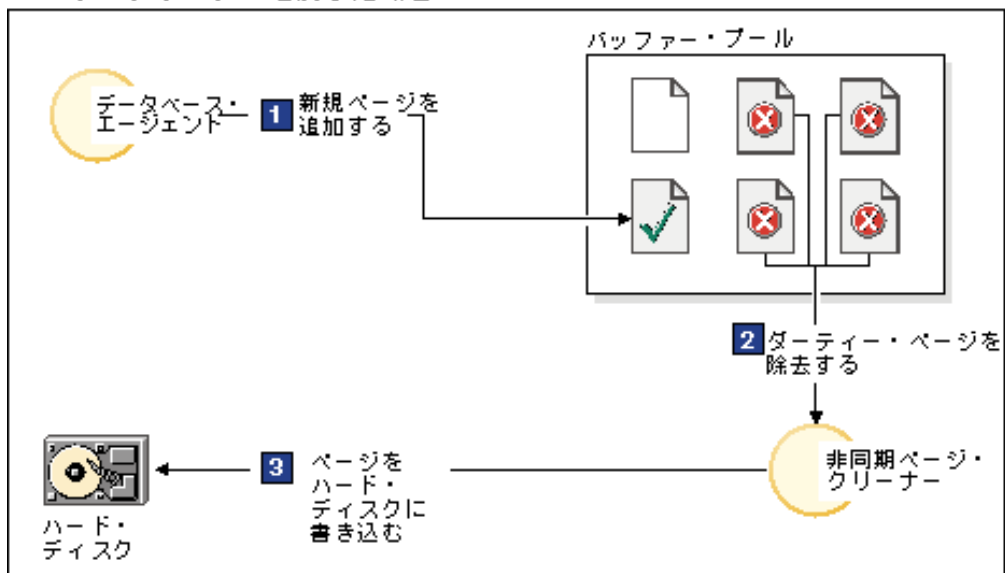


図 19. 非同期ページ・クリーニング： ダーティ・ページがディスクに書き出されます。

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することによって再構築できるバッファ・プールの部分より大きな部分を、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・プール内で一番古いデータ変更を記述するログ・レコード

ページ・クリーナーは、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値を超えないように、ダーティ・ページをディスクに書き込みます。

$\text{logfilsiz} * \text{softmax} / 100$ (4 KB ページ単位)

詳細は次のとおりです。

- **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。例えば、**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、**pool_lsn_gap_clns** (起動されたバッファ・プール・ログ・スペース・クリーナー) モニター・エレメントでこの情報が提供されます。先行ページ・クリーニングを使用可能にした場合にはこの条件が発生しないはずであるため、**pool_lsn_gap_clns** の値は 0 になります。

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。

複数のデータベース・バッファ・プールの管理

各データベースは最低 1 つのバッファ・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・プールを作成することもできます。

ALTER BUFFERPOOL ステートメントを使ってバッファ・プールのサイズを変更できます。

新規データベースは、IBMDEFAULTBP というデフォルト・バッファ・プールを持っています。そのデフォルト・ページ・サイズは、データベースの作成時に指定されたページ・サイズに基づいています。デフォルトのページ・サイズは、情報データベースの構成パラメーター **pagesize** として保管されます。デフォルトのページ・サイズで表スペースを作成したときにそれを特定のバッファ・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・プールに割り当てられます。デフォルトのバッファ・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。

バッファ・プールのページ・サイズ

データベースを作成またはアップグレードした後で、追加のバッファ・プールを作成できます。デフォルトとして 8 KB ページ・サイズのデータベースを作成した場合、デフォルト・バッファ・プールがデフォルト・ページ・サイズ (この場合、8 KB) で作成されます。別の方法として、バッファ・プールを 8 KB のページ・サイズで作成し、同時に同じページ・サイズの 1 つ以上の表スペースを作成することもできます。この方法では、データベースの作成時に 4 KB のデフォルト・ページ・サイズを変更する必要はありません。異なるページ・サイズを使用するバッファ・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クールに割り当てる必要があります。このバッファーク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・バッファークール (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システム・バッファーク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ク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クールに割り当てます。

CREATE BUFFERPOOL ステートメントでバッファークールを作成するときにサイズを指定しない場合は、バッファークール・サイズは AUTOMATIC に設定され、DB2 によって管理されます。後でバッファークール・サイズを変更するには、ALTER BUFFERPOOL ステートメントを使用します。

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ク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 ステートメントで指定された場合、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クールのサイズが 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。

大容量バッファークールの利点

大容量バッファーク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ク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・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。
- 頻繁に使用するディスク・ストレージ・デバイス (カタログ表や頻繁に参照するユーザー表および索引を格納するストレージ・デバイスなど) において、入出力の競合を回避できます。また、TEMPORARY 表スペースを格納するディスク・ストレージ・デバイス上の入出力の競合が減少すると、照会で必要になるソートの速度も速くなります。

バッファーク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、バッファークールを 1 つだけ使用してください。

- 合計バッファークール・スペースが 10 000 ページ (4 KB 単位) より少ない
- 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ある

それ以外のすべての状況では、以下の理由により、複数のバッファークールの使用を検討してください。

- TEMPORARY 表スペースを別のバッファークールに割り当てることができるので、一時記憶を必要とする照会 (特にソートが多い照会など) 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クールに割り当てて検討してください。このバッファーク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クールに分離し、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。例えば、頻繁に更新される表や索引を入れるバッファークールを、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クールとは別々にするという場合などに役立ちます。
- 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、小さなバッファーク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、バッファークール内にデータを保持しておく必要はありません。この種のデータには小さなバッファークールを用意して、余ったメモリーを他のバッファークールのために空けておく方が効果的です。

データを別々のバッファークールに分けて入れると、パフォーマンスが向上してコストが比較的少ないという診断データが、統計および会計トレースから生成される場合があります。

バッファークールが複数あるシステムの調整においては、セルフチューニング・メモリー・マネージャー (STMM) を使用すると理想的です。

開始時のバッファークール・メモリーの割り振り

バッファークールを作成するか、バッファーク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クールの割り振りができるように、すべてのバッファークールに必要なメモリーの合計量がデータベース・マネージャーで使用できなければなりません。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ク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。新規バッファークールを作成する際や既存のバッファークールのサイズを増やす際に DEFERRED 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。

このメモリーがデータベースの開始時に使用できなかった場合には、データベース・マネージャーは最小サイズの 16 ページのシステム・バッファークール (各ページ・サイズに 1 つ) だけを使用し、警告が返されます。データベースは、そ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、この状態が継続します。パフォーマンスは最高になっていない可能性があります。データベースへの接続、バッファークール・サイズの再構成、またはその他の重要なタスクの実行が可能です。これらのタスクが完了したら、データベースを再始動します。長時間この状態でデータベースの操作を行うことは避けてください。

システム・バッファ・プールのみでデータベースを開始することを避けるには、**DB2_OVERRIDE_BPF** レジストリー変数を使用して、使用可能メモリの使用を最適化します。

先行ページ・クリーニング

DB2 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。この方法を使用すると、特定の時点で書き出すダーティ・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います。

この先行ページ・クリーニング方式がデフォルトのページ・クリーニング方式と異なっているのは、主に次の 2 つの点です。

- ページ・クリーナーは、**chnngpgs_thresh** データベース構成パラメーターの値に反応しない。

スワップアウトされるページの数が増えすぎると、ページ・クリーナーがバッファ・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。

- ページ・クリーナーはログの発行するログ・シーケンス番号 (LSN) ギャップ・トリガーに反応しない。

バッファ・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** データベース構成パラメーターによって許可されている量を超えると、データベースで **LSN** ギャップが生じていると言います。

デフォルトのページ・クリーニング方法では、**LSN** ギャップを検出するログがページ・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、ページ・クリーナーは **softmax** の値で許可されているものよりも古いページを書き出します。ページ・クリーナーは交互に、アイドル状態と、多数のページを書き込む集中的なアクティビティ状態とになります。このために **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** ギャップが検出される以前に、ページ・クリーナーのクリーニング速度が遅いため、**DB2** のログ・スペースが不足してしまうこともあり得ます。

先行ページ・クリーニング方法では、長時間に渡って一定数の書き込みを分散させることによってこの動作を調整しています。**LSN** ギャップの原因となってい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** ギャップになりそうなページをもクリーニングすることにより、ページ・クリーナーはそうした調整を行います。

この新しいページ・クリーニング方法を使用するには、**DB2_USE_ALTERNATE_PAGE_CLEANING** レジストリー変数をオンにします。

更新パフォーマンスの向上

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。

このプロトコルには、以下のステップが含まれます。

1. 更新されるページが `pinned` (滞留) 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、この変更を取り消したり、再実行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ヘッダーに保管されます。
2. 更新がページに適用されます。
3. ページがアンラッチされます。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、ディスクにまだ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、そのデータがディスクに書き込まれます。

パフォーマンスを向上させるため、これらの入出力操作は、システム負荷が落ち着くまで、あるいはリカバリー可能性を保証したり、リカバリー時間を制限したりする必要があるときまで後回しにされます。特に、以下の場合にダーティー・ページがディスクに書き込まれます。

- 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ている場合
- ページ・クリーナーがページで作業している場合。これが生じる可能性があるのは、次のような場合です。
 - 他のエージェントがそのページをスワップアウトされる対象として選択している場合
 - **chngpgs_thresh** データベース構成パラメーター値を超える場合。これにより、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・クリーニングをデータベースで使用可能である場合は、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値を超える場合。これにより、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。
 - クリーン・ページの数が高すぎる場合。ページ・クリーナーは、先行ページ・クリーニング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 LSN 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場合。ページ・クリーナーは、先行ページ・クリーニングのこの条件に対してのみ反応します。
- ページが、`NOT LOGGED INITIALLY` 節で定義された表の一部であり、更新の後に `COMMIT` ステートメントが続く場合。 `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。

バッファー・プール・ヒット率の計算数式

バッファー・プール・ヒット率は、照会で必要とされたデータを外部ストレージから読み取る必要があった頻度と比較して、メモリー内で検出された頻度を示します。ヒット・レートおよびヒット率は、バッファー・プール・モニター・エレメントに基づいた数式を使用して計算できます。

ローカル・バッファ・プール

表2. ローカル・バッファ・プール・ヒット率の数式： これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファ・プール・ヒット率の計算式
データ・ページ	$(\text{pool_data_lbp_pages_found} - \text{pool_async_data_lbp_pages_found} - \text{pool_temp_data_l_reads}) / \text{pool_data_l_reads} \times 100$
索引ページ	$((\text{pool_index_lbp_pages_found} - \text{pool_async_index_lbp_pages_found} - \text{pool_temp_index_l_reads}) / \text{pool_index_l_reads}) \times 100$
一時データ・ページ	$(\text{pool_temp_data_l_reads} - \text{pool_temp_data_p_reads}) / \text{pool_temp_data_l_reads} \times 100$
一時索引ページ	$(\text{pool_temp_index_l_reads} - \text{pool_temp_index_p_reads}) / \text{pool_temp_index_l_reads} \times 100$
XML ストレージ・オブジェクト (XDA) ページ	$(\text{pool_xda_gbp_l_reads} - \text{pool_xda_gbp_p_reads}) / \text{pool_xda_gbp_l_reads} \times 100$
全体のヒット率	$((\text{pool_data_lbp_pages_found} + \text{pool_index_lbp_pages_found} + \text{pool_xda_lbp_pages_found} - \text{pool_async_data_lbp_pages_found} - \text{pool_async_index_lbp_pages_found} - \text{pool_async_xda_lbp_pages_found}) / (\text{pool_data_l_reads} + \text{pool_index_l_reads} + \text{pool_xda_l_reads} + \text{pool_temp_data_l_reads} + \text{pool_temp_xda_l_reads} + \text{pool_temp_index_l_reads})) \times 100$

ローカル・バッファ・プール (環境)

環境で LBP のヒット率を計算するときには、GBP に依存しないデータ・ページのヒット率も考慮する必要があります。データ・ページがから要求されると、まず LBP が調べられます。ページが LBP で検出された場合は、**pool_data_lbp_pages_found** モニター・エレメントが増分されます。ページが LBP で検出されなかった場合は、ディスクから直接読み取られ、**pool_data_p_reads** モニター・エレメントが増分されます。ただし、GBP に依存するページがローカル・バッファ・プールで検出された場合には、**pool_data_lbp_pages_found** モニターのカウンターも増分されます。このような場合にこれらのページ・アクセス・カウントを区別する唯一の方法として、新規モニター・エレメント **pool_data_gbp_indep_pages_found_in_lbp** が用意されています。

表3. GBP には依存しない (LBP) ヒット率の数式： これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファ・プール・ヒット率の計算式
データ・ページ	$(\text{pool_data_gbp_indep_pages_found_in_lbp} / (\text{pool_data_gbp_indep_pages_found_in_lbp} + \text{pool_data_p_reads} + \text{pool_temp_data_p_reads} - \text{pool_data_gbp_p_reads})) \times 100$
索引ページ	$(\text{pool_index_gbp_indep_pages_found_in_lbp} / (\text{pool_index_gbp_indep_pages_found_in_lbp} + \text{pool_index_p_reads} + \text{pool_temp_index_p_reads} - \text{pool_index_gbp_p_reads})) \times 100$
XML ストレージ・オブジェクト (XDA) ページ	$(\text{pool_xda_gbp_indep_pages_found_in_lbp} / (\text{pool_xda_gbp_indep_pages_found_in_lbp} + \text{pool_xda_p_reads} + \text{pool_temp_xda_p_reads} - \text{pool_xda_gbp_p_reads})) \times 100$

GBP に依存しないページの場合は、LBP サイズを調整するとヒット率に影響が生じます。一時ページ・アクセスや NOT LOGGED INITIALLY 操作のような GBP に依存しない特定のタイプの操作では、GBP に依存しないページ・ヒット率をモニターすることができるでしょう。

別の方法として、GBP に依存するページ・エージェントのローカル・バッファ・プール・ヒット率を計算するために、以下の数式を使用することもできます。

表4. GBP に依存する (LBP) ヒット率の数式： これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファークール・ヒット率の計算式
データ・ページ	$((\text{pool_data_lbp_pages_found} - \text{pool_data_gbp_indep_pages_found_in_lbp}) / (\text{pool_data_l_reads} - \text{pool_data_gbp_indep_pages_found_in_lbp} - (\text{pool_data_p_reads} - \text{pool_data_gbp_p_reads}))) \times 100$
索引ページ	$((\text{pool_index_lbp_pages_found} - \text{pool_index_gbp_indep_pages_found_in_lbp}) / (\text{pool_index_l_reads} - \text{pool_index_gbp_indep_pages_found_in_lbp} - (\text{pool_index_p_reads} - \text{pool_index_gbp_p_reads}))) \times 100$
XML ストレージ・オブジェクト (XDA) ページ	$((\text{pool_xda_lbp_pages_found} - \text{pool_xda_gbp_indep_pages_found_in_lbp}) / (\text{pool_xda_l_reads} - \text{pool_xda_gbp_indep_pages_found_in_lbp} - (\text{pool_xda_p_reads} - \text{pool_xda_gbp_p_reads}))) \times 100$

両方のバッファークールの調整方法を決定したり、 GBP および LBP の調整後に結果を妥当性検査したりする場合に、 LBP ヒット率および GBP ヒット率の両方を比較する必要があります。

グループ・バッファークール (環境)

環境におけるグループ・バッファークール・ヒット率を計算するための数式は、他の環境で使用されるヒット率の数式とは異なります。このような相違が生じるのは、のにおけるグループ・バッファークールが、データのページを取得するためにそれぞれの内のローカル・バッファークールを扱う方法に起因しています。バッファークール・モニター・エレメントに基づく以下の数式を使用すると、ローカル・バッファークールとグループ・バッファークールの両方に関するデータ・ページ、索引ページ、XML ストレージ・オブジェクト・ページのヒット率を計算できます。

表5. グループ・バッファークール (GBP) ヒット率の数式： これらの数式は、ヒット率をパーセンテージで表します。

ページのタイプ	バッファークール・ヒット率の計算式
データ・ページ	$((\text{pool_data_gbp_l_reads} - \text{pool_data_gbp_p_reads}) / \text{pool_data_gbp_l_reads}) \times 100$
索引ページ	$((\text{pool_index_gbp_l_reads} - \text{pool_index_gbp_p_reads}) / \text{pool_index_gbp_l_reads}) \times 100$
XML ストレージ・オブジェクト (XDA) ページ	$((\text{pool_xda_gbp_l_reads} - \text{pool_xda_gbp_p_reads}) / \text{pool_xda_gbp_l_reads}) \times 100$
全体のヒット率	$((\text{pool_data_gbp_l_reads} + \text{pool_index_gbp_l_reads} + \text{pool_xda_gbp_l_reads} - \text{pool_data_gbp_p_reads} - \text{pool_index_gbp_p_reads} - \text{pool_xda_gbp_p_reads}) / (\text{pool_data_gbp_l_reads} + \text{pool_index_gbp_l_reads} + \text{pool_xda_gbp_l_reads})) \times 100$

バッファークール・ヒット率を計算するための前述の数式に加え、下記の数式を使用すると、プリフェッチされた時間ページが GBP で検出されたパーセンテージが示されます。

データ・ページのプリフェッチ

$$((\text{pool_async_data_gbp_l_reads} - \text{pool_async_data_gbp_p_reads}) / \text{pool_async_data_gbp_l_reads}) \times 100$$

索引ページのプリフェッチ

$$((\text{pool_async_index_gbp_l_reads} - \text{pool_async_index_gbp_p_reads}) / \text{pool_async_index_gbp_l_reads}) \times 100$$

XML ストレージ・オブジェクト (XDA) ページのプリフェッチ

$$\frac{(\text{pool_async_xda_gbp_l_reads} - \text{pool_async_xda_gbp_p_reads})}{\text{pool_async_xda_gbp_l_reads}} \times 100$$

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、1 つ以上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。

プリフェッチには次の 3 つのカテゴリがあります。

- 順次プリフェッチ では、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・プールに読み込みます。
- 先読みプリフェッチ は、索引内で先読みし、ISCAN-FETCH および索引スキャン操作がアクセスするページを正確に判定し、それらのページをプリフェッチします。
- リスト・プリフェッチ (リスト順次プリフェッチ と呼ばれる) では、連続していない一連のデータ・ページが効率的にプリフェッチされます。

スマート・データ・プリフェッチ は、データ・クラスタリングの度合いに応じて順次検出または先読みプリフェッチを使用する方法です。順次検出プリフェッチはデータ・ページが順に格納されている場合に使用され、先読みプリフェッチはデータ・ページが不良な状態にクラスタ化されている場合に使用されます。スマート・データ・プリフェッチにより、データベース・システムは、連続したページにデータが格納されている場合に可能になるパフォーマンス上の利点を十分に活用できます。また、不良な状態にクラスタ化されているデータも効率良くプリフェッチできます。不良な状態にクラスタ化されているデータが照会のパフォーマンスに悪影響を及ぼすことはなくなるため、表の再編成など、コストがかかる操作を行う必要は少なくなります。

スマート索引プリフェッチ は、索引の密度に応じて順次検出と先読みプリフェッチのいずれかを使用する方法です。順次検出プリフェッチは索引が順に格納されている場合に使用され、先読みプリフェッチは索引の密度が低い場合に使用されます。スマート索引プリフェッチにより、データベース・システムは、索引が順に格納されている場合に可能になるパフォーマンス上の利点を十分に活用できます。また、密度が低い索引も効率良くプリフェッチできます。スマート索引プリフェッチでは、索引の再編成など、コストがかかる操作を行う必要は少なくなります。

スマート・データ・プリフェッチとスマート索引プリフェッチのサポートは索引スキャン操作にのみ適用され、XML 索引、拡張索引、および Text Search テキスト索引はサポートしません。スマート・データ・プリフェッチは、論理索引であって物理索引ではないため、グローバルな範囲がクラスタ化された表の索引のスキャン中は使用できません。また、スマート・データ・プリフェッチでは、ISCAN-FETCH によりグローバル範囲パーティション索引がスキャンされる場合、データの先読みプリフェッチは使用されません。スマート・データ・プリフェッチのための索引スキャン中に索引の述部が評価され、オプティマイザーによってその索引スキャンの

対象となる行が多くないと判断された場合は、先読みプリフェッチは使用不可になります。また、スマート索引プリフェッチを、範囲がクラスター化された表の索引に使用することもできません。

注: オプティマイザーは、ISCAN-FETCH または索引スキャン操作で使用可能にすべきデータまたは索引のプリフェッチ・タイプを判別します。次に示すのは、スマート・データ・プリフェッチおよびスマート索引プリフェッチのプリフェッチ技法の種類です。

- 順次検出プリフェッチ
- 先読みプリフェッチ
- 順次先読みプリフェッチ

この技法では、順次プリフェッチと先読みプリフェッチの両方が使用可能です。これはデフォルトです。この技法では、プリフェッチされないページのしきい値に到達するまで、または場合によってはオプティマイザーによる MAXPAGES 見積もりを超えるまでは、初めに順次検出プリフェッチが使用されます。プリフェッチされないページのしきい値に到達するか、MAXPAGES 見積もりを超えた場合、先読みプリフェッチが使用可能になります。

ただし、索引スキャンが完全修飾キーに対して行われ、索引がユニークであり、索引スキャンの対象としてオプティマイザーが見積もったページ数 (MAXPAGES) が 1 ページ以下である場合、オプティマイザーはどのプリフェッチ技法も選択しないことがあります。この場合、MAXPAGES 見積もりは信頼できる可能性が高く、プリフェッチは必要ありません。

データ・ページのプリフェッチはデータベース・マネージャー・エージェントの読み取りとは異なります。このエージェントの読み取りは、1 ページまたは少数の連続ページが取り出されるときに使用されますが、アプリケーションに転送されるのは 1 ページのデータのみです。

プリフェッチと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重要な影響を与えます。そのような並列スキャンによりデータの消費率が高くなり、それによってさらに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である場合、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なります。順次スキャンの際にプリフェッチが行われない場合には、エージェントが入出力を待機するので照会の実行速度が遅くなります。並列スキャンの際に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間、すべてのサブエージェントが待機しなければならないことがあります。

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、プリフェッチ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。

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・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。

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。表のスキャンや表のソートなどの場合、データベース・マネージャーは適切なタイプのプリフェッチを選択します。例えば、以下の例ではおそらく表スキャンが必要になるので、順次プリフェッチに適した候補と言えます。

```
SELECT NAME FROM EMPLOYEE
```

スマート・データ・プリフェッチおよびスマート索引プリフェッチでは、順次プリフェッチと先読みプリフェッチの両方が使用可能です。これはデフォルトです。プリフェッチされないページのしきい値、または場合によってはオプティマイザーによる MAXPAGES 見積もりを超えるまで、初めに順次検出プリフェッチが使用されます。プリフェッチされないページのしきい値に到達するか、MAXPAGES 見積もりを超えた場合、先読みプリフェッチが使用可能になります。

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、瞬時には判断しにくい場合があります。そのような場合、データベース・マネージャーは入出力をモニターし、順次ページ読み取りが行われている場合にはプリフェッチを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 (順次検出 と言う) は、索引ページとデータ・ページの両方に適用されます。データベース・マネージャーが順次検出プリフェッチを実行するか先読みプリフェッチを実行するかを制御するには、**seqdetect** データベース構成パラメーターを使用します。

例えば、順次検出が使用可能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。同様に、多数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検出した場合には、索引ページのプリフェッチが行われると考えられます。

表スペースの PREFETCHSIZE オプションの含意

CREATE TABLESPACE または ALTER TABLESPACE のいずれかのステートメントで PREFETCHSIZE 節を使用すると、データ・プリフェッチの実行中に表スペースから読み取られるプリフェッチ・ページ数を指定できます。指定した値 (または「AUTOMATIC」) は、SYSCAT.TABLESPACES カタログ・ビューの PREFETCHSIZE 列に格納されます。

PREFETCHSIZE の値には、表スペース・コンテナの数、各コンテナ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 (データベース・マネージャーが別のコンテナを使用する前にコンテナに書

き込むページ数) を掛け合わせた値を、明示的に設定することをお勧めします。例えば、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を 2 つ持っている場合には、プリフェッチ・サイズを 32 ページに設定することができます。各コンテナに 5 つの物理ディスクがある場合は、プリフェッチ・サイズを 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・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、別の作業単位に必要なページが、バッファ・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、表スペースに対して指定された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。表スペースのプリフェッチ・サイズを調整する助けとして、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。次のような情報を集めることができます。

- 照会時に入出力待機時間が生じているか (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)
- プリフェッチが行われているか (データベース・システム・モニターによって提供されている `pool_async_data_reads` (バッファ・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り)

照会時にデータのプリフェッチが行われているのに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、エクステントが別個のコンテナにある場合には、複数の入出力操作を並列に実行できます。より高いパフォーマンスを得るには、コンテナが別個の物理装置を使用するように構成します。

先読みプリフェッチ:

先読みプリフェッチは索引で先読みをして、`ISCAN-FETCH` と索引スキャン操作によってアクセスする正確なデータ・ページと索引リーフ・ページを判別し、それらをプリフェッチします。

先読みプリフェッチでは索引のスキャン中に必要なすべてのデータと索引ページが提供される (不要なページは提供されない) 一方で、これらのページを探すために追加のリソースが必要になります。高度な順次データおよび索引の場合、通常は順次検出プリフェッチのほうが先読みプリフェッチよりも効率的です。

スマート・データ・プリフェッチおよびスマート索引プリフェッチでは、順次プリフェッチと先読みプリフェッチの両方が使用可能です。これはデフォルトです。プリフェッチされないページのしきい値、または場合によってはオプティマイザーによる `MAXPAGES` 見積もりを超えるまで、初めに順次検出プリフェッチが使用されます。プリフェッチされないページのしきい値に到達するか、`MAXPAGES` 見積もりを超えた場合、先読みプリフェッチが使用可能になります。

seqdetect データベース構成パラメーターを使用して、データベース・マネージャーが先読みプリフェッチを実行するか順次検出プリフェッチを実行するかを制御します。

制約事項

索引のスキャン中に索引述部を評価する必要がある場合で、特定の索引スキャンを行う索引述部の複合選択率が 90% 未満である (対象となる行が多くない) とオプティマイザーが判断した場合、その索引スキャンに対するデータの先読みプリフェッチは使用不可になります。これは特定の索引スキャンに対するすべての索引述部を考慮した複合選択度であることに注意してください。述部の選択度が低い場合に、照会オプティマイザーが先読みプリフェッチを使用可能にした場合、不要なページを数多くプリフェッチすることになる可能性があります。

データの先読みプリフェッチは、範囲パーティション表で非パーティション索引をスキャンしている間も使用不可になります。これは、プリフェッチ要求に、複数のパーティションからのページ参照が含まれないようにするためです。

スマート・データ・プリフェッチとスマート索引プリフェッチの場合は、先読みプリフェッチが使用できます。その場合、これらのプリフェッチ技法は索引スキャン操作にのみ適用され、XML 索引、拡張索引、および Text Search テキスト索引はサポートしません。

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高い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ます。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트는、コンテナ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・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・プール・メモリーを効率的に使用することができます。バッファ・プール・メモリーは、以下の場合に浪費される可能性があります。

- 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・プールに指定されたブロック・サイズより小さい場合。
- 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・プールのページ・エリアにすでに存在している。

入出力サーバーは各バッファ・プール・ブロックの一部のページが無駄に消費されるのを許容しますが、無駄に消費されるページが多過ぎる場合、入出力サーバーはブロックに基づかないでバッファ・プールのページ・エリアにプリフェッチし、結果として最適なパフォーマンスは得られません。

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・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・プールを作成するには、`CREATE BUFFERPOOL` または `ALTER BUFFERPOOL` ステートメントを使用します。

注: ブロック・ベースのバッファ・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・プールのブロック・エリアは無駄になります。

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、データ・ページが連続していない場合でも、効率的にこれらの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使用できます。

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。例えば、オプティマイザーは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・ページを決定できます。

```
INDEX IX1: NAME    ASC,
            DEPT   ASC,
            MGR    DESC,
            SALARY DESC,
            YEARS  ASC
```

その後、次の検索基準を使用します。

```
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、索引スキャンで取得した RID のリストをソートするステップが組み込まれません。

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、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 のアクティビティ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。

十分な数の入出力サーバーを構成すると (**num_ioservers** データベース構成パラメーターを使用)、プリフェッチを有効活用でき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・ページはパフォーマンスに影響を与えることなくページアウトされます。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、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
- 並列に作業できる入出力サーバーの最大数

データベース・マネージャーがシステム構成に基づくインテリジェント値を選択できるようにするため、**num_ioservers** の値を **AUTOMATIC** に設定することを考慮してください。

並列入出力を使用したプリフェッチの図:

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、入出力サーバーが使用されます。

133 ページの図 20 にこのプロセスが示されています。

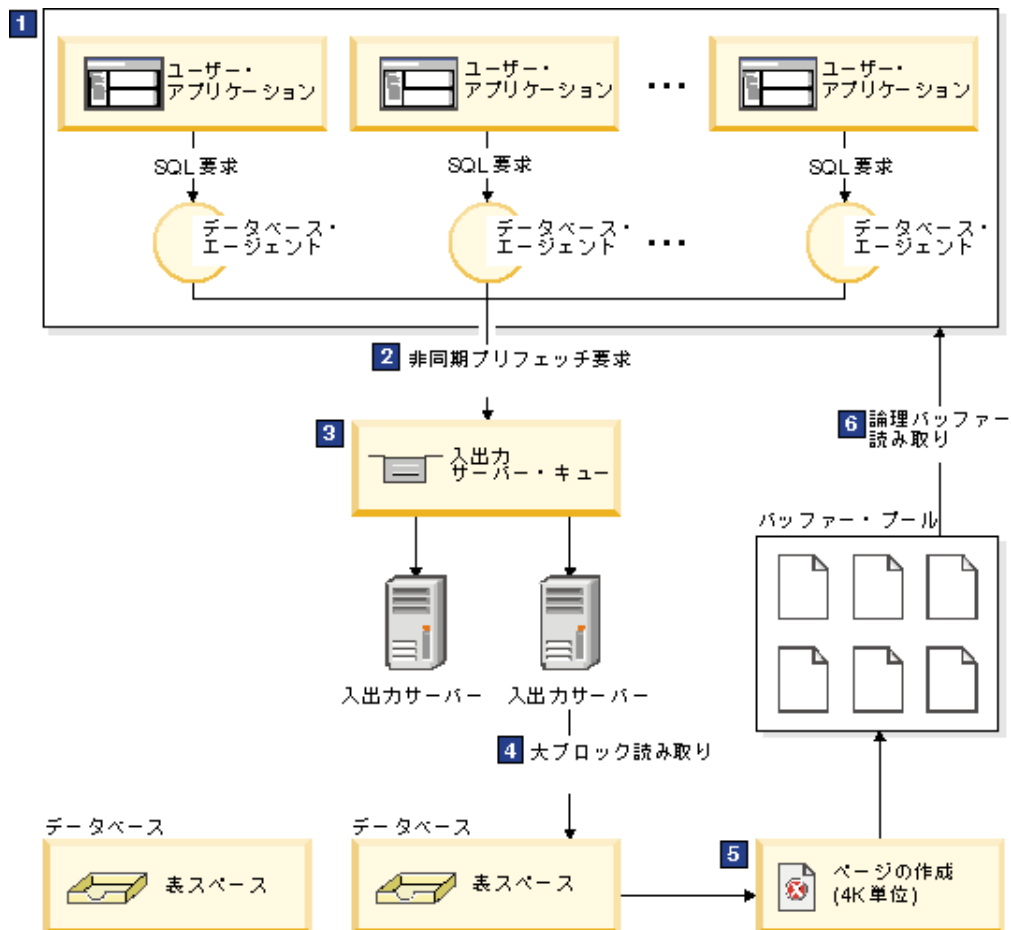


図 20. 入出力サーバーを使用したデータのプリフェッチ

- 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・エージェントに要求を渡します。
- 2、3** データベース・エージェントは、要求を満たすために必要なデータを取得するためにプリフェッチを使用する必要があると判断し、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・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で指定された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・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。

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、単一の照会の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、複数のコンテナを並列で読み取ることができます。並列入出力により、入出力スループットが大幅に向上する場合があります。

それぞれの入出力サーバーはコンテナごとにワークロードを処理できますが、並列入出力を実行できる入出力サーバーの実際数は、要求されたデータが分散されている物理装置の数が限度となります。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 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したがって、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。
- 先読みプリフェッチ の場合、データ・ページの各リストは、エクステント境界に従って、より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。
- 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にしたがって、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およびリストア の場合、並列入出力要求の数は、バックアップ・バッファ・サイズをエクステント・サイズで除算した数 (ただし最大値はコンテナの数) 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 の場合、並列入出力要求は、順次プリフェッチで行われるのと同じ方法で開始され、分割されます。データはバッファ・プールにリストアされません。リストア・バッファからディスクに直接移動します。
- データをロード する場合、DISK_PARALLELISM コマンド・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・コンテナを累積した数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・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。
- コンテナが、別々の物理ドライブに存在すること。

- 並列処理の多重度に一貫性を持たせるために、すべてのコンテナが同じサイズであること。

他のコンテナより小さいコンテナが 1 つ以上ある場合には、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。次の例を考慮してください。

- 他より小さいコンテナが満杯になると、それから先のデータは残りの、他のコンテナに保管されることになるので、コンテナ間の均衡がとれなくなります。コンテナが不均衡になると、データのプリフェッチができるコンテナの数が、コンテナ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、小さなコンテナには他のコンテナ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 つのコンテナが他より大きいときに、他のコンテナがすべて満杯になったとすると、その大きいコンテナが追加のデータを保管するための唯一のコンテナになります。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ません。
- 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、複数のプロセッサで照会を実行することによって、照会の経過時間を短縮することができます。各プロセッサ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、その入出力容量を確保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、およびプロセッサの速度によって異なります。

IOCP の構成 (AIX):

DB2 for Linux, UNIX, and Windows ソフトウェアのインストール済み環境では、AIX サーバー上で入出力完了ポート (IOCP) を構成する必要はありません。ただし、この構成手順は、パフォーマンス改善の目的では推奨されます。DB2 インスタンスに加わるホストごとに、以下の手順を実行する必要があります。

始める前に

db2setup コマンドの実行による DB2 ソフトウェアのインストールによって、IOCP が使用可能になっていて、IOCP ポートの状況が「Available」に設定されていること。

手順

1. **lslpp** コマンドを入力して、システムに IOCP モジュールがインストールされているかどうかを確認します。

```
$ lslpp -l bos.iocp.rte
```

出力結果は、以下の例に類似したものになるはずです。

Fileset	Level	State	Description
Path: /usr/lib/objrepos bos.iocp.rte	5.3.9.0	APPLIED	I/O Completion Ports API
Path: /etc/objrepos bos.iocp.rte	5.3.0.50	COMMITTED	I/O Completion Ports API

2. **lsdev** コマンドを入力し、IOCP ポートの状況が Available かどうかを確認します。

```
$ lsdev -Cc iocp
```

出力結果は、以下の例と一致するはずです。

```
iocp0 Available I/O Completion Ports
```

IOCP ポート状況が Defined の場合は、状況を Available に変更します。

- a. root としてサーバーにログインし、次のコマンドを発行します。

```
# smitty iocp
```

- b. 「入出力完了ポートの特性の変更/表示 (Change / Show Characteristics of I/O Completion Ports)」を選択します。

- c. システム再始動時に構成される状態を「定義済み (Defined)」から「使用可能 (Available)」に変更します。この変更を有効にするには、システム・リブートを実行するか、**cfgmgr** コマンドを発行します。

- d. **lsdev** コマンドを再入力して、IOCP ポートの状況が Available に変更されていることを確認します。

最初のユーザー接続のシナリオでのデータベース非アクティブ化動作

ユーザーが最初に接続するときに、データベースは活動化されます。単一パーティション環境では、データベースがメモリーにロードされ、最後のユーザーが切断するまでこの状態のままになります。同じ動作が複数パーティション環境にも適用されます。この場合、最初のユーザー接続によってデータベースが、そのデータベースのローカル・パーティションとカタログ・パーティションの両方で活動化されます。

最後のユーザーが切断すると、データベースのシャットダウンが、ローカル・パーティションと、このユーザーがそのデータベースにとって最後のアクティブ・ユーザー接続であるリモート・パーティションの両方で行われます。最初の接続と最後の切断に基づいた、このデータベースの活動化および非アクティブ化は、**暗黙的活動化**とも呼ばれています。活動化は最初のユーザー接続によって開始され、ユーザーが **CONNECT RESET** (これにより、データベースが暗黙的に非アクティブ化されるようになる) を実行するまで (またはユーザーが接続を終了またはドロップするまで)、活動化は有効なままです。

データベースをメモリーにロードするプロセスは、これに大きく関係しています。これは、バッファー・プールを含むすべてのデータベース・コンポーネントの初期化を包含しており、特にパフォーマンス重視の環境では最小限にとどめなければならないタイプの処理です。この動作は、複数パーティション環境で特に重要なものです。その環境では、1 つのデータベース・パーティションから発行された照会

が、ターゲット・データ・セットの一部を含む他のパーティションに達します。これらのデータベース・パーティションは、ユーザー・アプリケーションの接続および切断動作に応じて、活動化または非アクティブ化されます。ユーザーが、初めてデータベース・パーティションに達する照会を発行するとき、照会はそのパーティションを最初に活動化するコストを想定します。そのユーザーが切断するとき、それに先だって他の接続がそのリモート・パーティションに対して確立されていなかった場合、データベースは非アクティブ化されます。次の着信照会がそのリモート・パーティションにアクセスする必要がある場合、そのパーティション上のデータベースは、最初に活動化される必要があります。このコストは、データベース（または該当する場合には、データベース・パーティション）の活動化および非アクティブ化ごとに生じます。

この動作への唯一の例外が生じるのは、ユーザーが **ACTIVATE DATABASE** コマンドを発行することにより、データベースを明示的に活動化することを選択する場合です。このコマンドが正常に完了した後、最後のユーザーが切断したとしても、データベースはメモリーに残っています。これは、単一パーティション環境と複数パーティション環境の両方に適用されます。そのようなデータベースを非アクティブ化するには、**DEACTIVATE DATABASE** コマンドを発行してください。どちらのコマンドも有効範囲内でグローバルです。つまり、該当する場合には、すべてのデータベース・パーティション上でデータベースを活動化または非アクティブ化します。データベースをメモリーにロードすることは処理の負担が大きいという性質を考え、データベース接続を介した暗黙的な活動化に依存するよりも、**ACTIVATE DATABASE** コマンドを使用して明示的にデータベースを活動化することを考慮してください。

ソート・パフォーマンスのチューニング

照会において、ソート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・ヒープを正しく構成することは、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場合（例えば、**ORDER BY** 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る場合

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因子があります。

- 次の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**)、および共有ソートのソート・ヒープしきい値 (**sheapthres_shr**)。これらは、そのインスタンス全体をソートするために使用できる合計メモリー量を制御します。
- 大量のソートを必要とするワークロードにおけるステートメントの数
- 不要なソートを避けるのに役立つ索引の存在と不在

- ソートの必要を最小限に抑える機能がないアプリケーション論理の使用
- 並列ソート。これはソートのパフォーマンスを向上させますが、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しか行えません。
- ソートがオーバーフロー するかどうか。ソートされたデータがソート・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) に収まらない場合は、データベースが所有する一時表にデータがオーバーフローします。
- ソートの結果がパイプ接続 されるかどうか。ソートされたリストを保管する一時表を使わなくてもソートされたデータを直接返せる場合は、パイプ・ソートになります。

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・ヒープは解放されません。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。

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。

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。

1. 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。
2. 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。

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もできます。

3. これらのステートメントを設計アドバイザーへの入力として使用します。これにより、ソートの必要を減らす索引が識別され、作成できます。

セルフチューニング・メモリー・マネージャー (STMM) を使用すると、ソートに必要なメモリー・リソースを、自動的かつ動的に、割り振りおよび割り振り解除できます。この機能を使用するには、以下のことを行ってください。

- **self_tuning_mem** 構成パラメーターを ON に設定して、データベースのセルフチューニング・メモリーを使用可能にします。
- **sortheap** および **sheapthres_shr** 構成パラメーターを AUTOMATIC に設定します。
- **sheapthres** 構成パラメーターを 0 に設定します。

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して、**sortheap**、**sheapthres_shr**、および **sheapthres** 構成パラメーターの設定に役立てることもできます。データベース・マネージャーごとに、またデータベースごとに、次のことを実行してください。

1. 代表的なワークロードを設定し稼働します。

2. 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・ヒープ合計 (**sort_heap_allocated** モニター・エレメントの値)
 - アクティブ・ソートおよびアクティブ・ハッシュ結合 (**active_sorts** および **active_hash_joins** モニター・エレメントの値)
3. データベースごとに、**sortheap** を 使用中のソート・ヒープ合計 の平均値に設定します。

注: ソートで長いキーが使用される場合、**sortheap** 構成パラメーターの値を増やす必要があるかもしれません。

4. **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。
 - a. 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。
 - b. 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。

- c. **sheapthres** には、アクティブ・ソートの平均数に、上記で算出したソート・ヒープの平均サイズを掛けた値を設定します。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。

IBM InfoSphere Optim Query Workload Tunerは、単一の SQL ステートメントのパフォーマンスと SQL ステートメント・グループのパフォーマンス (照会ワークロードと呼ばれる) を向上するツールを提供しています。この製品について詳しくは、<http://www.ibm.com/software/data/optim/query-workload-tuner-db2-luw/index.html>の製品概要のページを参照してください。また、バージョン 3.1.1 以降では、ワークロード設計アドバイザーを使用して、DB2 設計アドバイザー・ウィザードで利用できた多くの操作を実行できます。詳しくは、<http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.datatools.qrytune.workloadtunedb2luw.doc/topics/genrecsdsgn.html>のワークロード設計アドバイザーの資料を参照してください。

データ編成

時間と共に、レコードがより多くのデータ・ページに分散されるため、表のデータはフラグメント化し、表と索引のサイズが増加します。これによって、照会実行中に読み取る必要のあるページ数が増加します。表と索引の再編成によってデータはコンパクトになり、無駄なスペースが再利用されて、データ・アクセスが改善されます。

手順

索引または表の再編成を実行するためのステップは、次のとおりです。

1. 再編成しなければならない表や索引があるか判断します。
2. 再編成の方式を選択します。
3. 対象となるオブジェクトの再編成を実行します。
4. オプション: 再編成の進行状況をモニターします。

5. 再編成に成功したかどうかを確認します。 オフラインでの表再編成および索引再編成の場合、操作は同期になり、結果は操作完了時に明らかになります。オンラインでの表再編成の場合、操作は非同期になり、詳細は履歴ファイルで確認できます。
6. 再編成されたオブジェクトについての統計を収集します。
7. 再編成されたオブジェクトにアクセスするアプリケーションを再バインドします。

表の再編成

表データに多くの変更を加えた後、順序どおりになっていないデータ・ページ上に論理上の順次データが置かれることがあるので、データベース・マネージャーはデータにアクセスするために追加の読み取り操作を実行することが必要な場合があります。また、多数の行を削除した場合には、追加の読み取り操作が必要になります。この場合、表を索引と一致させ、スペースを再利用するために、表を再編成することを検討できます。

また、システム・カタログ表も再編成できます。

通常、表の再編成は統計の更新より時間がかかるため、**RUNSTATS** コマンドを実行してデータの現在の統計をリフレッシュしてから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。

以下の状況は、表の再編成が必要であることを示す場合があります。

- 照会でアクセスする表に対して、大量の挿入、更新、および削除アクティビティが発生した。
- クラスタ率の高い索引を使用する照会のパフォーマンスに、顕著な変化があった。
- **RUNSTATS** コマンドを実行して表の統計をリフレッシュしてもパフォーマンスが向上しない。
- **REORGCHK** コマンドの出力が、表を再編成する必要があることを示している。

注: DB2 V9.7 フィックスパック 1 以降のリリースを使用すると、特定のデータ・パーティションのデータを再編成することによって、(システム生成の XML パス索引以外の) パーティション索引のみが含まれるデータ・パーティション表のデータ可用性を高くすることができます。パーティション・レベルの再編成によって、指定されたデータ・パーティションでの表の再編成が実行されます。その間、表のその他のデータ・パーティションにはアクセスできます。パーティション表に対する **REORGCHK** コマンドの出力には、パーティション・レベルの再編成の実行に関する統計と推奨値が含まれます。

REORG TABLE コマンドと **REORG INDEXES ALL** コマンドをデータ・パーティション表で実行すると、さまざまなデータ・パーティションを同時に再編成したり、1 つのパーティション上にある複数のパーティション索引を再編成したりできます。幾つかのデータ・パーティションを同時に再編成したり、1 つのパーティションで複数のパーティション索引を再編成したりする場合、ユーザーは無関係のパーティショ

ンにはアクセスできますが、影響を受けるパーティションにはアクセスできません。同じ表で同時に作動する REORG コマンドを発行するには、以下のすべての基準を満たしている必要があります。

- 各 REORG コマンドの **ON DATA PARTITION** 節では、異なるパーティションを指定する必要があります。
- 各 REORG コマンドでは、**ALLOW NO ACCESS** モードを使用してデータ・パーティションへのアクセスを制限しなければなりません。
- **REORG TABLE** コマンドを発行する場合、パーティション表にある索引はパーティション索引のみでなければなりません。この表には、非パーティション索引（システム生成の XML パス索引を除く）を定義することはできません。

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます：表の再編成。タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

表を再編成する方式の選択

表再編成には、従来の再編成（オフライン）とインプレース再編成（オンライン）の2つの方法があります。

オフライン再編成がデフォルトの動作です。オンライン再編成操作を指定するには、**REORG TABLE** コマンドで **INPLACE** オプションを使用します。

オンライン表移動のストアード・プロシージャを使用してインプレース再編成を行うという代わりの方法も使用可能です。『**ADMIN_MOVE_TABLE** プロシージャの使用によるオンラインの表の移動』を参照してください。

それぞれの方法には、以下のセクションに要約されているように利点と欠点があります。再編成方式を選択する際、どちらの方法がユーザーの優先順位に適合する利点を備えているか考慮してください。例えば、パフォーマンスよりも、失敗したときにリカバリーできることのほうが重要な場合は、オンライン再編成の方式が適している可能性があります。

オフライン再編成の利点

この方法では以下の利点があります。

- 最も高速な表再編成操作。特に、ラージ・オブジェクト (LOB) またはロング・フィールドのデータが含まれていない場合に当てはまります。
- 完了時に表と索引が完全にクラスタ化されている。
- 表の再編成後に索引が自動的に再作成される。索引の再作成のための別個のステップはありません。
- シャドー・コピーを作成するために **TEMPORARY** 表スペースを使用する。これにより、ターゲット表または索引が含まれる表スペースに関するスペース要件が軽減されます。
- データを再クラスタ化するために、クラスタリング索引以外の索引を使用する。

オフライン再編成の欠点

この方法には、以下の特徴があります。

- 表アクセスが限られている。REORG 操作のソートと作成フェーズで可能なのは読み取りアクセスのみです。
- 再編成されている表のシャドー・コピーに関するスペース要件が大きい。
- REORG プロセスを十分に制御できない。オフライン REORG 操作を一時停止して再始動することはできません。

オンライン再編成の利点

この方法では以下の利点があります。

- すべての表アクセスが可能。REORG 操作の切り捨てフェーズ時は例外です。
- REORG プロセスを十分に制御できる。バックグラウンドで非同期的に実行したり、一時停止、再開、または停止したりすることが可能です。例えば、表に対して更新操作または削除操作が大量に実行されている場合に進行中の REORG 操作を一時停止できます。
- 失敗時にはプロセスをリカバリーできる。
- 表が徐々に処理されるので、作業用ストレージに対する要件が軽減する。
- REORG 操作が完了する前でさえ、再編成の利点をすぐに活かせる。

オンライン再編成の欠点

この方法には、以下の特徴があります。

- REORG 操作時に表にアクセスするトランザクション・タイプによっては、データまたは索引のクラスター化が不完全になる。
- オフライン REORG 操作に比べてパフォーマンスが低い。
- 移動される行の数、表に定義される索引数、およびそうした索引のサイズによっては、ロギング要件が高くなる可能性がある。
- 索引は保持されているが再作成されていないので、後に索引再編成を行う必要性が生じる可能性がある。

表 6. オンライン再編成とオフライン再編成の比較

特性	オフライン再編成	オンライン再編成
パフォーマンス	高速	低速
完了時のデータのクラスター係数	良好	完全にはクラスター化されない
並行性 (表へのアクセス)	アクセスなしから読み取り専用の範囲	読み取り専用からフル・アクセスの範囲
データ・ストレージ・スペース所要量	大きい	大きくない
ロギング・ストレージ・スペース所要量	大きくない	大きくなることもある
ユーザー制御 (処理を一時停止/再開できるかどうか)	あまりよく制御できない	よく制御できる
リカバリー可能性	リカバリー不能	リカバリー可能

表 6. オンライン再編成とオフライン再編成の比較 (続き)

特性	オフライン再編成	オンライン再編成
索引再ビルド	行われる	行われない
すべてのタイプの表のサポート	はい	いいえ
クラスタリング索引以外の索引を指定する機能	はい	いいえ
TEMPORARY 表スペースの使用	はい	いいえ

表 7. オンライン再編成およびオフライン再編成に関してサポートされる表タイプ

表タイプ	オフライン再編成のサポート	オンライン再編成のサポート
マルチディメンション・クラスタリング表 (MDC)	あり ¹	いいえ
挿入時クラスタリング表 (ITC)	あり ¹	いいえ
範囲がクラスタ化された表 (RCT)	なし ²	いいえ
付加モードの表	いいえ	なし ³
ロング・フィールドまたはラージ・オブジェクト (LOB) データのある表	あり ⁴	いいえ
システム・カタログ表: SYSIBM.SYSDBAUTH, SYSIBM.SYSROUTINEAUTH, SYSIBM.SYSSEQUENCES, SYSIBM.SYSTABLES	はい	いいえ
注:		
<ol style="list-style-type: none"> 1. クラスタ化は MDC ブロック索引経由で自動的に維持されるため、MDC 表の再編成にはスペース再利用のみが関係します。指定できる索引はありません。同様に、ITC 表では、クラスタリング索引を使用して再編成を指定することはできません。 2. RCT の範囲域は常にクラスタ化されたままになります。 3. オンライン再編成は、付加モードが使用不可になってから実行できます。 4. ロング・フィールドまたはラージ・オブジェクト (LOB) データの再編成にはかなりの時間がかかる可能性があり、照会パフォーマンスが向上することはありません。スペース再利用のためにのみ行う必要があります。 		

表再編成の進行状況のモニター

表 REORG 操作の現在の進行状況は、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、再編成の対象となっている表が格納されているデータベースに対して **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 REORG 操作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・システム・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 メッセージが履歴ファイルに書き込まれます。インプレース表 REORG 操作の場合、状況は「PAUSED (一時停止)」と記録されます。

従来の (オフライン) 表再編成

従来の表再編成では、シャドー・コピー方式を使用して、再編成される表の完全なコピーを作成します。

従来のオフライン表再編成操作には、4 つのフェーズがあります。

1. ソート - このフェーズでは、索引が **REORG TABLE** コマンドで指定された場合、またはクラスタリング索引が表で定義された場合、表の行は最初にその索引に従ってソートされます。 **INDEXSCAN** オプションが指定された場合、索引スキャンが表をソートするのに使用されます。それ以外の場合、表スキャン・ソートが使用されます。このフェーズはクラスタリング表の **REORG** 操作にのみ適用されます。スペース再利用 **REORG** 操作は、ビルド・フェーズから始まります。
2. ビルド - このフェーズでは、再編成された表全体のコピーが、その表スペースの中か、**REORG TABLE** コマンドで指定された **TEMPORARY** 表スペース内のいずれかに作成されます。
3. 置換 - このフェーズでは、**TEMPORARY** 表スペースからのコピーによって元の表オブジェクトが置換されるか、再編成されている表の表スペース内に新しく作成されたオブジェクトへのポインターが作成されます。
4. 索引すべての再作成 - このフェーズでは、表に定義されたすべての索引が再作成されます。

スナップショット・モニターまたはスナップショット管理ビューを使用して、表の **REORG** 操作の進行をモニターし、現在のフェーズを識別します。

オンライン・モードよりもオフライン・モードの方が、ロック状態による制限が厳しくなります。コピーの作成中に、表への読み取りアクセスは可能です。しかし、元の表を再編成したコピーに置き換える際、または索引を再作成する際は、表の排他的アクセスが必要となります。

表の **REORG** 処理の間は、終始 **IX** 表スペース・ロックが必要です。ビルド・フェーズでは、**U** ロックが取得され、これが表に対して保持されます。**U** ロックでは、ロック所有者は表のデータを更新できます。他のアプリケーションがデータを更新することはできませんが、読み取りアクセスは許可されます。置換フェーズが開始すると、**U** ロックは **Z** ロックにアップグレードされます。このフェーズ中は、他のアプリケーションはデータにアクセスできません。このロックは、表の **REORG** 操作が完了するまで保持されます。

オフライン再編成処理によって、いくつかのファイルが作成されます。これらのファイルは、データベース・ディレクトリーに格納されます。これらのファイル名には、表スペースとオブジェクト ID が接頭部に付きます。例えば、0030002.ROR は表スペース ID が 3 で表 ID が 2 である、表の **REORG** 操作の状態ファイルです。

以下のリストには、オフライン表 **REORG** 操作の際にシステム管理スペース (SMS) 表スペースに作成される一時ファイルが示されています。

- .DTR - データ・シャドー・コピー・ファイル

- .LFR - ロング・フィールド・ファイル
- .LAR - ロング・フィールド割り振りファイル
- .RLB - LOB データ・ファイル
- .RBA - LOB 割り振りファイル
- .BMR - マルチディメンション・クラスタリング (MDC) 表および挿入時クラスタリング (ITC) 表のブロック・オブジェクト・ファイル

以下は、索引 REORG 操作の際に作成される一時ファイルです。

- .IN1 - シャドー・コピー・ファイル

以下のリストには、ソート・フェーズの際に、システム TEMPORARY 表スペースに作成される一時ファイルが示されています。

- .TDA - データ・ファイル
- .TIX - 索引ファイル
- .TLF - ロング・フィールド・ファイル
- .TLA - ロング・フィールド割り振りファイル
- .TLB - LOB ファイル
- .TBA - LOB 割り振りファイル
- .TBM - ブロック・オブジェクト・ファイル

再編成処理に関連するファイルは、システムから手動で除去しないでください。

オフラインで表を再編成する:

オフラインでの表の再編成は、表をデフラグする最も速い方法です。再編成によって表のスペース所要量は削減され、データ・アクセスおよび照会パフォーマンスが改善されます。

始める前に

再編成する表に対する、SYSADM 権限、SYSCTRL 権限、SYSMAINT 権限、DBADM 権限、または SQLADM 権限か、CONTROL 特権が必要です。表を再編成するにはデータベース接続も必要です。

このタスクについて

再編成を必要とする表を特定したなら、それらの表 (およびオプションで、それらの表で定義された索引) に対して、REORG ユーティリティを実行できます。

手順

1. **REORG TABLE** コマンドを使用して表を再編成するには、単に表の名前を指定します。例えば、

```
reorg table employee
```

特定の TEMPORARY 表スペースを使用して、表を再編成できます。例えば、

```
reorg table employee use mytemp
```

表を再編成し、特定の索引にしたがって行を再配列することができます。例えば、

```
reorg table employee index myindex
```

2. SQL CALL ステートメントを使用して表を再編成するには、ADMIN_CMD プロシージャを使用して **REORG TABLE** コマンドを指定します。例えば、

```
call sysproc.admin_cmd ('reorg table employee')
```

3. 管理アプリケーション・プログラミング・インターフェースを使用して表を再編成するには、db2Reorg API を呼び出します。

次のタスク

表の再編成後、その表の統計を収集します。こうして、オプティマイザーによって照会アクセス・プランの評価で使用されるデータは、最も正確なものになります。

オフライン表再編成のリカバリー:

オフライン表再編成は、置換フェーズが始まるまでは、「全か無か」の処理です。ソートまたはビルド・フェーズでシステムが破損した場合、REORG 操作はロールバックされ、クラッシュ・リカバリーの際に再実行されません。

置換フェーズの開始後にシステムが破損する場合、REORG 操作を完了する必要があります。これは、すべての再編成作業がすでに行われ、元の表が使用できない状態になっている可能性があるためです。クラッシュ・リカバリー中に再編成された表オブジェクトの一時ファイルが必要になりますが、ソートに使用した TEMPORARY 表スペースは必要ありません。リカバリーでは置換フェーズが初めから再開され、リカバリーにはコピー・オブジェクト内のすべてのデータが必要となります。この場合、システム管理スペース (SMS) 表スペースとデータベース管理スペース (DMS) 表スペースの間に相違点があります。再編成が同じ表スペース内で行われた場合、SMS では再編成された表オブジェクトを 1 つのオブジェクトから他のオブジェクトにコピーする必要がありますが、DMS では再編成された表オブジェクトをポイントするだけで、元の表がドロップされます。索引は再作成されませんが、クラッシュ・リカバリー中に無効とマークされ、データベースは標準の規則に従って、索引をいつ再作成するか (データベースを再開するときか、最初の索引アクセスのときか) を決定します。

索引の再作成フェーズで破損する場合、新しい表オブジェクトが既に存在するので再実行は行われません。索引は、前述のように処理されます。

ロールフォワード・リカバリー時には、古いバージョンの表がディスク上にある場合、REORG 操作が再実行されます。ロールフォワード・ユーティリティでは、作成フェーズ中にログに記録されたレコード ID (RID) を使用して作成フェーズと置換フェーズが繰り返され、再編成された表を作成した操作が再適用されます。索引は、前述のように処理されます。もともと TEMPORARY 表スペースが使用されていた場合に限り、再編成されたオブジェクトのコピー用の TEMPORARY 表スペースが必要となります。ロールフォワード・リカバリー中に、複数の REORG 操作が同時に再実行されることがあります (並列リカバリー)。

オフライン表再編成のパフォーマンスの改善:

オフライン表再編成のパフォーマンスは、データベース環境の特性によって大きく左右されます。

REORG 操作を ALLOW NO ACCESS モードで実行しても、ALLOW READ ACCESS モードで実行しても、パフォーマンスにはほとんど違いはありません。違いは、ALLOW READ ACCESS モードの REORG 操作時に、このユーティリティーは他のアプリケーションがスキャンを完了し、ロックを解放するまで待たなければ、表を置換できないという点です。どちらのモードで実行されている場合でも、REORG 操作の索引再ビルド・フェーズ中は、表は使用不可になります。

パフォーマンス向上のためのヒント

- 十分なスペースが存在する場合は、TEMPORARY 表スペースは使わずに、元の表と再編成された表のコピー両方に同じ表スペースを使います。これによって、再編成された表を TEMPORARY 表スペースからコピーするのに要する時間を節約できます。
- REORG 操作中に保守する必要がある索引を減らすため、表を再編成する前に不要な索引をドロップすることを考慮します。
- 再編成された表が常駐する表スペースのプリフェッチ・サイズが正しく設定されていることを確認します。
- データベース構成パラメーター **sortheap** および **sheapthres** を調整して、ソートに使用できるスペースを制御します。それぞれのプロセッサは専用ソートを実行するため、**sheapthres** の値は少なくとも **sortheap x number-of-processors** でなければなりません。
- ページ・クリーナーの数を調整して、バッファ・プール内のダーティ索引ページが可能な限りすぐに除去されるようにしてください。

インプレース (オンライン) 表再編成

インプレース表再編成を使用すると、表のデータにフル・アクセスしながらその表を再編成できます。このようにデータに中断なくアクセスすると、表 REORG 操作の速度が低下します。

制約事項: インプレース (オンライン) 再編成は、DB2 pureScale環境ではサポートされていません。DB2 pureScale環境でインプレース再編成を実行しようとする、失敗して SQL1419N を出します。

インプレースまたはオンライン表 REORG 操作の際、表の各部分が順次再編成されていきます。データは TEMPORARY 表スペースにコピーされません。代わりに行が既存の表オブジェクト内で移動してクラスター化が再確立され、フリー・スペースが再利用され、オーバーフロー行が除去されます。

オンライン表 REORG 操作には、以下の 4 つの主要なフェーズがあります。

1. n ページを選択する

このフェーズでは、データベース・マネージャーが n ページの範囲を選択します。 n は REORG 処理を行う連続するページのエクステンツ・サイズで、最小は 32 です。

2. 範囲を空ける

REORG ユーティリティは、この範囲内のすべての行を、表内のフリー・ページに移動します。移動する各行は、行の新しいロケーションのレコード ID (RID) を含む REORG 表ポインター (RP) レコードを残します。行は、そのデータを含む REORG 表オーバーフロー (RO) レコードとして表の空きページに配置されます。このユーティリティが行の集合の移動を完了すると、表内のデータにアクセスしているすべてのアプリケーションが完了するまで待機します。こうした『古いスキャナー』は、表データにアクセスする際に古い RID を使用します。この待機期間中に開始されるすべての表アクセス (『新しいスキャナー』) は、データにアクセスするために新しい RID を使用します。すべての古いスキャナーが完了すると、REORG ユーティリティは RP レコードを削除し、RO レコードを標準レコードに変換することによって、移動した行をクリーンアップします。

3. 範囲を埋める

特定の範囲内にあるすべての行が空くと、行が再編成された形式 (使用されている索引に従ってソートされ、定義されている PCTFREE 制限に従った形式) で再び書き込まれます。範囲内のすべてのページが再び書き込まれると、次の連続した n ページが表で選択され、再び処理が始まります。

4. 表を切り捨てる

デフォルトでは、表のすべてのページが再編成されると、スペースの再利用のために表が切り捨てられます。NOTTRUNCATE オプションが指定されている場合は、再編成された表は切り捨てられません。

オンライン表 REORG 操作中に作成されるファイル

オンライン表 REORG 操作中に、データベース・パーティションごとに .OLR 状態ファイルが作成されます。このバイナリー・ファイルには、xxxxyyyy.OLR という形式の名前があります。xxxx は表スペース ID、yyyy はオブジェクト ID です (それぞれ 16 進数形式)。このファイルには、一時停止状態のオンライン REORG 操作を再開するために必要な以下の情報が含まれています。

- REORG 操作のタイプ
- 再編成されている表のライフ・ログ・シーケンス番号 (LSN)
- 次に空になる範囲
- REORG 操作がデータをクラスター化しているか、それとも単にスペースを再利用しているだけか
- データをクラスター化するのに使用されている索引の ID

.OLR ファイルに関してはチェックサムが実行されます。ファイルが破損してチェックサム・エラーが発生した場合や表 LSN がライフ LSN と一致しなかった場合は、新しい REORG 操作が開始され、新しい状態ファイルが作成されます。

.OLR 状態ファイルが削除された場合は、REORG 処理を再開できず、SQL2219N が戻されて、新しい REORG 操作を開始する必要があります。

再編成処理に関連するファイルは、システムから手動で除去しないでください。

オンラインで表を再編成する:

オンラインまたはインプレース表再編成では、表の再編成中に表にアクセスすることができます。

始める前に

再編成する表に対する、SYSADM 権限、SYSCTRL 権限、SYSMAINT 権限、DBADM 権限、または SQLADM 権限か、CONTROL 特権が必要です。表を再編成するにはデータベース接続も必要です。

このタスクについて

再編成を必要とする表を特定したなら、それらの表 (およびオプションで、それらの表で定義された索引) に対して、REORG ユーティリティーを実行できます。

手順

- **REORG TABLE** コマンドを使用してオンラインで表を再編成するには、表の名前および **INPLACE** パラメーターを指定します。例えば、

```
reorg table employee inplace
```
- **SQL CALL** ステートメントを使用してオンラインで表を再編成するには、**ADMIN_CMD** プロシージャを使用して **REORG TABLE** コマンドを指定します。例えば、

```
call sysproc.admin_cmd ('reorg table employee inplace')
```
- 管理アプリケーション・プログラミング・インターフェースを使用してオンラインで表を再編成するには、db2Reorg API を呼び出します。

次のタスク

表の再編成後、その表の統計を収集します。こうして、オプティマイザーによって照会アクセス・プランの評価で使用されるデータは、最も正確なものになります。

オンライン表再編成のリカバリー:

オンライン表再編成の失敗は、ディスク満杯やロギング・エラーといった処理エラーによるものが多数です。オンライン表再編成が失敗した場合、SQLCA メッセージが履歴ファイルに書き込まれます。

実行時に障害が発生すると、オンライン表 REORG 操作が一時停止し、クラッシュ・リカバリー時にロールバックします。その後、**REORG TABLE** コマンドで **RESUME** パラメーターを指定すると、REORG 操作を再開できます。このプロセス全体がログに記録されるので、オンライン表再編成は確実にリカバリー可能になります。

状況によっては、**num_log_span** データベース構成パラメーターの値によって設定されている限度を、オンライン表 REORG 操作が超える場合があります。その場合、データベース・マネージャーは REORG ユーティリティーを強制的に PAUSE 状態にします。スナップショット・モニター出力では、REORG ユーティリティーの状態は「PAUSED」として示されます。

オンライン表 REORG 一時停止は、割り込みによるものです。つまり、ユーザーによって (**REORG TABLE** コマンドの **PAUSE** パラメーター、または **FORCE APPLICATION**

コマンドを使用して) 起動されることもあれば、状況によっては (例えば、システムが破損した場合) データベース・マネージャーによって起動されることもあります。

パーティション・データベース環境の 1 つ以上のデータベース・パーティションにエラーが発生した場合、戻される **SQLCODE** はエラーを報告した最初のデータベース・パーティションからのものになります。

オンライン表再編成の一時停止と再開:

進行中のオンライン表再編成は、一時停止して、再開することができます。

始める前に

オンライン再編成を一時停止または再開する表に対する、**SYSADM** 権限、**SYSCTRL** 権限、**SYSMAINT** 権限、**DBADM** 権限、または **SQLADM** 権限か、**CONTROL** 特権が必要です。オンライン表再編成を一時停止または再開するには、データベース接続も必要です。

手順

1. **REORG TABLE** コマンドを使用してオンライン表再編成を一時停止するには、表の名前、**INPLACE** パラメーター、および **PAUSE** パラメーターを指定します。例えば、

```
reorg table employee inplace pause
```

2. 一時停止したオンライン表再編成を再開するには、**RESUME** パラメーターを指定します。例えば、

```
reorg table employee inplace resume
```

オンラインでの表の **REORG** 操作が一時停止されると、その表の新規の再編成は開始できません。新規の再編成処理を開始する前に、一時停止した操作を再開または停止しなければなりません。

RESUME 要求に続いて、再編成プロセスでは、現在の **RESUME** 要求で指定された切り捨てオプションが常に優先されます。例えば、**NOTRUNCATE** パラメーターが現在の **RESUME** 要求で指定されていない場合、元の **REORG TABLE** コマンド (または以前の **RESUME** 要求) で指定された **NOTRUNCATE** パラメーターは無視されません。

リストア操作およびロールフォワード操作後に、表の **REORG** 操作を再開することはできません。

オンライン表再編成のロックおよび並行性に関する考慮事項:

オンライン表再編成における最も重要な面の一つは、ロックの制御方法です。これは、アプリケーション並列性に関して重要なことであるためです。

オンライン表 **REORG** 操作は、以下のロックを保持できます。

- 表スペースへの書き込みアクセスを確実にするために、**REORG** 操作の影響を受ける表スペースに対する **IX** ロックが取得されます。

- 表ロックが取得され、REORG のすべての操作の間保持されます。ロックのレベルは、再編成中に有効なアクセス・モードによって決まります。
 - ALLOW WRITE ACCESS が指定された場合、IS 表ロックが取得されます。
 - ALLOW READ ACCESS が指定された場合、S 表ロックが取得されます。
- 切り捨てフェーズでは、表の S ロックが要求されます。S ロックが取得されるまでは、並行トランザクションによる行の挿入が可能です。こうした挿入行は REORG ユーティリティから参照できず、表が切り捨てられない可能性があります。S 表ロックが取得されると、切り捨てられなかった表の行が移動され、表が圧縮されます。表が圧縮された後、切り捨てポイントが判別された時点で表にアクセスしていたすべてのトランザクションが完了すると初めて、切り捨てが行われます。
- 以下のように、表ロックのタイプによっては、行ロックが取得されることがあります。
 - 表に対する S ロックが保持されている場合、個々の行レベルの S ロックは不要なので、ロックはそれ以上必要ありません。
 - 表に対する IS ロックが保持されている場合は、行が移動される前に NS 行ロックが取得され、移動完了後に解放されます。
- オンライン表 REORG 操作時に、特定の内部ロックが取得される場合もあります。

ロックはオンライン表 REORG 操作および同時ユーザー・アプリケーションの両方のパフォーマンスに影響を与えます。オンライン表再編成時に生じるロック・アクティビティについて理解するために、ロック・スナップショット・データを使用できます。

表再編成のモニター

GET SNAPSHOT コマンド、**SNAPTAB_REORG** 管理ビュー、または **SNAP_GET_TAB_REORG** 表関数を使用して、表再編成操作の状況に関する情報を得ることができます。

手順

- SQL を使用して再編成操作に関する情報にアクセスするには、**SNAPTAB_REORG** 管理ビューを使用します。例えば、以下の照会により、現在接続中のデータベースの全データベース・パーティションにおける表再編成操作に関する詳細が返されます。再編成された表がまったくない場合、行は何も戻されません。

```
select
  substr(tabname, 1, 15) as tab_name,
  substr(tabschema, 1, 15) as tab_schema,
  reorg_phase,
  substr(reorg_type, 1, 20) as reorg_type,
  reorg_status,
  reorg_completion,
  dbpartitionnum
from sysibmadm.snaptab_reorg
order by dbpartitionnum
```

- スナップショット・モニターを使用して再編成操作に関する情報にアクセスするには、**GET SNAPSHOT FOR TABLES** コマンドを発行して、表再編成モニター・エレメントの値を調べます。

タスクの結果

オフラインでの表の REORG 操作は同期であるため、エラーはユーティリティの呼び出し元 (アプリケーションまたはコマンド行プロセッサ) に返されます。オンラインでの表の REORG 操作は非同期であるため、この場合のエラー・メッセージは CLP に返されません。オンラインでの表の REORG 操作中に返される SQL エラー・メッセージを表示するには、**LIST HISTORY REORG** コマンドを使用します。

オンラインでの表の REORG 操作は、db2Reorg 処理としてバックグラウンドで実行されます。この処理は、呼び出し元アプリケーションがデータベース接続を終了しても、引き続き実行されます。

索引の再編成

表が更新されるにつれて、索引パフォーマンスが低下する場合があります。

この低下は次のようにして発生する可能性があります。

- リーフ・ページがフラグメント化される。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なくなり、索引の密度が低くなる。リーフ・ページの密度が低いと、順次プリフェッチの効率が低下し、入出力待機数が増加します。ただし、スマート索引プリフェッチが使用可能になっていると、密度の低い索引が存在する場合、照会オプティマイザーは先読みプリフェッチに切り替えます。これにより、密度の低い索引がパフォーマンスに与える悪い影響が小さくなります。
- 索引のレベル数が増えすぎる。この場合、索引を再編成する必要があります。

索引の再編成では、次のものがが必要です。

- SYSADM、SYSMAINT、SYSCTRL、DBADM、または SQLADM 権限か、表およびその索引に対する CONTROL 特権
- REBUILD オプションを ALLOW READ オプションまたは WRITE ACCESS オプションとともに選択する場合は、索引を格納できる量のフリー・スペースが表スペース内に必要です。このスペースは、現行の索引サイズと等しくなければなりません。CREATE TABLE ステートメントを発行する際は、索引を LARGE 表スペースに置くことを考慮してください。
- 追加ログ・スペース。索引 reorg ユーティリティは、そのアクティビティをログに記録します。

CREATE INDEX ステートメントに MINPCTUSED オプションを指定した場合、キーが削除され、かつフリー・スペースが指定値より少なくなると、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。

索引クラスタリングを復元し、スペースを解放し、さらにリーフ・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。
- オフラインで表を再編成して索引を再作成することができるオプションを指定した **REORG TABLE** コマンドを使用します。

- REBUILD オプションを指定した **REORG INDEXES** コマンドを使用して、オンラインまたはオフラインで索引を再編成します。オンライン再編成は、実稼働環境で選択することもできます。なぜなら、この方法では、索引の再作成中にユーザーが表の読み取り/書き込みを行えるからです。

スペースを解放することが主な目的である場合は、**REORG** コマンドの **CLEANUP** オプションおよび **RECLAIM EXTENTS** オプションを使用することを検討してください。詳細については、関連リンクを参照してください。

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: 索引の再編成。タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

DB2 V9.7 フィックスパック 1 以降のリリースの場合、**REORG INDEXES ALL** コマンドをデータ・パーティション表で使用し、**ON DATA PARTITION** 節にパーティションを指定すると、単一のデータ・パーティションのパーティション索引が再編成されます。索引の再編成の際、影響を受けないパーティションは引き続き読み取れますが、書き込み可能なアクセスは影響を受けるパーティションにのみ限定されます。

REORG TABLE コマンドと **REORG INDEXES ALL** コマンドをデータ・パーティション表で実行すると、さまざまなデータ・パーティションを同時に再編成したり、1 つのパーティション上にある複数のパーティション索引を再編成したりできます。いくつかのデータ・パーティションを同時に再編成したり、1 つのパーティションで複数のパーティション索引を再編成したりする場合、ユーザーは影響を受けないパーティションにアクセスできます。同じ表で同時に作動する **REORG** コマンドを発行するには、以下のすべての基準を満たしている必要があります。

- 各 **REORG** コマンドの **ON DATA PARTITION** 節では、異なるパーティションを指定する必要があります。
- 各 **REORG** コマンドでは、**ALLOW NO ACCESS** モードを使用してデータ・パーティションへのアクセスを制限しなければなりません。
- **REORG TABLE** コマンドを発行する場合、パーティション表にある索引はパーティション索引のみでなければなりません。この表には、非パーティション索引 (システム生成の XML パス索引を除く) を定義することはできません。

注: **REORGCHK** コマンドからの出力には、索引の再編成の統計と推奨値が含まれます。パーティション表の場合、この出力にはパーティション索引と非パーティション索引の再編成の統計と推奨値が含まれています。また、スペースの再利用が目的である場合は、**ADMIN_GET_INDEX_INFO** 関数の **RECLAIMABLE_SPACE** 出力に、再利用可能なスペース量が示されます。この再利用可能なスペースを解放するには、**RECLAIM EXTENTS** オプションを指定して **REORG INDEXES** コマンドを使用します。

オンライン索引再編成

ALLOW WRITE ACCESS オプションおよび **REBUILD** オプションを指定して **REORG INDEXES** コマンドを使用すると、指定された表への読み取り/書き込みアクセスを継続しながら、その表の索引すべてが再作成されます。再編成中、索引に影響を与え

る変更が基礎表に加えられた場合、その変更内容はログに記録されます。reorg 操作では、索引を再ビルドしながら、ログ記録された変更内容が処理されます。

索引に影響を与える変更が基礎表に加えられた場合、その変更内容は内部メモリー・バッファーにも書き込まれます (そのようなスペースを使用できる場合)。内部バッファーは、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指定のメモリー域です。メモリー・バッファー・スペースを使用すると、索引 reorg ユーティリティーは、まずメモリーから変更を直接読み取り、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、変更を処理できます。割り振られたメモリーは、reorg 操作が完了した後に解放されます。

空間インデックス、マルチディメンション・クラスタリング (MDC) 表、または挿入時クラスタリング (ITC) 表では、ALLOW WRITE ACCESS モード (CLEANUP オプションあり、もしくはなし) でのオンライン索引再編成がサポートされています。

制約事項: REBUILD オプションを指定したオンライン再編成は、DB2 pureScale 環境ではサポートされていません。DB2 pureScale 環境で REBUILD オプションを指定してオンライン再編成を実行しようとする、失敗して SQL1419N が出されます。

オンライン索引再編成のロックおよび並行性に関する考慮事項

このトピックでは、オンライン索引再編成という用語は、ALLOW READ ACCESS パラメーターまたは ALLOW WRITE ACCESS パラメーターを使用して実行する索引再編成を意味します。

これらのオプションを指定すると、索引を再編成中の表にアクセスすることができます。REBUILD オプションを指定したオンライン索引再編成の間、新しい索引が追加コピーとして作成され、元の索引はそのまま維持されます。並行トランザクションは、新しい索引が作成されている間は元の索引を使用します。再編成操作の終わりに、元の索引が新しい索引に置き換えられます。その間にコミットされたトランザクションは、元の索引の置換後に新しい索引に反映されます。再編成操作が失敗してトランザクションがロールバックされた場合は、元の索引がそのまま維持されます。CLEANUP オプションと RECLAIM EXTENTS オプションを使用したオンライン索引の再編成中に、スペースが再利用され、表スペースのすべてのオブジェクトで利用できるようになります。

オンライン索引再編成操作は、以下のロックを保持できます。

- 表スペースへのアクセスを確実にするために、再編成操作の影響を受ける表スペースに対する IX ロックが取得されます。これには、表を保持している表スペースだけでなく、パーティション、および索引オブジェクトも含まれます。
- 影響を受ける表が再編成中に変更されないようにするために、X 表変更ロックが取得されます。
- 表ロックが取得され、再編成操作の最後まで保持されます。ロックのタイプは、以下のように、表タイプ、アクセス・モード、および再編成オプションによって異なります。
 - 非パーティション表の場合:
 - ALLOW READ ACCESS を指定した場合、表に対する U ロックが取得されません。

- **ALLOW WRITE ACCESS** を指定した場合、表に対する IN ロックが取得されま
す。
- **CLEANUP** を指定すると、READ アクセスの場合は表に対する S ロックが取
得され、WRITE アクセスの場合は IX ロックが取得されます。
- パーティション表の場合 (**ALLOW READ ACCESS** または **ALLOW WRITE ACCESS** を
指定した再編成はパーティション・レベルでのみサポート):
 - **ALLOW READ ACCESS** を指定した場合、パーティションに対する U ロックが
取得されます。
 - **ALLOW WRITE ACCESS** を指定した場合、パーティションに対する IS ロック
が取得されます。
 - **CLEANUP** を指定すると、READ アクセスの場合はパーティションに対する S
ロックが取得され、WRITE アクセスの場合は IX ロックが取得されます。
 - 指定したアクセス・モードまたはオプションにかかわらず、表に対する IS
ロックが取得されます。
- 索引再編成の終わりに、表またはパーティションに対する排他的 Z ロックが要求
されます。パーティション表に非パーティション索引が含まれる場合は、表とパ
ーティションに対する Z ロックが取得されます。このロックによって表およびパ
ーティションへのアクセスが中断され、元の索引を新しい索引に置き換えること
が可能になります。このロックは、再編成中にコミットされたトランザクション
が新しい索引に反映されるまで保持されます。
- システム・カタログ表 **SYSIBM.SYSTABLES** に対する IS 表ロックおよび NS
行ロックが取得されます。
- パーティション・レベルの再編成の場合は、システム・カタログ表
SYSIBM.SYSDATAPARTITIONS に対する IS 表ロックおよび NS 行ロックも取
得されます。
- オンライン索引再編成操作時に、特定の内部ロックも取得される場合がありま
す。
- オンライン索引再編成は、再編成操作が失敗した場合に並行性に影響を与えるこ
とがあります。例えば、メモリー不足、ディスク・スペース不足、またはロッ
ク・タイムアウトが原因で再編成が失敗する可能性があります。再編成トランザ
クションは、終了する前に、ある決まった更新を実行します。更新を実行するに
は、再編成は既存のトランザクションがコミットされるのを待たなければなりま
せん。この遅延によって、そのプロセス内の他のトランザクションがブロックさ
れる可能性があります。DB2 バージョン 9.7 フィックスパック 1 以降、再編成
は索引オブジェクトに対する特殊ドレーン・ロックを要求するようになりました
。再編成操作は既存のトランザクションが終了するのを待ちますが、索引オブ
ジェクトへのアクセスの新しい要求は許可されます。

索引再編成操作のモニター

このタスクについて

db2pd コマンドを使用して、データベース上の再編成操作の進行状況をモニターで
きます。

手順

db2pd コマンドに **-reorgs index** パラメーターを指定して発行します。

```
db2pd -reorgs index
```

タスクの結果

以下は、db2pd コマンドに **-reorgs index** パラメーターを使用して取得した出力例であり、2 つのパーティションがある範囲パーティション表の索引再編成の進行状況がレポートされています。

注: 最初の出力では、非パーティション索引の索引 REORG 状況がレポートされています。それに続く出力は、各パーティションのパーティション索引の索引再編成統計を報告しています。ここでは、出力ごとに 1 つのパーティションのみの索引再編成統計が報告されています。

```
Index Reorg Stats:
Retrieval Time: 02/08/2010 23:04:21
TbpaceID: -6      TableID: -32768
Schema: TEST1   TableName: BIGRPT
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:03:55   End Time: 02/08/2010 23:04:04
Total Duration: 00:00:08
Prev Index Duration: -
Cur Index Start: -
Cur Index: 0          Max Index: 2          Index ID: 0
Cur Phase: 0          ( - )      Max Phase: 0
Cur Count: 0          Max Count: 0
Total Row Count: 750000

Retrieval Time: 02/08/2010 23:04:21
TbpaceID: 2      TableID: 5
Schema: TEST1   TableName: BIGRPT
PartitionID: 0  MaxPartition: 2
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:04:04   End Time: 02/08/2010 23:04:08
Total Duration: 00:00:04
Prev Index Duration: -
Cur Index Start: -
Cur Index: 0          Max Index: 2          Index ID: 0
Cur Phase: 0          ( - )      Max Phase: 0
Cur Count: 0          Max Count: 0
Total Row Count: 375000

Retrieval Time: 02/08/2010 23:04:21
TbpaceID: 2      TableID: 6
Schema: TEST1   TableName: BIGRPT
PartitionID: 1  MaxPartition: 2
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:04:08   End Time: 02/08/2010 23:04:12
Total Duration: 00:00:04
Prev Index Duration: -
Cur Index Start: -
Cur Index: 0          Max Index: 2          Index ID: 0
Cur Phase: 0          ( - )      Max Phase: 0
Cur Count: 0          Max Count: 0
Total Row Count: 375000
```


表および索引を再編成するタイミングの決定

表データに多くの変更を加えた後、論理的には連続しているデータが、順序どおりにになっていない物理データ・ページ上に置かれる場合があります (特に、多くの更新操作がオーバーフロー・レコードを作成した場合)。このようにデータが編成された場合、データベース・マネージャーは、必要な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。

このタスクについて

表再編成は無駄なスペースを除去しながらデータをデフラグします。さらに、オーバーフロー・レコードを取り込むために行を再配列し、データ・アクセスおよび最終的には照会パフォーマンスを改善します。特定の索引に従ってデータを再配列するよう指定すると、照会によるデータへのアクセスが、最小限の読み取り操作で可能となります。

多数の変更が表のデータに加えられると、索引のパフォーマンスが低下するおそれがあります。索引リーフ・ページがフラグメント化または不適切にクラスター化され、最適なパフォーマンスを得るのに必要とされる数以上のレベルが、索引で作成されるおそれがあります。これらの問題すべてにより、余分な入出力が引き起こされ、パフォーマンスが低下します。

以下の要因のいずれかがある場合、表または索引の再編成をすると良いでしょう。

- 表が最後に再編成されて以来、大量の挿入、更新、および削除アクティビティーが表に対して実行された
- 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** コマンド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** コマンドからの出力が、表またはその索引を再編成することによりパフォーマンスを向上できることを提案する

場合によっては、**REORGCHK** ユーティリティーにより、いつでも (表の **REORG** 操作実行後でも) 表の再編成が推奨されることがあります。例えば、32 KB のページ・サイズを使用していて、平均レコード長が 15 バイト、各ページの最大レコード数が 253 である場合、各ページには使用不可能なバイト数が $32700 - (15 \times 253) = 28905$ あることとなります。この場合、ページの約 88% がフリー・スペースということになります。**REORGCHK** ユーティリティーによる推奨内容を分析し、潜在的な効果を、再編成の実行コストと照らし合わせて評価することができます。

- スペースの再利用が主要な関心事である場合は、**CLEANUP** および **RECLAIM EXTENTS** オプションを指定した **REORG** コマンドを使用できます。

ADMIN_GET_INDEX_INFO 関数および **ADMIN_GET_TAB_INFO** 関数の **RECLAIMABLE_SPACE** 出力に、再利用可能なスペース量 (キロバイト) が示されます。 **ADMIN_GET_INDEX_INFO** 関数と **ADMIN_GET_TAB_INFO** 関数を実行する前に **REORG** コマンドに **CLEANUP** オプションを指定して発行した場合、関数の出力に再

利用可能な最大スペースが示されます。この情報を使用して、いつ RECLAIM EXTENTS を指定した REORG を実行すれば、表および索引のサイズの削減に役立つのか調べてください。

REORGCHK コマンドはデータ編成に関する統計情報を戻すので、特定の表または索引で再編成が必要かどうかを判別するのに役立ちます。関心事がスペースの再利用のみである場合は、**ADMIN_GET_INDEX_INFO** 関数および **ADMIN_GET_TAB_INFO** 関数の **RECLAIMABLE_SPACE** 出力に、再利用可能なスペース量の概要が示されます。しかし、定期的または特定の時に **SYSSTAT** ビューに対して特定の照会を実行するならば、履歴が作成され、それはパフォーマンスへの影響を示唆する傾向を特定するのに役立ちます。

表または索引の再編成が必要かどうかを判別するには、**SYSSTAT** ビューを照会して、以下の統計をモニターします。

- 行のオーバーフロー

SYSSTAT.TABLES ビューの **OVERFLOW** 列を照会して、オーバーフロー値をモニターします。この値は、元のページに収まらない行の数を示しています。可変長列が原因でレコード長が拡張し、データ・ページ上の割り当てられた場所に行が収まらないほどになると、行データはオーバーフローします。列が表に追加されると、長さの変更も生じます。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に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列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。表データの再編成により、行のオーバーフローを排除できます。

- 統計のフェッチ

SYSSTAT.INDEXES カタログ・ビューにある以下の列を照会して、索引の順序で表にアクセスする場合のプリフェッチャーの効果性を確認します。これらの統計では、基礎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、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、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を発揮できないということです。これは、表スペースの **PREFETCHSIZE** 値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、
- **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・アクセスの間でフェッチされる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、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、乱れた順序で挿入が行われることにより発生します。また、値の範囲にアクセスするのに索引が使用されるなら、照会パフォーマンスに影響が及びます。

- `AVERAGE_SEQUENCE_FETCH_GAP` 列は、索引を使用して表の行をフェッチする際の、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、表はまとまりがないか、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。
- 削除済みとしてマーク付けされているがまだ削除されていないレコード ID (RID) を含む索引リーフ・ページの数

通常、RID に削除対象としてマークが付けられた時点で、それら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・ページの数を取得するには、`SYSSTAT.INDEXES` ビューの `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` 列に保管されません。

この情報を使用すると、`CLEANUP ALL` オプションを指定して `REORG INDEXES` コマンドを実行することにより再利用できるスペースを見積もることができます。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再利用するには、`CLEANUP PAGES` オプションを指定して `REORG INDEXES` コマンドを実行します。

- 索引のクラスター率およびクラスター係数の統計

一般に、高度なクラスターリングが可能なのは、1 つの表において索引の 1 つのみです。クラスター率統計は、`SYSCAT.INDEXES` カタログ・ビューの `CLUSTERRATIO` 列に保管されます。この値 (0 から 100 まで) は、索引でのデータ・クラスターリングの程度を表わします。詳細な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。これらのクラスターリング統計のいずれか 1 つだけを、`SYSCAT.INDEXES` カタログ・ビューに記録できます。`CLUSTERFACTOR` 値と `CLUSTERRATIO` 値を比較するには、`CLUSTERFACTOR` 値に 100 を乗算してパーセンテージの値を求めます。

索引専用アクセスではない索引スキャンのパフォーマンスは、索引の密度が高い方が良い可能性があります。密度が低いと、この種のスキャンでは入出力が増えます。なぜなら、データ・ページが次のアクセス時までバッファー・プール内に残る可能性は低いためです。バッファー・サイズを大きくすると、密度の低い索引のパフォーマンスが向上することがあります。また、スマート索引プリフェッチを有効にすると、密度の低い索引のパフォーマンスが向上し、索引に `REORG` コマンドを実行する必要があまりなくなります。スマート索引プリフェッチは、密度の低い索引が存在する場合は必ず順次検出プリフェッチから先読みプリフェッチに切り替えることにより、これを達成します。

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、この索引におけるデータのクラスタリングが不十分であることが判明した場合、表を再編成して、データの再クラスタリングを行うこともできます。また、スマート・データ・プリフェッチを有効にすると、クラスタ化の不十分なデータのパフォーマンスが向上し、表に **REORG** コマンドを実行する必要があまりなくなります。スマート・データ・プリフェッチは、不良な状態にクラスタ化されたデータ・ページが存在する場合は必ず順次検出プリフェッチから先読みプリフェッチに切り替えることにより、これを達成します。

- リーフ・ページの数

索引が使用するリーフ・ページの数を知るには、SYSCAT.INDEXES ビューの NLEAF 列を照会します。この数から、索引の完全なスキャンに必要な索引ページ入出力の回数が分かります。

理想としては、索引の占めるスペースをできるだけ少なくして、索引スキャンに必要な入出力回数を削減します。ランダム更新アクティビティによりページ分割が行われ、索引のサイズが大きくなる場合があります。表の **REORG** 操作中、各索引は最小のスペースで再ビルドできます。

デフォルトでは、索引の作成時に、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 オプションを指定します。索引の再編成時はいつでも、指定した 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、その余分のスペースで索引の追加挿入に対応できるため、索引の再編成の頻度が減ります。

- 空のデータ・ページの数

表の空ページの数を知るには、SYSCAT.TABLES ビューの FPAGES 列および NPAGES 列を照会してから、FPAGES の値 (使用中のページ数合計) から NPAGES の値 (行を含むページの数) を引きます。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、空ページは再利用され、表に使用されるスペースの量は削減されます。また、表スキャン中に空ページがバッファ・プール中に読み取られるので、未使用ページの再利用により、スキャンのパフォーマンスが向上します。

表にある使用中ページの総数 (FPAGES) が「NPARTITIONS *1 エクステン・サイズ」以下の場合、表の再編成は推奨されません。表がパーティション表である場合、NPARTITIONS はデータ・パーティションの個数を表します。パーティション表でない場合、この値は 1 です。パーティション・データベース環境において、 $FPAGES \leq (\text{表のデータベース・パーティション・グループにあるデータベース・パーティションの数}) * (\text{NPARTITIONS} * 1 \text{ エクステン・サイズ})$ であるなら、表の再編成は推奨されません。

表または索引の再編成前に、照会パフォーマンス低下のコストと、表または索引の再編成のコスト (処理時間、経過時間、並行性の低下) 間のトレードオフを考慮してください。

表および索引の再編成のコスト

REBUILD オプションを指定して表再編成または索引再編成を実行することによって、ある程度のオーバーヘッドが発生するため、オブジェクトの再編成を行うかどうかを決める時点でこのオーバーヘッドについて考慮する必要があります。

REBUILD オプションを指定した表および索引の再編成のコストには、以下のものが含まれます。

- ユーティリティを実行する際の処理時間
- REORG ユーティリティの実行中の並行性の低下 (ロックが原因)。
- 追加のストレージ要件。
 - オフライン表再編成を行うためには、表のシャドー・コピーを保持するためのストレージ・スペースがさらに必要になります。
 - オンラインまたはインプレース表再編成では、追加のログ・スペースが必要になります。
 - オフライン索引再編成はログ・スペースをあまり必要とせず、シャドー・コピーを必要としません。
 - オンライン索引再編成を行うためには、索引のシャドー・コピーを保持するためのログ・スペースとストレージ・スペースが追加で必要になります。

場合によっては、再編成された表は元の表よりも大きくなります。以下の状況では、再編成後に表が大きくなる可能性があります。

- 行の順序を決定するために索引が使用されているクラスタリング REORG 表操作では、表レコードが可変長の場合、再編成された表内のあるページに含まれる行数が元の表よりも少なくなることがあり、そのためにより多くのスペースが必要となる可能性があります。
- 各ページに残されたフリー・スペースの量 (PCTFREE 値によって示される) が、最後の再編成時よりも増加した場合。

オフライン表再編成のスペース所要量

オフライン再編成はシャドー・コピー方式をとるので、表のもう 1 つのコピーを収容するのに十分な追加のストレージが必要です。シャドー・コピーは元の表が常駐する表スペースか、ユーザーが指定した TEMPORARY 表スペースのいずれかに作成されます。

表スキャン・ソートが使用される場合は、ソート処理に追加の TEMPORARY 表スペース・ストレージが必要になることがあります。必要な追加スペースが再編成される表のサイズと同じ大きさになる場合もあります。クラスタリング索引がシステム管理スペース (SMS) タイプまたはユニーク・データベース管理スペース (DMS) タイプの場合、この索引の再作成ではソートは必要ありません。むしろ、この索引は新しく再編成されたデータをスキャンすることによって再作成されます。再作成されるほかの索引にはソートが必要であり、最大で再編成される表のサイズの一時スペースが必要となる可能性があります。

オフラインの表 REORG 操作では、制御ログ・レコードはほとんど生成されないのので、消費されるログ・スペース量はそれほど大きくありません。REORG ユーティリティで索引が使用されない場合、表データのログ・レコードのみが作成されま

す。索引が指定された場合、またはクラスタリング索引が表に存在する場合、レコード ID (RID) は、新しいバージョンの表に配置された順にログに記録されます。それぞれの RID ログ・レコードは最大 8000 の RID (各 RID は 4 バイトを消費) を保持します。これにより、オフラインの表 REORG 操作時にログ・スペースに関する問題が生じる恐れがあります。RID はデータベースがリカバリー可能の場合にのみログに記録されることに注意してください。

オンライン表再編成のログ・スペース所要量

オンラインの表 REORG 操作に必要なログ・スペースは、オフラインの表 REORG に必要な量より大きいのが普通です。スペース所要量は、再編成される行の数、索引の数、索引キーのサイズ、および表の最初の編成状態がどの程度悪かったかによって決まります。表に関連したログ・スペース使用量についての標準のベンチマークを設定することをお勧めします。

表の各行は、オンラインの表 REORG 操作中に 2 回移動される可能性があります。それぞれの索引ごとに、表の各行は新しいロケーションを反映するよう索引キーを更新し、古いロケーションへのすべてのアクセスが完了した後に、索引キーが再び更新されて古い RID の参照が除去されます。行が戻されたとき、索引キーへの更新が再び実行されます。このアクティビティすべてはログ対象となり、オンライン表再編成が完全にリカバリー可能にされます。それぞれの行 (1 つの索引を想定しています) に、少なくとも 2 つのデータ・ログ・レコード (それぞれに行データが含まれる) と 4 つの索引ログ・レコード (それぞれにキー・データが含まれる) が存在することになります。特にクラスタリング索引は索引ページを埋め尽くして、索引の分割やマージを引き起こす傾向がありますが、これについてもログに記録される必要があります。

オンラインの表 REORG ユーティリティは頻繁に内部 COMMIT ステートメントを実行するので、通常は多数のアクティブ・ログを保持しません。このユーティリティが S 表ロックを要求する場合には、切り捨てフェーズで例外が生じる可能性があります。このユーティリティがロックを取得できない場合は待機するので、他のトランザクションがその間にログを埋め尽くしてしまう可能性があります。

表および索引の再編成の必要性を少なくする

様々な方式を使用して、表および索引の再編成の必要性 (および関連するコスト) を少なくできます。

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、以下のようになります。

- 複数パーティション表を使用します。
- マルチディメンション・クラスタリング (MDC) 表を作成します。MDC 表のクラスタリングは、CREATE TABLE ステートメントの ORGANIZE BY DIMENSIONS 節で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 ユーティリティは MDC 表の再編成を依然として勧める場合があります。
- 挿入時クラスタリング (ITC) 表を作成します。ITC 表では、循環アクセス・パターンがある場合 (例えば、類似した時刻に挿入されたすべてのデータを削除す

る場合など)、そのスペースをシステムに解放できます。そのような場合、スペースを解放する REORG RECLAIM EXTENTS コマンドを使用して、表を再編成する必要性を少なくすることができます。

- 表で APPEND モードを有効にします。例えば新しい行の索引キー値が、常に高いキー値である場合は、表のクラスタリング属性はそのキー値を表の最後に置こうとします。この場合、APPEND モードを有効にすると、クラスタリング索引を使用するよりも良い選択となり得ます。

表の再編成の必要性をさらに少なくするには、表を作成した後に以下のタスクを行ってください。

- ロードまたは表の再編成の操作の際にフリー・スペースとして残す、各ページのパーセンテージ (PCTFREE) を指定するように表を変更します。
- PCTFREE オプションを指定してクラスタリング索引を作成します。
- 表にロードする前にデータをソートします。

これらのタスクを実行した後、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。ただし、データがさらに挿入され、表のページが満杯になると、表の最後にレコードが追加され、クラスタ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表の REORG 操作またはソートを実行し、ロード操作を行うと、その索引はデータの順序を保持しようとするので、RUNSTATS ユーティリティによって収集された CLUSTERRATIO または CLUSTERFACTOR 統計は改善されます。

注: 先読みプリフェッチを有効にすると、REORGCHK コマンドの公式 F4 で表の再編成が効果的であると示されている場合でも、表を再編成する必要はあまりなくなります。

索引の再作成の必要性を少なくする

索引の再編成を使用して索引の再作成の必要性を少なくするには、以下のようになります。

- PCTFREE オプションまたは LEVEL2 PCTFREE オプションを指定して索引を作成します。
- MINPCTUSED オプションを使って索引を作成します。代わりに REORG INDEXES コマンドの CLEANUP ALL オプションを使用して、リーフ・ページをマージすることも考慮してください。
- REORG INDEXES コマンドで RECLAIM EXTENTS オプションを使用して、オンライン方式でスペースを表スペースに解放します。この操作により、索引全体を再作成することなくスペースを再利用できるようになります。

注: 先読みプリフェッチを有効にすると、REORGCHK コマンドの公式 F4 で索引の再編成が効果的であると示されている場合でも、索引の再編成を使用して索引を再作成する必要はあまりなくなります。

自動再編成

表データに多くの変更を加えた後、表およびその索引をフラグメント化できるようになります。論理的には連続しているデータは、連続していないページ上にある可能性があります。その場合、データベース・マネージャーは、データにアクセスするために余分の読み取り操作を実行することが必要になります。

RUNSTATS ユーティリティによって収集される統計情報には、表内のデータの分布が表示されます。それらの統計情報を分析すると、再編成が必要になるタイミングや必要な再編成の種類が示されることがあります。

自動再編成プロセスでは、**REORGCHK** ユーティリティの一部である公式を使用して、表または索引の再編成の必要性を判別します。再編成が必要かどうかを調べるため、統計情報の更新の原因となった表および索引が定期的に評価され、必要な場合には再編成などの操作をスケジュールします。

自動再編成機能を有効または無効にするには、**auto_reorg**、**auto_tbl_maint**、および **auto_maint** データベース構成パラメーターを使用します。

パーティション・データベース環境の場合、自動再編成の開始はカタログ・データベース・パーティション上で行われ、こうした構成パラメーターを有効にする必要があるのはそのパーティション上のみです。ただし、**REORG** 操作は、ターゲット表の存在しているすべてのデータベース・パーティション上で実行されます。

表および索引を再編成するタイミングと方法がわからない場合は、自動再編成をデータベース保守プランの一部に含めることができます。

マルチディメンション・クラスタリング (MDC) 表および挿入時クラスタリング (ITC) 表を再編成して、スペースを再利用することもできます。MDC 表および ITC 表からのエクステントの解放は、表が DMS 表スペースに入っている場合のみサポートされます。MDC 表および ITC 表からのエクステントの解放は、**REORG** コマンドの **RECLAIM EXTENTS** オプションを使用してオンライン方式で行うことができます。

また、索引の自動スペース再利用をスケジュールすることもできます。**REORG** コマンドで **RECLAIM EXTENTS** オプションを使用すると、スペースがオンライン方式で表スペースに解放されます。この操作により、索引全体を再作成することなくスペースを再利用できるようになります。**REORG** コマンドの **REBUILD** オプションも、スペースを再利用します。

データ・パーティション表での自動再編成

DB2 バージョン 9.7 フィックスパック 1 以前のリリースでは、自動再編成によって、表全体でデータ・パーティション表の再編成がサポートされています。DB2 V9.7 フィックスパック 1 以降のリリースでは、自動再編成によって、パーティション表の複数のデータ・パーティションの再編成と、パーティション表の 1 つのデータ・パーティションにおけるパーティション索引の再編成がサポートされています。

データ・パーティション表全体が ALLOW NO ACCESS モードにならないようにするため、再編成が必要なパーティション索引のデータ・パーティション・レベルにおいて **REORG INDEXES ALL** 操作が自動再編成によって実行されます。自動再編成では、再編成が必要な非パーティション索引において **REORG INDEX** 操作が実行されます。

自動再編成では、データ・パーティション表において以下の **REORG TABLE** 操作が実行されます。

- 非パーティション索引 (システム生成の XML パス索引以外) が表に定義されており、再編成が必要なパーティションが 1 つだけの場合、**ON DATA PARTITION** 節を指定した **REORG TABLE** 操作が自動再編成によって実行され、再編成が必要なパーティションが指定されます。そうでない場合は、自動再編成によって、**ON DATA PARTITION** 節を使用せずに表全体で **REORG TABLE** 操作が実行されます。
- 非パーティション索引 (システム生成の XML パス索引以外) が表に定義されていない場合、再編成が必要な各パーティションに対して、**ON DATA PARTITION** 節を指定した **REORG TABLE** 操作が自動再編成によって実行されます。

VOLATILE 表での自動再編成

VOLATILE 表に対する自動索引再編成を使用可能にできます。自動再編成処理は、VOLATILE 表で索引の再編成が必要かどうかを判別し、REORG INDEX CLEANUP をスケジューリングします。索引再編成は、VOLATILE 表において定期的に行われ、これらの表で定義されている索引が再使用できるスペースを解放します。

VOLATILE 表は頻りに更新されるので、この表で統計を収集することはできません。再編成する必要がある索引を判別するために、自動再編成では統計の代わりに numInxPseudoEmptyPagesForVolatile 属性が使用されます。AUTO_REORG ポリシー内のこの属性は、疑似削除キーを含んだ空の索引ページが索引に何個存在するようになったら索引再編成を起動するかを示します。

VOLATILE 表内での自動索引再編成を有効にするには、**DB2_WORKLOAD** レジストリ変数を SAP に設定し、自動再編成を有効にし、numInxPseudoEmptyPagesForVolatile 属性を設定する必要があります。

表および索引の自動再編成を有効にする

表および索引の自動再編成を使用すると、データを再編成するタイミングおよび方法について気にせずすみませす。

このタスクについて

よく編成された表および索引データを得ることは、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。挿入、更新、および削除操作を数多く行うと、論理上は連続する表データが、不連続のデータ・ページに存在するようになることがあります。そうすると、データベース・マネージャは、読み取り操作を余分に実行してデータにアクセスしなければなりません。また、相当数の行が削除された表のデータにアクセスする際にも、読み取り操作が余分に必要となります。ユーザー表だけでなく、システム・カタログ表を再編成できるように、DB2 サーバーを設定することができます。

手順

データベースでの自動再編成を有効にするには、次の構成パラメーターをそれぞれ ON に設定します。

- **auto_maint**
- **auto_tbl_maint**
- **auto_reorg**

VOLATILE 表における自動索引再編成を有効にする

自動再編成を有効にすると、VOLATILE 表で索引再編成を実行できます。

このタスクについて

VOLATILE 表における自動索引再編成を有効にすると、リフレッシュ・インターバルごとに、VOLATILE 表の索引が再編成を必要としているかどうかを自動再編成が検査し、**REORG** コマンドを使用して必要な操作をスケジュールに入れます。

手順

VOLATILE 表における自動索引再編成を有効にするには、以下の手順を実行します。

1. **DB2_WORKLOAD** レジストリー変数を SAP に設定します。次の例は、**db2set** コマンドを使用してこの変数を設定する方法を示しています。

```
db2set DB2_WORKLOAD=SAP
```

データベースを再始動して、この設定を有効にします。

2. **auto_reorg** データベース構成パラメーターを ON に設定します。次の例は、**DB2 CLP** コマンド行インターフェースを使用してこのデータベース構成パラメーターを設定する方法を示しています。

```
UPDATE DB CFG FOR SAMPLE USING auto_reorg ON
```

auto_maint および **auto_tbl_maint** データベース構成パラメーターも ON に設定されていることを確認します。デフォルトでは、**auto_maint** および **auto_tbl_maint** は ON に設定されています。

3. **AUTOMAINT_SET_POLICY** または **AUTOMAINT_SET_POLICYFILE** プロシージャを呼び出して、**AUTO_REORG** ポリシー内の **numInxPseudoEmptyPagesForVolatileTables** 属性を設定します。この属性は、索引再編成の実行に必要な疑似削除キーを持つ空の索引ページの最小数を示します。次の例は、この属性の設定方法を示しています。

```
CALL SYSPROC.AUTOMAINT_SET_POLICY
('AUTO_REORG',
 BLOB('<?xml version="1.0" encoding="UTF-8"?>
<DB2AutoReorgPolicy
xmlns="http://www.ibm.com/xmlns/prod/db2/autonomic/config" >

<ReorgOptions dictionaryOption="Keep" indexReorgMode="Online"
useSystemTempTableSpace="false" numInxPseudoEmptyPagesForVolatileTables="20" />

<ReorgTableScope maxOfflineReorgTableSize="0">
<FilterClause>TABSCHEMA NOT LIKE 'SYS%'\</FilterClause>
</ReorgTableScope>
</DB2AutoReorgPolicy>')
)
```

MON_GET_INDEX 表関数に照会することにより
PSEUDO_EMPTY_PAGES、EMPTY_PAGES_DELETED、および
EMPTY_PAGES_REUSED 列の値をモニターすると、
numInxPseudoEmptyPagesForVolatileTables 属性に適した値を判別するのに役立ち
ます。

ExampleBANK のシナリオ: 表および索引のスペースの再利用

このシナリオでは、大規模な顧客基盤と多くの支店を持つ金融機関である ExampleBANK を取り上げます。この組織は、表および索引のスペースを再利用することを考えています。

ExampleBANK では、ユーザーが表と索引を使用できない状態にならないように、表と索引のスペースを再利用してそれらのサイズを一貫して管理します。ExampleBANK のように、絶え間なく変更される大量のデータを処理する組織には、表と索引のサイズを管理する方法が必要です。表および索引のスペースの再利用は、この目的を達成するのに有用です。

ExampleBANK のシナリオ: 表および索引のスペースの再利用 - スペース管理ポリシー

ExampleBANK のデータベース管理者である Olivia は、データベースのサイズを効率的に管理することに何年も苦労してきました。

通常の業務の最中に、不要になったデータを除去するために、表でバッチ削除が行われます。表および関連付けられている索引には、未使用のフリー・スペースがありますが、これは同じ表スペースにある他のオブジェクトがバッチ削除の完了後に使用することはできません。ExampleBank には、この未使用スペースを解放するためのスペース管理ポリシーがあります。Olivia は毎月、影響を受けたデータベースをオフラインにして、表と索引を再編成できるようにします。オブジェクトを再編成すると、スペースが解放されます。ダウン時間を最小化するために、作業はオフピーク時に行われます。

この表および索引のスペースの管理ポリシーには時間がかかり、また手操作による介入が必要です。また、このタスクを完了するために Olivia がデータベースをオフラインにするため、再編成の間、影響を受ける表と索引をユーザーが使用することはできません。

Olivia は、表および索引のスペースを再利用するための新しいコマンドとステートメント・パラメーターについて聞きました。表および索引に必要なスペースを管理するための新しい方法がもたらされました。

ExampleBANK のシナリオ: 表および索引のスペースの再利用 - 挿入時クラスタリング表の作成

挿入時クラスタリング (ITC) 表により、Olivia と ExampleBANK は、手操作による介入やデータベースのダウン時間なしにデータベース・サイズをより効率的に管理できるようになります。

Olivia は、以下のようにして挿入時クラスタリング表をテストとして作成します (ORGANIZE BY INSERT TIME 節を次のように使用すると、表が ITC 表として作成されます)。

```
DB2 CREATE TABLE T1(c1 int, c2 char(100), ...) IN TABLESPACE1 ORGANIZE BY INSERT TIME;
DB2 CREATE INDEX INX1 ON T1(C1);
```

ExampleBANK のシナリオ: 表および索引のスペースの再利用 - 表のスペース再利用の有効性の評価

時間が経過して、通常の操作が ITC 表で行われています。

ある時点でこの表にバッチ削除が実行され、オブジェクトの大部分が空になりました。Olivia はこのトラップされたスペースを TABLESPACE1 の他のオブジェクトで使用できるようにしようと考えます。Olivia は、ADMIN_GET_TAB_INFO_V97 関数を使用してスペース再利用の有効性を評価することができます。この関数は、表 T1 のデータ部分によって占有されている物理スペースに関する情報を収集できます。

Olivia は、以下のように表 T1 のデータに使用されている現行スペースの値を収集します。

```
SELECT DATA_OBJECT_L_SIZE,DATA_OBJECT_P_SIZE,RECLAIMABLE_SPACE
FROM TABLE (SYSPROC.ADMIN_GET_TAB_INFO_V97('OLIVIA', 'T1'))
```

DATA_OBJECT_L_SIZE	DATA_OBJECT_P_SIZE	RECLAIMABLE_SPACE
6500354	10242064	2560025

1 record(s) selected.

Olivia は、以下のように REORG コマンドを使用して、ユーザーに対して表へのアクセスを許可しながら、バッチ削除処理後に空になった再利用可能スペースをシステムに解放します。

```
REORG TABLE T1 RECLAIM EXTENTS
```

次に Olivia は以下のコマンドを実行し、最後にもう一度実行して、表スペースにどの程度のスペースが解放されたかを判別します。

```
SELECT DATA_OBJECT_L_SIZE,DATA_OBJECT_P_SIZE,RECLAIMABLE_SPACE
FROM TABLE (SYSPROC.ADMIN_GET_TAB_INFO_V97('OLIVIA', 'T1'))
```

DATA_OBJECT_L_SIZE	DATA_OBJECT_P_SIZE	RECLAIMABLE_SPACE
6500354	7682039	0

1 record(s) selected.

この結果は、データに使用されていた 2,560,025 KB のスペースが再利用されたことを示しています。

ExampleBANK のシナリオ: 表および索引のスペースの再利用 - 索引のスペース再利用の有効性の評価

Olivia はバッチ削除の後、表 T1 のデータ部分から再利用したスペースに注目します。Olivia はこの表の索引に、何らかのクリーンアップをしなければならないことを理解しています。

索引のスペースの再利用により、索引に占有されていたスペースを回復でき、その一方で、ユーザーはその索引を引き続き使用することができます。

表のスペースを再利用する場合と同様に、そうしたスペースは、他のオブジェクトで再利用できるように表スペースに戻されます。

Olivia は、以下のように ADMIN_GET_INDEX_INFO 関数を使用して再利用可能なスペースの量を調べます。

```
SELECT SUBSTR(INDNAME,1,10) AS INDNAME, IID, INDEX_OBJECT_L_SIZE, INDEX_OBJECT_P_SIZE,
RECLAIMABLE_SPACE FROM TABLE(SYSPROC.ADMIN_GET_INDEX_INFO('','OLIVIA','T1'))
AS INDEXINFO WHERE INDNAME='INX1'
```

INDNAME	IID	INDEX_OBJECT_L_SIZE	INDEX_OBJECT_P_SIZE	RECLAIMABLE_SPACE
INX1	3	1106752	1106752	846592

1 record(s) selected.

Olivia は、以下のように索引のスペースを再利用するために追加された新しいパラメーターを指定した **REORG** コマンドを使用して、索引のクリーンアップを完了しました。

```
REORG INDEXES ALL FOR TABLE T1 ALLOW WRITE ACCESS CLEANUP ALL RECLAIM EXTENTS
```

次に Olivia は以下のコマンドを実行し、最後にもう一度実行して、表スペースにどの程度のスペースが解放されたかを判別します。

```
SELECT SUBSTR(INDNAME,1,10) AS INDNAME, IID, INDEX_OBJECT_L_SIZE, INDEX_OBJECT_P_SIZE,
RECLAIMABLE_SPACE FROM TABLE(SYSPROC.ADMIN_GET_INDEX_INFO('','OLIVIA','T1'))
AS INDEXINFO WHERE INDNAME='INX1'
```

INDNAME	IID	INDEX_OBJECT_L_SIZE	INDEX_OBJECT_P_SIZE	RECLAIMABLE_SPACE
INX1	3	259776	259776	0

1 record(s) selected.

この結果は、見積もりで 846,592 KB のスペースが再利用されたことを示しています。Olivia は、スペースの再利用後の物理サイズから元の物理サイズを減算すると、再利用された実際のスペースが 846,976 KB であることに気がきます。

ExampleBANK のシナリオ: 表および索引のスペースの再利用 - 既存の表から挿入時クラスタリング表への変換

Olivia は、挿入時クラスタリング表を使用することの利点を知ります。Olivia は、このソリューションを実動データベースの既存の表に使用したいと考えています。この変更は、オンライン表移動ユーティリティを使用して実行することができます。

Olivia は、次のスキーマを持つシステムに存在する表を持っています。このシナリオでは、およその挿入時刻順 (C4) でデータを配置するのに便利な列が表に実際にあるとします。

```
CREATE TABLE EXMP.T1 (C1 INT, C2 CHAR(50), C3 BIGINT, C4 DATE) IN TABLESPACE1
CREATE INDEX INX1 ON EXMP.T1(C4)
```

Olivia は次のようにして、変換のターゲット表を作成します。

```
DB2 CREATE TABLE EXMP.NEW1(C1 INT, C2 CHAR(50), C3 BIGINT, C4 DATE)
IN TABLESPACE1 ORGANIZE BY INSERT TIME
```

このスキーマは元の表と同一のものですが、Olivia は、ORGANIZE BY INSERT TIME キーワードを使用して、必ずこの表が時刻によりクラスター化されるようにします。

Olivia はオンライン表移動ストアード・プロシージャを使用して変換を実行します。

クラスタリング索引は列 C4 にあるため、Olivia は挿入時刻のおよその順序がわかります。このような列がない表の場合、挿入時クラスタリング表への移動によるスペース再利用の効果がはっきりとわからない場合があります。このように効果がすぐに現れない理由は、新しいデータが古いデータと一緒にグループ化されるためです。

```
DB2 CALL SYSPROC.ADMIN_MOVE_TABLE('EXMP', 'T1', 'NEWT1', NULL, 'MOVE')
```

EXMP.T1 は時刻クラスタリング表形式になりました。これで、その後バッチ削除を行った後でエクステンツを再利用する準備ができました。

ExampleBANK のシナリオ: 表および索引のスペースの再利用 - 索引の一般的な保守

Olivia は、索引と表の一部で、スペースの消費と動作が詳しく追跡されていないことに気付きます。影響を受けた表スペースのすべてのスペースをクリーンアップして再利用できるかどうかを、スクリプトによって定期的に調べることができます。

Olivia は次のような REORGCHK コマンドを使用して、索引のクリーンアップで効果が上がるかどうかを判別します。

```
REORGCHK ON TABLE USER1.TBL1;
RUNSTATS 中 ....
```

表の統計:

```
F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (データ・ページ数の有効スペース使用率) > 70
F3: 100 * (必須ページ数 / 合計ページ数) > 80
```

SCHEMA.NAME	CARD	OV	NP	FP	ACTBLK	TSIZE	F1	F2	F3	REORG
Table: USER1.TBL1	3400	0	201	295	-	775200	0	67	70	---

索引統計:

```
F4: CLUSTERRATIO または正規化 CLUSTERFACTOR > 80
F5: 100 * (リーフ・ページの使用スペース / 空ではないリーフ・ページで使用可能なスペース) > MIN(50, (100 - PCTFREE))
F6: (100 - PCTFREE) * (1 つ下のレベルの索引で使用可能な合計スペース / すべてのキーに必要な合計スペース) < 100
F7: 100 * (疑似削除された RID の数 / RID の合計数) < 20
F8: 100 * (疑似的な空のリーフ・ページ数 / リーフ・ページの合計数) < 20
```

SCHEMA.NAME	INDCARD	LEAF	ELEAF	LVLS	NDEL	KEYS	LEAF_REC SIZE
Table: USER1.TBL1							
Index: USER1.TBL1_INX1	3400	244	0	3	1595	2882	205

(continued)

NLEAF_REC SIZE	LEAF_PAGE_OVERHEAD	NLEAF_PAGE_OVERHEAD	PCT_PAGES_SAVED	F4	F5	F6
205	132		132	0	100	64 9

```
(continued)
F7 F8 REORG
-----
31 0 ---*-
-----
```

CLUSTERRATIO or normalized CLUSTERFACTOR (F4) will indicate REORG is necessary for indexes that are not in the same sequence as the base table. When multiple indexes are defined on a table, one or more indexes may be flagged as needing REORG. Specify the most important index for REORG sequencing.

Tables defined using the ORGANIZE BY clause and the corresponding dimension indexes have a '*' suffix to their names. The cardinality of a dimension index is equal to the Active blocks statistic of the table.

この出力の公式 F7 で、TBL1_INX1 の索引では、**REORG** コマンドを使用した索引のクリーンアップで効果が上がることが示されています。Olivia は、索引をクリーンアップするコマンドを以下のように実行します。

```
REORG INDEXES ALL FOR TABLE USER1.TBL1 ALLOW WRITE ACCESS CLEANUP;
```

REORG INDEXES CLEANUP コマンドによってスペースが解放されたことにより現在再利用できるスペースの量を判断するために、Olivia は **ADMIN_GET_INDEX_INFO** ルーチンを使用します。

```
SELECT RECLAIMABLE_SPACE
      FROM TABLE(sysproc.admin_get_index_info('T','USER1', 'TBL1'))
      AS t";
```

```
RECLAIMABLE_SPACE
-----
                14736
```

1 record(s) selected.

Olivia がこの値 (KB 単位) を大きいと見なした場合、以下のようにして **REORG INDEX RECLAIM EXTENTS** コマンドを実行できます。

```
REORG INDEXES ALL FOR TABLE USER1.TBL1 ALLOW WRITE ACCESS RECLAIM EXTENTS;
```

Olivia はこの作業を定期的なインターバルでスケジュールして、問題の索引に必要な以上のスペースが保持されないようにすることができます。定期的にスケジュールされたこの作業により、他のユーザーが問題の索引の使用を禁止されることはありません。

ExampleBANK のシナリオ: 表および索引のスペースの再利用 - **AUTO_REORG** ポリシー

Olivia は手動チェックを実行する代わりに自動的に索引再編成を管理する方法を選択することができます。

Olivia はサンプル・ディレクトリー `SQLLIB/samples/automaintcfg` に `DB2AutoReorgPolicySample.xml` というファイルを見つけます。このファイルで、Olivia は以下の行に注目します。

```
<ReorgOptions dictionaryOption="Keep" indexReorgMode="Offline"
  useSystemTempTableSpace="false" />
```

Olivia は、`reclaimExtentsSizeForIndexObjects` しきい値が 51,200 KB (50 MB) を超えたら **RECLAIM EXTENTS** オプションを指定した自動再編成を実行すると決定します。Olivia は、`DB2AutoReorgPolicySample.xml` を `autoreorg_policy.xml` というファイルにコピーし、サンプル内の行を以下の値に変更します。

```
<ReorgOptions dictionaryOption="Keep" indexReorgMode="Online"
  useSystemTempTableSpace="false" indexReclaimExtents="51200" />
```

次に Olivia は、ポリシーを次のように設定します。

```
cp $HOME/autoreorg_policy.xml $HOME/sql1lib/tmp/.  
db2 "call sysproc.automaint_set_policyfile( 'AUTO_REORG', 'autoreorg_policy.xml')"
```

最後のステップとして、Olivia は AUTO REORG が有効になったことを確認します。詳しくは関連タスクを参照してください。

これでポリシーが定義され、AUTO REORG が有効化されたため、reclaimExtentsSizeForIndexObjects しきい値が 51,200 KB (50 MB) を超えると自動再編成が開始します。

アプリケーション設計

データベース・アプリケーション設計は、アプリケーション・パフォーマンスに影響を与える要因の 1 つです。データベース・アプリケーションのパフォーマンスを最大化するうえで役立つアプリケーション設計上の考慮事項の詳細は、このセクションで調べてください。

アプリケーションのプロセス、並行性、およびリカバリー

すべての SQL プログラムは、アプリケーション・プロセス またはエージェントの一部として実行されます。アプリケーション・プロセスには 1 つ以上のプログラムの実行が関係しており、これがデータベース・マネージャーがリソースを割り当てたりロックしたりする場合の単位となります。異なるいくつかのプログラムの実行、または同じプログラムの複数の異なる実行には、異なる複数のアプリケーション・プロセスが関係していることがあります。

同時に複数のアプリケーション・プロセスが同じデータへのアクセスを要求できます。そのような状況でデータ保全性を維持するために使用されるメカニズムとしてロックがあります。これは、例えば 2 つのアプリケーション・プロセスが同時に同じデータ行を更新することを防ぐなどの効果があります。

データベース・マネージャーは、あるアプリケーション・プロセスによって行われた変更でまだコミットされていないものが、誤って他のプロセスによって認識されることがないように、ロックを獲得します。アプリケーション・プロセスが終了すると、データベース・マネージャーは、そのプロセスのために獲得および保持していたロックをすべて解除します。ただし、それより早い時期にロックを解除するように、アプリケーション・プロセスから明示的に要求することもできます。これはコミット操作を使用して行います。これにより作業単位の間を獲得したロックが解除され、さらに作業単位の間に加えられた変更がデータベースにコミットされます。

作業単位 (UOW) とは、アプリケーション・プロセス内のリカバリー可能な一連の操作のことをいいます。作業単位は、アプリケーション・プロセスが開始された時、またはアプリケーション・プロセスの終了以外の理由で直前の UOW が終了した時に開始します。作業単位は、コミット操作、ロールバック操作、またはアプリケーション・プロセスの終了によって終了します。コミットまたはロールバック操作は、終了する UOW の中で行われたデータベースへの変更には影響しません。

データベース・マネージャーは、アプリケーション・プロセスにより行われた変更でまだコミットされていないものをバックアウトするための手段を備えています。これは、アプリケーション・プロセス側に障害が発生したとき、またはデッドロック

クやロック・タイムアウト状態になった場合に必要になる場合があります。アプリケーション・プロセスで行ったデータベースへの変更を取り消すように、同じプロセスで明示的に要求することができます。これはロールバック 操作を使って行います。

このような変更がコミットされないまま残っている間は、他のアプリケーション・プロセスはそれらの変更を認識することはできませんし、変更をロールバックすることも可能です。ただし、優先される分離レベルが非コミット読み取り (UR) である場合には、この限りではありません。データベースの変更内容がコミットされると、他のアプリケーション・プロセスからその変更内容にアクセスできるようになり、ロールバックできなくなります。

DB2 コール・レベル・インターフェース (CLI) および組み込み SQL を使用すると、**並行トランザクション** と呼ばれる接続モードを使用できます。これは、それぞれが独立したトランザクションである複数の接続をサポートします。1 つのアプリケーションに、同じデータベースへの複数の同時接続を持たせることができます。

データベース・マネージャーがアプリケーション・プロセスのために獲得したロックは、UOW が終了するまで保持されます。ただし、分離レベルがカーソル固定 (CS、カーソルが行から行に移動されるとロックは解除される) か非コミット読み取り (UR) の場合はこの限りではありません。

アプリケーション・プロセスがそれ自体のロックのために操作できなくなるということは決してありません。ただし、アプリケーションが並行してトランザクションを使用する場合、一方のトランザクションによるロックのために他方のトランザクションの操作が影響を受ける可能性があります。

UOW の開始と終了によって、アプリケーション・プロセス内の整合点 が定義されます。例えば、銀行業務のトランザクションで、ある口座から別の口座へ資金を振り込むことがあります。このようなトランザクションでは、その資金を第 1 の口座から減算してから、第 2 の口座に加算する、ということが必要になります。減算のステップの直後の段階では、データに不整合があります。資金を第 2 の口座に加算して初めて整合性が取り戻されます。両方のステップが完了したときに、コミット 操作を実行して UOW を終了させれば、他のアプリケーション・プロセスが変更内容を利用できるようになります。1 つの UOW が終了する前に障害が発生した場合、データベース・マネージャーはコミットされていない変更内容をロールバックし、データ整合性を回復します。

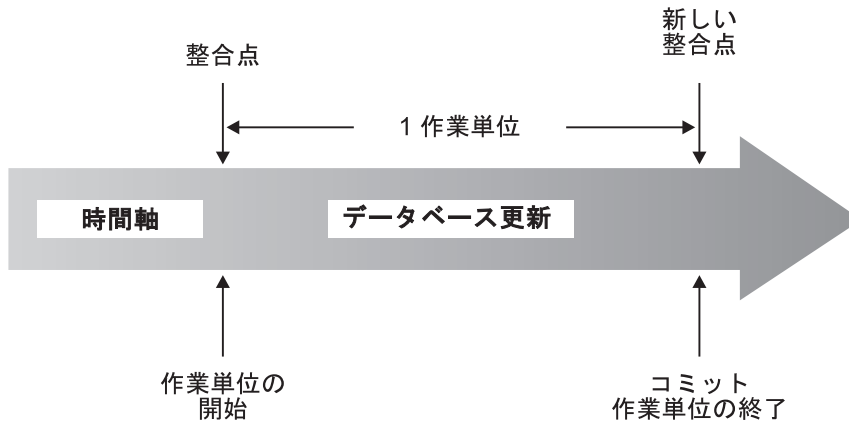


図 21. COMMIT ステートメントの作業単位

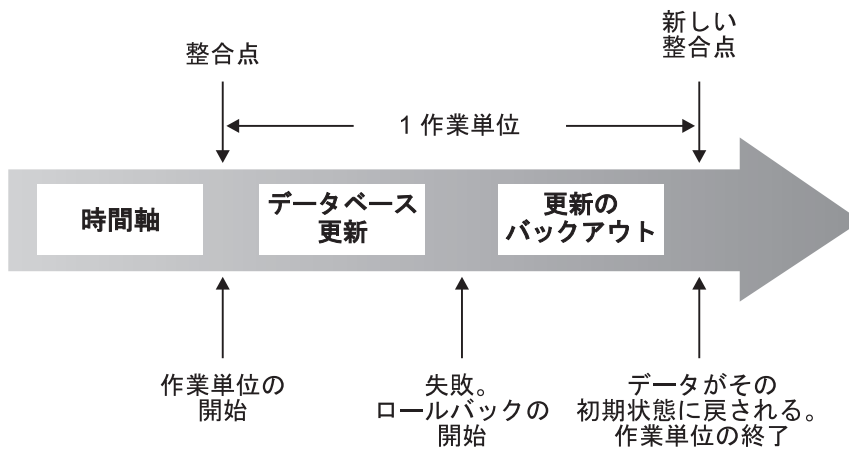


図 22. ROLLBACK ステートメントの作業単位

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、データ保全性を確保しつつ、ユーザーがこれらの変更を行え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有することです。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、両方とも同じ行を読み取り、その読み取ったデータに基づいてそのいずれかの列の新しい値を計算するとします。A がその行を更新し、次に B もその行を更新すると、A の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 がある値を更新し、それがコミットされる前に B がその値を読み取るとします。その後 A がその更新を取り消した場合、B が実行する計算は、無効なデータに基づくものとなる可能性があります。

- **反復不能読み取り。**アプリケーション A がある行を読み取り、次いで他の要求を処理するとします。その間、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、A が元の行を再び読み取ろうとすると、修正された行が表示されます。あるいは元の行が削除されていることがわかる場合もあります。
- **幻像読み取り。**アプリケーション A が、いくつかの検索基準に基づいて行セットを読み取る照会を実行するとします。アプリケーション B が新しいデータを挿入するか、既存データを更新しますが、それはアプリケーション A の照会を満たすこととなります。アプリケーション A が同じ作業単位の中で照会を再び実行すると、追加の（「幻像」）値が返されます。

グローバル一時表では、並行性の問題はありませぬ。グローバル一時表を使用できるのは、それを宣言または作成したアプリケーションのみだからです。

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) を参照する SQL ステートメントをサブミットできます。そのようなデータ・ソース (それぞれが 1 つの DBMS およびデータで構成される) を参照するために、DB2 サーバーはニックネームを使用します。ニックネームは、他の DBMS に属する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 サーバーは、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。例えば、表およびビューについての情報を移動する場合、その情報への参照 (ニックネームを使用する) 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できます。アプリケーションがニックネームでデータにアクセスすると、DB2 サーバーはそのデータ・ソースにある並行性制御プロトコルを使用して、分離レベルが確実に強制されるようにします。DB2 サーバーは、データ・ソースで要求された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。

分離レベル

アプリケーション・プロセスに関連付けられた分離レベルは、そのプロセスによりアクセスされているデータのロックの度合い、または並行して実行されている他のプロセスからそのデータを分離する度合いを決定します。分離レベルは、作業単位の持続期間内で有効です。

したがって、アプリケーション・プロセスの分離レベルは、以下を指定します。

- アプリケーションによって読み取りまたは更新が行われる行を、並行して実行される他のアプリケーション・プロセスから使用できる度合い。
- 並行して実行される他のアプリケーション・プロセスの更新アクティビティによってアプリケーションが影響を受ける度合い。

静的 SQL ステートメントの分離レベルは、パッケージの属性として指定され、そのパッケージを使用するアプリケーション・プロセスに適用されます。分離レベルは、ISOLATION バインドまたはプリコンパイル・オプションを設定することにより、プログラム準備処理中に指定されます。動的 SQL ステートメントの場合、デ

フォルトの分離レベルは、ステートメントを作成するパッケージに指定された分離レベルです。SET CURRENT ISOLATION ステートメントを使用すると、セッション内で発行される動的 SQL ステートメントに対して別の分離レベルを指定できます。詳しくは、『CURRENT ISOLATION 特殊レジスター』を参照してください。静的 SQL ステートメントと動的 SQL ステートメントのどちらの場合でも、select-statement 内の isolation-clause は、特殊レジスター (設定されている場合) と BIND オプションの両方の値をオーバーライドします。詳しくは、『Select-statement』を参照してください。

分離レベルはロックにより強制され、並行アプリケーション・プロセスによるデータ・アクセスは、使用されるロックのタイプに応じて制限または禁止されます。宣言済み一時表とその行は、宣言したアプリケーションしかアクセスできないので、ロックできません。

データベース・マネージャーでは、大きく分けて次の 3 つのロック・カテゴリーがサポートされています。

共有 (S)

S ロックでは、並行アプリケーション・プロセスの操作は、データへの読み取り専用操作に限定されます。

更新 (U)

U ロックでは、並行アプリケーション・プロセスの操作は、行の更新を宣言したのではない限り、データへの読み取り専用操作に限定されます。データベース・マネージャーは、行を現在見ているプロセスがそれを更新する可能性があると想定します。

排他 (X)

X ロックでは、同時アプリケーション・プロセスがどのような形であれ、そのデータにアクセスできないようにします。これは、読み取りはできてもデータの変更はできない非コミット読み取り (UR) の分離レベルのアプリケーション・プロセスには当てはまりません。

分離レベルとは関係なく、データベース・マネージャーは、挿入、更新、または削除の対象となる行のすべてに排他ロックをかけます。このため、どの分離レベルでも、アプリケーション・プロセスが 1 作業単位の間に変更する行は、その作業単位が完了するまで他のアプリケーション・プロセスにより変更されることは決してありません。

データベース・マネージャーは 4 つの分離レベルをサポートします。

- 177 ページの『反復可能読み取り (RR)』
- 177 ページの『読み取り固定 (RS)』
- 178 ページの『カーソル固定 (CS)』
- 179 ページの『非コミット読み取り (UR)』

注: 一部のホスト・データベース・サーバーはコミットなし (NC) 分離レベルをサポートします。その他のデータベース・サーバーでは、この分離レベルは非コミット読み取り分離レベルに似た動作をします。

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更新したりする場合には、後で説明されているものほど注意が必要になります。

反復可能読み取り (RR)

反復可能読み取り 分離レベルでは、1 つの作業単位 (UOW) の間にアプリケーションが参照する行がすべてロックされます。アプリケーションが同じ作業単位の中で 2 回 SELECT ステートメントを発行した場合には、いずれの場合も同じ結果が返されます。RR では、更新の消失の可能性はなく、コミットされていないデータへのアクセス、反復不能読み取り、および幻像読み取りは行えません。

RR では、アプリケーションは、UOW が完了するまでに、必要な回数だけ行の取得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 UOW が完了するまで、結果セットに影響を与える行を更新、削除、または挿入することができません。RR 分離レベルの下で実行されるアプリケーションは、コミットされていない他のアプリケーションによる変更は認識できません。この分離レベルでは、戻されるデータすべてをアプリケーションが認識するまでは、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、それらのデータはすべて未変更のままになります。

取得される行だけでなく、参照されるすべての行がロックされます。例えば、10000 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 行しか該当しなくとも、それら 10000 行すべてにロックがかけられます。照会が再度実行された場合には、照会によって参照される行のリストに加えられることになる行については、別のアプリケーションが挿入または更新を行うことができません。これにより、幻像読み取りを防ぎます。

RR は多数のロックを獲得できるため、この数が **locklist** および **maxlocks** データベース構成パラメーターで指定した限度を超える可能性があります。ロック・エスカレーションが発生する可能性がある場合、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・ロックを獲得することがあります。表レベルのロックをかけたくない場合は、読み取り固定分離レベルを使用します。

参照制約を評価する際、ユーザーが以前に設定した分離レベルに関係なく、外部表のスキャン時に使用される分離レベルが DB2 サーバーによって RR にアップグレードされることがあります。これが起こるとさらに多くのロックがコミットの時まで保持されるため、デッドロックやロックのタイムアウトが発生する可能性が高くなります。これらの問題を避けるには、参照整合性スキャンが代わりに使用できる、外部キー列のみを含む索引を作成します。

読み取り固定 (RS)

読み取り固定 分離レベルでは、ある作業単位の間にはアプリケーションが取得する行のみにロックをかけます。RS は、UOW 中に読み取られた修飾行が、UOW 完了時まで他のアプリケーション・プロセスによって変更できないようにします。また他のアプリケーション・プロセスによって行に加えられた変更が、そのプロセスによってコミットされるまで読み取れないようにします。RS では、コミットされていないデータへのアクセスおよび反復不能読み取りは行えません。ただし、幻像読

み取りは行えます。幻像読み取りは、古い値は元のアプリケーションの検索条件を満たしていないものの、新しい更新済みの値は検索条件を満たしている行を並行更新することによって、もたらされる場合があります。

例えば幻像読み取り行は、次のような状況で発生することがあります。

1. アプリケーションのプロセス P1 が、一定の検索条件を満たす行のセット n を読み取る。
2. 次にアプリケーション・プロセス P2 が、検索条件を満たす 1 行以上の行を挿入し、それらの新規挿入行をコミットする。
3. P1 が同じ検索条件で行のセットを再度読み取り、元の行と P2 によって挿入された行を両方とも獲得する。

DB2 pureScale環境では、この分離レベルで稼働しているアプリケーションは、行が別のメンバーに対して並行更新された場合に、以前にコミットされた行値をリジェクトすることがあります。この動作をオーバーライドするには、WAIT_FOR_OUTCOME オプションを指定します。

この分離レベルでは、戻されるデータすべてをアプリケーションが認識するまでは、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、それらのデータはすべて未変更のままになります。

RS 分離レベルでは、高度の並行性が提供されると共に、データの表示が一定になります。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。

RS 分離レベルは以下の条件下で作動するアプリケーションに適しています。

- 並行環境で作動する
- 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
- 1 つの作業単位の間同じ照会を 2 回以上発行しない、または 1 つの作業単位の間同じ照会を 2 回以上発行したときに同じ結果セットを得る必要がない

カーソル固定 (CS)

カーソル固定 分離レベルは、トランザクションの際にアクセスする行にカーソルを置いたまま、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、変更がコミットされるまでロックは保持されます。

この分離レベルでは、更新可能なカーソルがある行に置かれている間、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。CS では、他のアプリケーションの非コミット・データにアクセスすることはできません。ただし、反復不能読み取りおよび幻像読み取りは行えます。

CS はデフォルトの分離レベルです。コミットされたデータだけを認識する必要があり、並行性を最大にする場合にこれは適しています。

DB2 pureScale環境では、この分離レベルで稼働しているアプリケーションは、行が別のメンバーに対して並行更新された場合に、以前にコミットされた行値を返すか、リジェクトすることがあります。並行アクセス解決設定の WAIT FOR OUTCOME オプションを使用すると、この動作をオーバーライドできます。

注: バージョン 9.7 で導入された *currently committed* セマンティクスでは、今までのようにコミットされたデータのみが返されますが、読み取り側は更新側が行ロックを解除するまで待機しなくなりました。代わりに読み取り側は、現在コミット済みのバージョンに基づくデータ、つまり書き込み操作の開始前のデータを返します。

非コミット読み取り (UR)

非コミット読み取り 分離レベルでは、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。さらに UR の場合、別のアプリケーションが表を変更またはドロップしようとするのでない限り、読み取り中の行に別のアプリケーションがアクセスすることが可能です。

UR では、コミットされていないデータへのアクセス、反復不能読み取り、および幻像読み取りが可能です。この分離レベルは、読み取り専用表に対して照会を実行する場合、または SELECT ステートメントのみを発行する場合で、かつ他のアプリケーションからコミットされていないデータを見られることが問題にはならない場合に適しています。

UR 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。
- トランザクションの処理中は、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、および索引は使用できません。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。UR で作動している更新可能なカーソルは、CS 分離レベルの場合と同じ働きをします。

非コミット読み取りを行うアプリケーションが未確定カーソルを使用する場合、実行時に CS 分離レベルを使用する可能性があります。PREP または BIND コマンドの BLOCKING オプションの値が UNAMBIG (デフォルト) である場合、未確定カーソルが CS にエスカレートされる場合があります。このエスカレーションを防ぐには、以下のようにします。

- 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 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 および STATICREADONLY YES 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を読み取り専用として扱えます。

分離レベルの比較

表 8 は、サポートされる分離レベルについて要約しています。

表 8. 分離レベルの比較

	UR	CS	RS	RR
アプリケーションは、他のアプリケーションが処理した変更内容でコミットされていないものを認識できますか?	はい	いいえ	いいえ	いいえ
アプリケーションは、他のアプリケーションが処理した変更内容でコミットされていないものを更新できますか?	いいえ	いいえ	いいえ	いいえ
ステートメントの再実行は、他のアプリケーション・プロセスに影響される可能性がありますか? ¹	はい	はい	はい	いいえ ²
更新された行が他のアプリケーション・プロセスにより更新される可能性がありますか? ³	いいえ	いいえ	いいえ	いいえ
更新された行が、UR 以外の分離レベルで実行中の他のアプリケーション・プロセスにより読み取られる可能性がありますか?	いいえ	いいえ	いいえ	いいえ
更新された行が、UR の分離レベルで実行中の他のアプリケーション・プロセスにより読み取られる可能性がありますか?	はい	はい	はい	はい
アクセスされた行が他のアプリケーション・プロセスにより更新される可能性がありますか? ⁴	はい	はい	いいえ	いいえ
アクセスされた行が他のアプリケーション・プロセスにより読み取られる可能性がありますか?	はい	はい	はい	はい
現在行が他のアプリケーション・プロセスにより更新または削除される可能性がありますか? ⁵	はい/いいえ ⁶	はい/いいえ ⁶	いいえ	いいえ

表 8. 分離レベルの比較 (続き)

	UR	CS	RS	RR
注:				
1.	幻像読み取り現象 の例には、次のようなものがあります。まず作業単位 UW1 が、ある検索条件を満たしている一連の n 個の行を読み取ります。作業単位 UW2 が、その同じ検索条件を満たす 1 つ以上の行を挿入してからコミットします。その後 UW1 が同じ検索条件で読み取りを繰り返すと、別の結果セットが認識されます。最初に読み取った行のほかに UW2 で挿入された行が追加されています。			
2.	読み取りを行ってから次の読み取りを行うまでの間に、ラベル・ベースのアクセス制御 (LBAC) 資格情報が変化した場合、アクセス可能な行が異なるために、2 度目の読み取りの結果は異なる場合があります。			
3.	アプリケーションが表に対する読み取りと書き込みの両方を行っている場合、分離レベルはアプリケーションに保護を提供しません。例えば、アプリケーションは表でカーソルをオープンし、それからその同じ表に挿入、更新、または削除の実行を実行するとします。オープン・カーソルでさらに行を取り出してゆくと、アプリケーションが矛盾するデータを見つける場合があります。			
4.	反復不能読み取り現象 の例には、次のようなものがあります。まず作業単位 UW1 が行を読み取ります。作業単位 UW2 がその行を変更し、コミットします。その後 UW1 がもう一度その行を読み取ると、値が異なる場合があります。			
5.	ダーティ読み取り現象 の例には、次のようなものがあります。まず作業単位 UW1 が行を変更します。UW1 がコミットする前に、作業単位 UW2 がその行を読み取ります。次に UW1 が変更内容をロールバックすると、UW2 は存在しないデータを読み取ったこととなります。			
6.	UR または CS では、カーソルが更新可能でない場合、現在行を他のアプリケーション・プロセスによって更新または削除できる場合もあります。例えば、バッファリングによって、クライアントの現在行とサーバーの現在行との間に相違が生じる場合があります。さらに CS で <code>currently committed</code> セマンティクスを使用しているときに、読み取り中の行に非コミット更新保留が含まれる可能性があります。この場合は、常に現在コミット済みの行がアプリケーションに返されます。			

分離レベルのまとめ

表 9 は、さまざまな分離レベルに関連した並行性の問題のリストです。

表 9. 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	
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
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

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処理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

ックになる可能性も、分離レベルごとに異なります。表 10 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。

表 10. 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

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分離されるかを定めるものなので、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。

このタスクについて

指定する分離レベルは、作業単位 (UOW) の持続期間中に有効です。SQL または XQuery ステートメントのコンパイル時に使用される分離レベルを判別するために、以下のヒューリスティックが使用されます。

- 静的 SQL の場合:
 - ステートメントに *isolation-clause* が指定されている場合は、その節の値が使用されます。
 - ステートメントに *isolation-clause* が指定されていない場合は、データベースへのパッケージのバインド時にそのパッケージ用に指定された分離レベルが使用されます。
- 動的 SQL の場合:
 - ステートメントに *isolation-clause* が指定されている場合は、その節の値が使用されます。
 - ステートメントに *isolation-clause* が指定されておらず、SET CURRENT ISOLATION ステートメントが現行セッション内で発行されている場合は、CURRENT ISOLATION 特殊レジスターの値が使用されます。
 - ステートメントに *isolation-clause* が指定されておらず、SET CURRENT ISOLATION ステートメントが現行セッション内で発行されていない場合は、データベースへのパッケージのバインド時にそのパッケージ用に指定された分離レベルが使用されます。
- 静的および動的 XQuery ステートメントの場合、環境の分離レベルによって、XQuery 式の評価の際に使用される分離レベルが決まります。

注: 作成された商用アプリケーションの多くでは、分離レベルを選択するための手段を備えています。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。

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。

手順

- ステートメント・レベルの場合:

注: XQuery ステートメントの分離レベルについては、ステートメント・レベルで指定できません。

WITH 節を使用します。WITH 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。

この分離レベルは、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。

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

• 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 特殊レジスターにより、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動的 SQL ステートメントすべてに、その分離レベルが適用されます。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...RESET ステートメントが発行されるまで有効です。

• 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、PREP または BIND コマンドの ISOLATION オプションを使用します。sqlprep または sqlabndx 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・レベルとしてカーソル固定 (CS) が使用されます。

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 = 'pkgname'
     and pkgschema = 'pkgschema'
```

ここで、*pkgname* はパッケージの非修飾名、*pkgschema* はパッケージのスキーマ名です。これらの名前は、両方とも大文字で指定しなければなりません。

• 実行時に JDBC または SQLJ で作業する場合:

注: JDBC および SQLJ は、DB2 サーバー上の 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。

SQLJ でパッケージを作成 (およびその分離レベルを指定) するには、SQLJ プロファイル・カスタマイザー (**db2sqljcustomize** コマンド) を使用します。

- 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・インターフェイス (CLI) を使用すると、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* 引数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。

- **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 (CLP) 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。

REXX および CLP は、デフォルトの CS 分離レベルを使用してデータベースに接続されます。この分離レベルに変更しても、接続状態は変更されません。

REXX アプリケーションで使用されている分離レベルを調べるには、SQLISL の事前定義された REXX 変数の値を確認します。この値は、**CHANGE ISOLATION LEVEL** コマンドを実行するたびに更新されます。

タスクの結果

currently committed セマンティクス

currently committed セマンティクスを使用すると、コミットされたデータだけが読み取り側に返されます。ただし、読み取り側は、書き込み側により行ロックが解除されるまで待機しません。代わりに、読み取り側は、現在コミット済みのバージョンのデータ (つまり書き込み操作が開始される前のバージョンのデータ) に基づくデータを返します。

行レベルのロックを行うカーソル固定 (CS) 分離レベルでは、ロック・タイムアウトおよびデッドロックが生じる可能性があります。これは特に、このような問題を回避するように設計されていないアプリケーションの場合に生じる可能性があります。一部の高スループットのデータベース・アプリケーションでは、トランザクション処理中に発行されるロックの待機を許容できません。また、非コミット・データの処理は許容できなくても、読み取りトランザクションにおける非ブロッキング動作を必要とするアプリケーションもあります。

デフォルトでは、*currently committed* セマンティクスは新規データベースでオンになっています。この新しい動作を利用するためにアプリケーションに変更を加える必要はありません。デフォルトの動作をオーバーライドするには、**cur_commit** データベース構成パラメーターを **DISABLED** に設定します。動作のオーバーライドは、例えば、内部ロジックを同期するために書き込み側でブロッキングが必要なアプリ

ケーションの場合などに役立つことがあります。V9.5 以前からのデータベース・アップグレード時には、それまでのリリースと同じ動作を維持するために **cur_commit** 構成パラメーターが **DISABLED** に設定されます。カーソル固定スキャン時に **currently committed** を使用する場合は、アップグレード後に **cur_commit** 構成パラメーターを **ON** に設定する必要があります。

currently committed セマンティクスは、カタログ表が関係しない読み取り専用スキャンおよび制約の評価または強制に使用される内部スキャンにのみ適用されます。**currently committed** セマンティクスはスキャン・レベルで決定されるため、書き込み側のアクセス・プランに **currently committed** スキャンが含まれる可能性があります。例えば、読み取り専用副照会のスキャンには、**currently committed** セマンティクスが関係する可能性があります。

currently committed セマンティクスは分離レベル・セマンティクスに従うので、**currently committed** セマンティクスで実行されているアプリケーションは、引き続き分離レベルを考慮に入れます。

currently committed セマンティクスでは、書き込み側のログ・スペースを増やす必要があります。トランザクション中のデータ行の最初の更新をログに記録するために、追加スペースが必要です。行の現在コミット済みイメージを取得するためにこのデータが必要となります。ワークロードによっては、使用される合計ログ・スペースに関して、このことがかなりの影響を与えることもあれば、微々たる影響しかない場合もあります。追加のログ・スペースに関する要件は、**cur_commit** データベース構成パラメーターが **DISABLED** に設定されている場合には当てはまりません。

制約事項

currently committed セマンティクスには、以下の制約事項が適用されます。

- データの更新または削除操作に使用されるセクション内のターゲット表オブジェクトは、**currently committed** セマンティクスを使用しません。変更される行は、更新操作の一部である照会述部を満たした後で変化することがないように、ロック保護する必要があります。
- 行に対して非コミット変更を行ったトランザクションは、行の現在コミット済みバージョンを判別するために、現在コミット済みリーダーに強制的に該当ログ・レコードにアクセスさせます。ログ・バッファーにもうないログ・レコードでも物理的に読み取ることは可能ですが、**currently committed** セマンティクスはログ・アーカイブからのログ・ファイルの取得をサポートしていません。これが影響するのは、無限ロギングを使用するように構成されたデータベースのみです。
- 以下のスキャンは、**currently committed** セマンティクスを使用しません。
 - カタログ表スキャン
 - 参照整合性制約を適用するために使用されるスキャン
 - **LONG VARCHAR** または **LONG VARGRAPHIC** 列を参照するスキャン
 - 範囲がクラスター化された表 (RCT) スキャン
 - 空間索引または拡張索引を使用するスキャン

例

currently committed セマンティクスを使用してデッドロックを回避する以下のシナリオについて考えてみましょう。このシナリオでは、ステップ 1 に示すように 2 つのアプリケーションが 2 つの別々の表を更新しますが、まだコミットしていません。その後、ステップ 2 に示すように各アプリケーションは、他方のアプリケーションが更新した表を、読み取り専用カーソルを使用して読み取ろうとします。これらのアプリケーションは CS 分離レベルで実行されています。

ステップ	アプリケーション A	アプリケーション B
1	update T1 set col1 = ? where col2 = ?	update T2 set col1 = ? where col2 = ?
2	select col1, col3, col4 from T2 where col2 >= ?	select col1, col5, from T1 where col5 = ? and col2 = ?
3	commit	commit

currently committed セマンティクスを使用しない場合、カーソル固定分離レベルで実行されているこうしたアプリケーションではデッドロックが生じ、いずれかのアプリケーションが失敗することがあります。これは、各アプリケーションが、他のアプリケーションによって更新されているデータを読み取る必要がある場合に生じます。

currently committed セマンティクスでは、ステップ 2 で照会を実行しているアプリケーションの一方が、他方のアプリケーションによって現在更新中のデータを必要とする場合、前者のアプリケーションはロックの解除を待機しません。そのため、デッドロックが発生することはありません。代わりに、前者のアプリケーションは、コミット済みバージョンの古いデータを見つけて使用します。

非コミット追加を無視するオプション

DB2_SKIPINSERTED レジストリー変数は、カーソル固定 (CS) や読み取り固定 (RS) の分離レベルを使用するステートメントで非コミット・データ挿入を無視できるかどうかを制御します。

非コミット挿入は、**DB2_SKIPINSERTED** レジストリー変数の値に応じて、2 つの方法のうち 1 つで処理されます。

- 値が ON の場合、DB2 サーバーは非コミット挿入を無視します。これにより多くの場合に並行性が向上し、ほとんどのアプリケーションで適している動作です。非コミット挿入は、まだ発生していなかのように扱われます。
- 値が OFF (デフォルト) の場合、DB2 サーバーは挿入操作が完了 (コミットまたはロールバック) するまで待機し、その後データを適宜処理します。これは特定のケースで適切です。例えば、
 - 2 つのアプリケーションが表を使用してアプリケーション間でデータを受け渡す (1 番目のアプリケーションが表にデータを挿入し 2 番目のアプリケーションが読み取る) と想定します。示されている順序で 2 番目のアプリケーションによってデータが処理され、読み取る次の行が 1 番目のアプリケーションによって挿入されている場合、2 番目のアプリケーションは、挿入操作がコミットされるまで待機する必要があります。
 - アプリケーションは、データを削除してからデータの新規イメージを挿入して、UPDATE ステートメントを回避します。

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、データベース・マネージャーは場合によっては、行が照会の述部を満たしたことがわかるようになるまで、CS または RS 分離スキャンの行ロックを据え置くことを許可します。

デフォルトでは、表または索引スキャン中に行レベルのロックが実行されると、データベース・マネージャーは、コミットメント状況が不明なスキャンされる行それぞれをロックしてから、行が照会の述部を満たしているかどうかを判別します。

そうしたスキャンの並行性を向上させるには、**DB2_EVALUNCOMMITTED** レジストリー変数を有効にして、非コミット・データで述部評価が生じるようにします。非コミット更新を含む行が照会を満たしていなくても、トランザクションが完了してから述部評価を行うと、行が照会を実際には満たしている場合があります。

DB2_SKIPDELETED レジストリー変数がある場合には、表スキャン時に非コミット削除行がスキップされ、索引スキャン時にはデータベース・マネージャーは削除キーをスキップします。

DB2_EVALUNCOMMITTED レジストリー変数の設定は、動的 SQL または XQuery ステートメントの場合はコンパイル時に適用され、静的 SQL または XQuery ステートメント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デプロイされません。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引き続き有効になります。静的 SQL または XQuery ステートメントの場合は、パッケージを再バインドすると、バインド時に有効なレジストリー変数の設定が適用されます。静的 SQL または XQuery ステートメントを暗黙的に再バインドすると、**DB2_EVALUNCOMMITTED** レジストリー変数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 11. RID 索引単独アクセス

述部	非コミットの評価
なし	いいえ
検索指数述部	はい

表 12. 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	いいえ
検索指数述部	はい

表 13. 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	いいえ	いいえ
なし	検索指数述部	いいえ	いいえ
検索指数述部	なし	はい	いいえ

表 13. 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
検索指数述部	検索指数述部	はい	いいえ

表 14. 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	いいえ	いいえ
なし	検索指数述部	いいえ	はい
検索指数述部	なし	はい	いいえ
検索指数述部	検索指数述部	はい	はい

例

以下の例では、デフォルトのロック動作と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

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) 分離レベルで生じません。

表 15. 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
connect to sample	connect to sample
+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 ステートメントは、表の最初の行に対する排他的ロックを保持し、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 の照会が結果セットを戻さないようにしています。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指定されます。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。

セッション 2 でのロックの待機は、非コミットの評価フィーチャーを使用して回避できます。このフィーチャーは最初に述部を評価し、次いで行をロックします。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せん。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、非コミットの評価フィーチャーで並行性がさらに向上する可能性があります。デフォルトのロック動作では、セッション 2 で最初に行ロックが獲得され、セッション 1 の UPDATE ステートメントがセッション 2 照会でロックされる行を変更しない場合でも、セッション 1 の検索条件付き UPDATE が実行されないようにします。セッション 1 の検索条件付き 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

- **DB2_EVALUNCOMMITTED** レジストリー変数は有効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。
- 行レベルのロックを有効にします。
- 検索指数部評価部が存在します。
- 非コミットの評価はシステム・カタログ表でのスキャンに適用できません。
- マルチディメンション・クラスタリング (MDC) 表または挿入時クラスタリング (ITC) 表の場合は索引スキャンのためにブロック・レベル・ロック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・レベル・ロックを据え置くことはできません。
- ロック据え置きは、インプレース表の再編成を実行している表に対しては発生しません。
- Iscan-Fetch プランの場合は、行レベルのロックはデータ・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。
- 表スキャン時に削除行は無条件にスキップされますが、削除済み索引キーがスキップされるのは **DB2_SKIPDELETED** レジストリー変数が有効な場合だけです。

最適なパフォーマンスを実現する照会の作成とチューニング

DB2 データベースのパフォーマンスに及ぼす SQL ステートメントの影響を、最小限にする方法がいくつかあります。

影響を最小限にするには、以下のようになります。

- DB2 オプティマイザーで最適化しやすい SQL ステートメントを作成します。DB2 オプティマイザーは、非等価結合述部、結合列でのデータ・タイプ不一致、不要な外部結合、および他の複雑な検索条件を含む SQL ステートメントを、効果的に実行できないおそれがあります。
- DB2 データベースを適切に構成して、DB2 最適化機能を駆使できるようにします。正確なカタログ統計を用意し、ワークロードにとって最良の最適化クラスを選ぶなら、DB2 オプティマイザーは最適の照会アクセス・プランを選択できます。
- DB2 Explain 機能を使用して、可能な照会アクセス・プランを検討し、最高のパフォーマンスを得るために照会をチューニングする方法を見極めます。

ベスト・プラクティスは、一般的なワークロード、ウェアハウス・ワークロード、および SAP ワークロードに適用されます。

アプリケーションの作成後に特定の照会パフォーマンス上の問題を扱う方法はいくつもありますが、基本的な作成とチューニングの良い方法を、早い段階で広範囲に適用すると、DB2 データベースのパフォーマンス向上に役立ちます。

照会パフォーマンスを考慮するのは、一回限りではありません。アプリケーション開発ライフ・サイクルの設計、開発、および実動フェーズの全体を通して考慮する必要があります。

SQL は非常に柔軟な言語です。したがって、同じ正しい結果を得るにも数多くの方法があります。また、こうした柔軟性ゆえ、DB2 オプティマイザーの利点を生かす面で、ある照会は他の照会よりも優れていることになります。

照会実行中、DB2 オプティマイザーは、各 SQL ステートメントごとに照会アクセス・プランを選択します。オプティマイザーは、選択可能な多数のアクセス・プランの実行コストをモデル化し、見積コストが最小のプランを選択します。照会中に複雑な検索条件が多数含まれている場合、DB2 オプティマイザーは、述部を書き直せることがあります。しかし、それが不可能なこともあります。

ビジネス・インテリジェンス (BI) アプリケーションなどで使用される複雑な照会の場合、SQL ステートメントの作成やコンパイルにかかる時間が長くなります。使用するデータベースを適切に設計および構成するなら、ステートメントのコンパイル時間を最小限にする助けとなります。この作業には、適切な最適化クラスを選択し、他のレジストリー変数を適切に設定することが含まれます。

またオプティマイザーは、正確にアクセス・プランを決定できるよう、正確な入力データを必要とします。そのためには、正確な統計を収集し、場合によっては統計ビューや列グループ統計などの拡張統計フィーチャーを使用することが必要となります。

DB2 ツール、特に DB2 Explain 機能を使用して、照会をチューニングすることができます。DB2 コンパイラーは、静的または動的照会の環境およびアクセス・プランに関する情報をキャプチャーできます。このキャプチャー情報を使用して、個々のステートメントの実行状況を把握します。これにより、それらの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チューニングして、パフォーマンスを向上できるようになります。

SQL ステートメントの作成

SQL は、構文的には異なっても意味的には等しい方法で関係式を指定できる強力な言語です。しかし、意味的に等しい各種バリエーションの中には、他に比べて最適化が容易なものがあります。DB2 オプティマイザーは強力な照会書き直し機能を備えていますが、SQL ステートメントをいつでも最適な形に書き直せるとは限りません。

ある SQL 構成体は、照会オプティマイザーによって検討されるアクセス・プランを制限するおそれがあり、これらの構造は可能な限り避けるか、または取り替える必要があります。

検索条件での複雑な式の回避:

検索条件で複雑な式を使用しないでください。複雑な式は、オプティマイザーがカタログ統計を使用して正確な選択可能性を見積もる上で妨げとなります。

またそれらの式により、述部の適用に使用するアクセス・プランの選択肢が狭められるおそれもあります。最適化の照会書き直しフェーズの間、オプティマイザーは多数の式を書き直して、オプティマイザーによる正確な選択可能性の見積もりが可能となるようにします。しかし、すべての可能性を扱うことはできません。

式での結合述部の回避:

式または結合述部を使用することにより、使用される結合方式が制限される場合があります。

次に示すタイプの関数がいずれも使用されていない場合、式を使用した結合述部には、ハッシュ結合方式の使用を検討します。

- SQL データの読み取りまたは変更を行う関数
- 非決定性の関数
- 外部アクションを指定する関数
- スクラッチパッドを使用する関数

関連する結合の両方のオペランドからの列を結合条件の同じ側で使用すると、ハッシュ結合も考慮されますが、多くの場合、これは選択できません。

ハッシュ結合を使用できない場合、その代わりにネストしたループ結合が使用されますが、これは処理速度が遅くなる可能性があります。またこのような場合、カーディナリティー推定値が不正確になる可能性があります。

ハッシュ結合の使用が考慮される、式を使用した結合の例を示します。

```
WHERE UPPER(CUST.LASTNAME) = TRANS.NAME
```

次に、ハッシュ結合の利点を活かすのではなく、ネストしたループ結合を使用する式による結合の例を示します。

```
WHERE RAND() > 0.5
```

ローカル述部での列に対する式の回避:

ローカル述部で列に対して式を適用する代わりに、式の逆関数を使用してください。

次の例を考慮してください。

```
XPRESSN(C) = 'constant'  
INTEGER(TRANS_DATE)/100 = 200802
```

これらのステートメントを以下のように書き直すことができます。

```
C = INVERSEXPRESSN('constant')  
TRANS_DATE BETWEEN 20080201 AND 20080229
```

式を列に対して適用するなら、索引開始および停止キーの使用が妨げられ、不正確な選択可能性の見積もりとなり、照会実行時に余分な処理が必要になります。

また、これらの式により照会書き直しの最適化が妨げられます。例えば、列が等しい場合の認識、定数による列の置き換え、および最大で 1 行のみが返されることを

認識することなどが妨げられます。最大で 1 行のみが返されることが証明された後にいっそうの最適化が可能となるため、最適化の機会はますます失われていきます。次の照会を考えてみましょう。

```
SELECT LASTNAME, CUST_ID, CUST_CODE FROM CUST
WHERE (CUST_ID * 100) + INT(CUST_CODE) = 123456 ORDER BY 1,2,3
```

これを以下のように書き直すことができます。

```
SELECT LASTNAME, CUST_ID, CUST_CODE FROM CUST
WHERE CUST_ID = 1234 AND CUST_CODE = '56' ORDER BY 1,2,3
```

CUST_ID で定義されたユニーク索引がある場合、書き直し版の照会において、照会オブティマイザーは、最大で 1 行のみが返されることを認識できます。こうして、不要な SORT 操作が入り込むのを回避できます。また、CUST_ID 列および CUST_CODE 列を 1234 および '56' で置き換えて、データまたは索引ページから値をコピーすることを回避できます。最後に、CUST_ID の述部を索引開始または停止キーとして適用することが可能となります。

述部に式が存在する際、それが明らかではないことがあります。こうしたことは、ビュー列が式によって定義されている場合に、ビューを参照している照会でよく生じます。例えば、次のビュー定義および照会を考慮してください。

```
CREATE VIEW CUST_V AS
  (SELECT LASTNAME, (CUST_ID * 100) + INT(CUST_CODE) AS CUST_KEY
   FROM CUST)
```

```
SELECT LASTNAME FROM CUST_V WHERE CUST_KEY = 123456
```

照会オブティマイザーは、照会をビュー定義とマージします。その結果、以下の照会となります。

```
SELECT LASTNAME FROM CUST
WHERE (CUST_ID * 100) + INT(CUST_CODE) = 123456
```

これは、前述の例と同様の問題がある述部です。 Explain 機能を使用して最適化済みの SQL を表示することにより、ビューのマージ結果を確認できます。

逆関数で表すのが難しい場合は、生成列の使用を考慮してください。例えば、LASTNAME IN ('Woo', 'woo', 'WOO', 'Woo', ...) で表現される基準に合致するラストネームを検索する場合は、生成列 UCASE(LASTNAME) = 'WOO' を以下のように作成できます。

```
CREATE TABLE CUSTOMER (
  LASTNAME VARCHAR(100),
  U_LASTNAME VARCHAR(100) GENERATED ALWAYS AS (UCASE(LASTNAME))
)
```

```
CREATE INDEX CUST_U_LASTNAME ON CUSTOMER(U_LASTNAME)
```

この例に示す DB2 Database for Linux, UNIX, and Windows バージョン 9.5 フィックスパック 1 で導入された大/小文字を区別しない検索のサポートは、こうした状況の解決を意図して設計されたものです。ロケールに依存する UCA ベースの照合に Sx 属性を使用すると、照合の強度を制御できます。例えば、属性 LFR_S1 を持つロケールに依存する UCA ベースの照合は、大/小文字およびアクセント符号を無視するフランス語照合です。

述部でノーオペレーション式を使用して 옵ティマイザーの見積もりを変化させることを避ける:

COALESCE(X, X) = X 形式の「ノーオペレーション」coalesce() 述部があると、これを使用するいずれの照会のプランでも見積もりエラーが発生します。今のところ、DB2 照会コンパイラーは、こうした述部を分析し、実際にはそれがすべての行によって満たされることを判断できません。

結果として、その述部により、照会プランのある部分から来る行の見積もり行数が不自然に削減されます。通常、この少なくされた行見積もりにより、照会プランの残りの部分における行およびコストの見積もりも削減されます。そして、各種候補プラン間の相対見積もりが変化したために、異なるプランが選択される結果となります。

この何もしない述部によって照会パフォーマンスが向上することがあるのはなぜでしょうか。「ノーオペレーション」coalesce() 述部を追加すると発生するエラーにより、最適なパフォーマンスを妨げている別のある部分がマスクされます。

パフォーマンス向上ツールの中には、力づくのテストを実行するものがあります。つまりそれらのツールは、照会の異なる箇所にその述部を繰り返し挿入して各列を操作することにより、エラーを入り込ませてより優れたパフォーマンス・プランに偶然行き着くケースを探ろうとします。これは、照会内に「ノーオペレーション」述部をハンド・コーディングする照会の開発者にも当てはまります。開発者は普通、データに関して先見性を持ち、述部の配置に役立てるようにします。

照会パフォーマンスの向上を図るこの方法は、短期的な解決策であって根本原因を扱いませんし、以下の影響が懸念されます。

- パフォーマンスの改善の可能性がある領域が隠蔽されます。
- この回避策によって、パフォーマンスの改善が恒久的に得られる保証はありません。なぜなら、結局は DB2 照会コンパイラーによる述部の扱いが向上する可能性がありますし、他の偶発的な要因に影響される可能性もあるからです。
- 同じ根本原因の影響を受ける他の照会が存在する可能性があり、結果としてシステム全体のパフォーマンスが低下するおそれがあります。

ベスト・プラクティス推奨事項に従っても、依然として最適なパフォーマンスが得られていないようなら、「ノーオペレーション」述部を導入するよりも、明示的な最適化ガイドラインを DB2 オプティマイザーに適用できます。『最適化プロファイルと最適化ガイドライン』を参照してください。

非等価結合述部の回避:

結合方式がネスト・ループに制限されているため、等価以外の比較演算子を使用する結合述部は避けてください。

さらに、オプティマイザーが、結合述部について正確な選択可能性の見積もりを計算できないおそれがあります。しかし、非等価結合述部を避けられない場合もあります。それらが必要な場合、必ず両方の表に適切な索引があるようにしてください。なぜなら、結合述部はネスト・ループ結合の内部表に適用されるからです。

非等価結合述部の一般的な例の 1 つは、スター・スキーマ内のディメンション・データをバージョン付けして、異なる各時点でのディメンションの状態を正確に表す必要があるケースです。これをスロー・チェンジ・ディメンションと呼ぶことがあります。スロー・チェンジ・ディメンションのタイプの 1 つは、各ディメンション行の有効な開始および終了日付を含めることに関するものです。ファクト表とディメンション表の結合では、ディメンションの主キーにおける結合に加えて、ファクトに関連付けられた日付が、ディメンションの開始および終了日付の範囲内であることを検査する必要があります。これをタイプ 6 のスロー・チェンジ・ディメンションと呼ぶことがあります。何らかのファクト・トランザクション日付でディメンション・バージョンをさらに修飾するための、ファクト表への再範囲結合は、コストが大きい可能性があります。例えば、

```
SELECT...
  FROM PRODUCT P, SALES F
  WHERE
    P.PROD_KEY = F.PROD_KEY AND
    F.SALE_DATE BETWEEN P.START_DATE AND
    P.END_DATE
```

この場合、(SALES.PROD_KEY, SALES.SALE_DATE) の索引があるようにしてください。

このシナリオにおいて、オプティマイザーがより優れた選択可能性の見積もりを計算する助けとなる、統計ビューの作成を考慮します。例えば、

```
CREATE STATISTICAL VIEW V_PROD_FACT AS
  SELECT P.*
  FROM PRODUCT P, SALES F
  WHERE
    P.PROD_KEY = F.PROD_KEY AND
    F.SALE_DATE BETWEEN P.START_DATE AND
    P.END_DATE

ALTER VIEW V_PROD_FACT ENABLE QUERY OPTIMIZATION

RUNSTATS ON TABLE DB2USER.V_PROD_FACT WITH DISTRIBUTION
```

照会ブロックに非等価結合述部がある場合、特殊なスター・スキーマ結合 (索引 ANDing によるスター型結合およびハブ型結合など) は考慮されません。(『照会がスター・スキーマ結合の必須基準に適合していることを確認する』を参照してください。)

不要な外部結合の回避:

照会のセマンティクスの中には、(左、右、または全) 外部結合が必要なものがあります。しかし、照会のセマンティクスで外部結合が不要で、矛盾するデータの処理に照会が使用されている場合は、矛盾するデータの問題に根本原因から対処するのが最善でしょう。

例えばスター・スキーマによるデータマートにおいて、データ整合性の問題のために、ファクト表には、トランザクション用の行が含まれていても、いくつかのディメンションについてマッチングする親ディメンション行が含まれていない可能性があります。これが生じることがあるのは、何らかの理由により、抽出、トランスフォーム、およびロード (ETL) 処理でいくつかのビジネス・キーを両立させることが

できないためです。このシナリオにおいて、ファクト表の行はディメンションと左外部結合されて、親がない場合でも必ずそれらが返されるようになっています。例えば、

```
SELECT...
FROM DAILY_SALES F
LEFT OUTER JOIN CUSTOMER C ON F.CUST_KEY = C.CUST_KEY
LEFT OUTER JOIN STORE S ON F.STORE_KEY = S.STORE_KEY
WHERE
C.CUST_NAME = 'SMITH'
```

この左外部結合により、特殊スター・スキーマ結合アクセス方式を含め、数多くの最適化処理が妨げられるおそれがあります。しかし場合によっては、照会最適マイザーによって、左外部結合から内部結合に自動的に書き直されることがあります。この例では、CUSTOMER と DAILY_SALES の左外部結合から、内部結合に変換される可能性があります。なぜなら、述部 C.CUST_NAME = 'SMITH' によって、この列の NULL 値を持つ行がすべて除去されるため、左外部結合は意味的に不要となるからです。それで、外部結合があるためにいくつかの最適化処理が失われても、すべての照会に悪影響が及ぶとは限りません。しかし、これらの制約に通じておき、絶対に必要とされているのでない限り、外部結合を使わないようにすることが大切です。

FETCH FIRST N ROWS ONLY 節と共に OPTIMIZE FOR N ROWS 節を使用する:

OPTIMIZE FOR *n* ROWS 節を使用すると、アプリケーションが *n* 行のみを取得するよう、最適マイザーに対して指示できます。しかし、照会では完全な結果セットが戻ります。FETCH FIRST *n* ROWS ONLY 節を使用すると、照会で *n* 行のみが戻るように指示できます。

DB2 データ・サーバーは、外部副選択のために FETCH FIRST *n* ROWS ONLY が指定されている場合に、自動的に OPTIMIZE FOR *n* ROWS の指定を想定しません。FETCH FIRST *n* ROWS と共に OPTIMIZE FOR *n* ROWS を指定するようにしてください。こうすると、バッファリング操作 (一時表への挿入、ソート、またはハッシュ結合のハッシュ表への挿入など) を最初に実行せずに、参照表から行を直接返す照会アクセス・プランが促進されます。

アプリケーションが、OPTIMIZE FOR *n* ROWS を指定してバッファリング運用を避ける照会アクセス・プランを促進していても、結果セット全体を取得するのであれば、パフォーマンスが低下するおそれがあります。なぜなら、結果セット全体が取得されている場合、最初の *n* 行を最も速く返す照会アクセス・プランが、最良の照会アクセス・プランとは限らないためです。

照会がスター・スキーマ結合の必須基準に適合していることを確認する:

最適マイザーは、スター・スキーマに基づく照会に、スター型結合、デカルト・ハブ結合、およびジグザグ結合という 3 つの特殊な結合方式を考慮します。これらの結合方式により、この種の照会のパフォーマンスを大幅に改善できます。

照会が、ジグザグ結合プラン、スター型結合プラン、またはデカルト・ハブ結合プランを行うためのスター・スキーマとして認識されるためには、以下の基準を満たす必要があります。

- 1 つのファクト表と 2 つ以上のディメンション表を持つスター型照会でなければならない。この照会に、共通の関連ディメンション表を持つ 2 つ以上のファクト表が含まれている (複数ファクト表照会) 場合、照会オプティマイザーはこの照会を、1 つの照会とその内部にある複数のスターに分割します。すると、複数のファクト表と結合する共通ディメンション表は、照会の中で複数回使用されます。Explain の出力では、これら複数のファクト表の照会に対する複数のジグザグ結合演算子が表示されます。
- ディメンション表には主キー、ユニーク制約、またはユニーク索引が定義されている必要があります。主キーは複合キーであっても構いません。ディメンション表に主キー、ユニーク制約、ユニーク索引がない場合には、デカルト・ハブ結合およびスター型結合方式のスターを検出するために、古いスター検出方式が使用されます。この場合、デカルト・ハブ結合とスター型結合は、197 ページの『デカルト・ハブ結合とスター型結合の代替基準』で説明されている基準を満たす必要があります。
- ディメンション表とファクト表は、ディメンション表の主キーの一部であるすべての列に equijoin 述部を使用して結合されなければなりません。
- デカルト・ハブ結合とジグザグ結合では、ファクト表に複数列索引が必要です。結合に含まれる列はその索引の一部であり、2 つ以上のディメンション表を対象とするファクト表から十分な結合列を持っている必要があります。
- デカルト・ハブ結合およびスター型索引 ANDing 結合の場合、ディメンション表またはスノーフレイクはファクト表をフィルタリングする必要があります。(フィルタリングは、オプティマイザーの見積もりに基づいて行われます。) また、ディメンション表が、フィルターとしてではなく単にルックアップ・タイプの結合としてファクト表に結合される場合でも、スター型結合が発生することがあります。

例えば、D1、D2、D3 という 3 つのディメンション表があるとします。ディメンション表 D1 は主キー A を持ち、列 A でファクト表と結合しています。ディメンション表 D2 は主キー (B,C) を持ち、列 B および列 C でファクト表と結合しています。最後に、ディメンション表 D3 は主キー D を持ち、列 D でファクト表と結合しています。サポートされる索引の使用法は以下のとおりです。

- (A,D)、(A,B,C)、または (C,B,D) という索引はそれぞれ 2 つ以上のディメンション表を対象としているため、これらの索引はいずれか 1 つで十分です。
- 索引 (A,B,C,D) も 3 つのディメンション表を対象としているため適切です。
- 索引 (A,B) はディメンション表 D2 を完全には対象としていないため使用できません。
- 索引 (B,A,C) は D2 という主キーと結合する列 B および C が索引内で隣接する位置ではないため、使用できません。

以下のいずれかが出現する場合、ディメンション表は、ジグザグ結合、スター型結合、デカルト・ハブ結合のいずれの結合方式にも含めることができません。

- サンプリング
- 非決定性関数
- 副次作用がある関数

また、ディメンション表からのキーの重複を招くスノーフレイク内でのジグザグ結合はサポートされません。

前述の特性を持たないディメンション表は、オプティマイザーがコスト分析を使用して、ジグザグ結合またはスター型結合に含める対象として考慮します。前述の特性を 1 つ以上持つディメンション表は、他の結合方式を使用して結合できます。

ネストしたジグザグ結合はサポートされていません。これは、ジグザグ結合プランは別のジグザグ結合プランに対するディメンション/スノーフレイクにはなれないということです。同様に、ネストしたスター型結合はサポートされていません。これは、スター型結合プランは別のスター型結合プランに対するディメンション/スノーフレイクにはなれないということです。さらに、ジグザグ結合プランとスター型結合プランは互いの中にネストすることはできません。

フェデレーテッド・ニックネームはジグザグ結合やスター型結合からは除外されません。ニックネームは、他の結合方式を使用して結合できます。スター全体がリモート DB2 データ・サーバーにプッシュダウンされる場合、このリモート・データ・サーバーはジグザグ結合またはスター型結合方式を使用して照会を実行します。

デカルト・ハブ結合とスター型結合の代替基準

ディメンション表に主キー、ユニーク制約、またはユニーク索引のいずれも定義されていない場合、照会がデカルト・ハブ結合またはスター型結合のためのスター・スキーマとして認識されるためには、以下の基準を満たす必要があります。

- 各照会ブロックごとに
 - 少なくとも 3 つの異なる表が結合されていること
 - 結合述部がすべて等価述部であること
 - 副照会が存在しないこと
 - 表間または照会ブロック外に相関や従属関係が存在しないこと
 - ファクト表について
 - 照会ブロック内で最大の表であること
 - 少なくとも 10,000 行あること
 - 基本表であること
 - 少なくとも 2 つのディメンション表、またはスノーフレイクという 2 つのグループに結合されていること
 - ディメンション表について
 - ファクト表ではないこと
 - 個別にファクト表に結合、またはスノーフレイク形式で結合可能
 - ディメンション表またはスノーフレイクについて
 - ファクト表をフィルタリングすること。(フィルタリングは、オプティマイザーの見積もりに基づいて行われます。)
 - ファクト表索引の列を使用するファクト表への結合述部があること。スター型結合またはハブ型結合を検討させるには、この基準が満たされなければなりません。とはいえ、ハブ型結合では単一のファクト表索引を使用する必要があるのみです。

左または右外部結合を表す照会ブロックは、2 つの表のみを参照できます。したがって、スター・スキーマ結合は適格ではありません。

옵ティマイザーがスター・スキーマ結合を認識するのに、参照整合性を明示的に宣言することは不要です。

照会をジグザグ結合に必要な基準に適合させる:

照会 옵ティマイザーは、表と照会がジグザグ結合の前提条件に適合していると判断し、かつこの方式によりパフォーマンスが改善される場合に、ジグザグ結合方式を使用します。

このタスクについて

表と照会がジグザグ結合に必要な前提条件に必ず適合するようにするために、このタスクを使用します。さらにこのタスクを使用すると、옵ティマイザーの選択では不十分な場合に、ジグザグ結合の使用に手を加えることができます。

ジグザグ結合方式は、デカルト積を実際にはマテリアライズせず、ディメンション表から行のデカルト積を計算し、複数列索引を使用してファクト表を厳密に調べるため、ファクト表は 2 つ以上のディメンション表を使って同時にフィルターされます。ファクト表を厳密に調べると、一致する行が見つかります。その後、ジグザグ結合によって、ファクト表の索引から次に利用できる値の組み合わせが返されます。次に使用できるこの値の組み合わせはフィードバックと呼ばれ、ファクト表で一致が検出されないディメンション表のデカルト積を使って求められるプローブ値をスキップオーバーするために使用されます。2 つ以上のディメンション表でファクト表を同時にフィルターすると共に、非生産的であると分かっているプローブをスキップすることにより、ジグザグ結合が大きなファクト表の照会に対して効率的な方法となります。

手順

1. ジグザグ結合に含まれる表が、必要な基準に適合していることを確認します。各ディメンション表には主キー、ユニーク制約、またはユニーク索引が定義されている必要があります。主キー、ユニーク制約、またはユニーク索引を定義するには、次の例のようなコマンドを使用します。

```
-- defining a unique constraint on a dimension table using composite keys
create table dim1 (
  d0 int not null,
  d1 int not null,
  c1 int,
  constraint pk1_uniq unique (d0, d1)
);

-- defining a primary key on a dimension table
create table dim2 (
  d2 int primary key not null,
  c2 int
);

-- creating a unique index on a dimension table
create table dim3 (
  d3 int,
  c3 int
);
create unique index uniq_ind on dim3(d3);
```

2. 適切な照会を作成します。この照会では、各ディメンション表の主キー、ユニーク索引、ユニーク制約のいずれかと、ファクト表の列との間に、等価結合述部が必要です。以下に例を示します。

```
select count(*)
  from dim1,dim2,dim3,fact
  where dim1.d0 = fact.f0
        and dim1.d1 = fact.f1
        and dim2.d2 = fact.f2
        and dim3.d3 = fact.f3
        and dim1.c1 = 10
        and dim2.c2 < 20;
```

3. ファクト表に、適切な複数列索引があることを確認します。複数列索引には、ディメンション表列との結合述部を持つファクト表からの列が含まれている必要があります。このような複数列索引を定義するには、次のようなコマンドを使用します。

```
create index fact_ind on fact (f0, f1, f2, f3);
```

適切な複数列索引がない場合、**db2exfmt** コマンドの出力に診断メッセージが通知として表示されます。

4. 照会を EXPLAIN モードで実行してから **db2exfmt** コマンドを実行し、EXPLAIN 出力をフォーマットします。出力を調べて、ジグザグ結合が使用されたかどうか、および所定のパフォーマンスが達成されたかどうかを判別します。
5. オプション: ジグザグ結合方式が使用されなかった場合、または所定のパフォーマンスが達成されなかった場合、別の複数列索引を作成する必要があります。**db2exfmt** コマンド出力の拡張診断情報 (Extended Diagnostic Information) セクションを確認します。この出力にエラー・メッセージがリストされている場合は、提案 (例えば新しい索引を生成するなど) に従います。
6. オプション: 所定のパフォーマンスが達成されなかった場合は、索引にギャップがあるかどうか判別します。**db2exfmt** 出力のギャップ情報 (Gap Info) セクションを確認します。

```
Gap Info:                Status
-----
Index Column 0:         No Gap
Index Column 1:         Positioning Gap
Index Column 2:         No Gap
Index Column 3:         No Gap
Index Column 4:         No Gap
```

このセクションに、複合索引と整合していない述部が照会の中に含まれているという出力がある場合は、索引ギャップを避けるために、新しい索引を追加するか、既存の索引を変更することを検討してください。

ジグザグ結合アクセス・プランの例:

次の例は、ジグザグ結合で可能な各種アクセス・プランでの **db2exfmt** コマンド出力を示しています。

これらの例では、DAILY_SALES のスター型照会をファクト表として、また CUSTOMER と PERIOD をディメンション表として使用します。この照会では、1996年3月に年齢レベル7の顧客に販売された製品の合計数量を照会しています。その結果は、顧客の収入レベルの記述ごとに集計されます。

```
select income_level_desc, sum(quantity_sold) "Quantity"
  from daily_sales s, customer c, period p
  where calendar_date between '1996-03-01' and '1996-03-31'
```

```

and p.perkey = s.perkey
and s.custkey = c.custkey
and age_level = 7
group by income_level_desc;

```

ジグザグ結合では、以下の 3 つのタイプのファクト表アクセス・プランが可能です。

- **索引スキャン・フェッチ・プラン:** このプランでは、索引スキャンがファクト表の索引にアクセスし、ファクト表から入力プローブ値と一致する RID を取り出します。これらのファクト表の RID は、必要なファクト表データをファクト表から取り出すために使用されます。次に、ディメンション表のペイロード列がディメンション表から取り出されて、ジグザグ結合演算子により結果行が出力されます。
- **単一プローブ・リスト・プリフェッチ・プラン:** このプランでは、ディメンション表とスノーフレイクの組み合わせから得られるすべてのプローブ行に対して、リスト・プリフェッチ・プランが実行されます。ファクト表での索引スキャンにより、入力プローブ値と一致するファクト表の RID が検索されます。**SORT**、**RIDSCAN**、および **FETCH** の演算子はデータ・ページ ID に従って RID をソートし、リスト・プリフェッチャーはファクト表データの取得を開始します。次に、ディメンション表のペイロード列がディメンション表から取り出されて、ジグザグ結合演算子により結果行が出力されます。
- **全プローブ・リスト・プリフェッチ・プラン:** このプランでは、ディメンション表とスノーフレイクの組み合わせから得られるすべてのプローブ行について、索引スキャンがファクト表索引にアクセスします。このようにして一致した RID はすべてまとめてファクト表のデータ・ページの順序にソートされ、リスト・プリフェッチャーはファクト表から必要なデータの取り出しを開始します。すべての RID をこの方式でソートすると、プリフェッチがより効率的に行われます。これらの照会には、2 つの別個の **ZZJOIN** 演算子が含まれます。その 1 つは、ファクト表とディメンション表の間の逆方向結合を表します。

例: ファクト表の索引スキャン・フェッチ・アクセス

```

                2.6623e+06
                ZZJOIN
                ( 5)
                7620.42
                5.37556
+-----+
292.2          40000          0.227781
TBSCAN        TBSCAN        FETCH
( 6)          ( 9)          ( 13)
56.2251       7596.78       11.8222
1             2.92          1.22778
|             |             /-----\
292.2          40000          0.227781   6.65576e+08
TEMP          TEMP          IXSCAN   TABLE: POPS
( 7)          ( 10)         ( 14)   DAILY_SALES
30.4233       4235.52       9.93701   Q3
1             2.92          1
|             |             |
292.2          40000          6.65576e+08
IXSCAN        FETCH        INDEX: POPS
( 8)          ( 11)       PER_CUST_ST_PROMO
29.9655       4235.07       Q3
1             2.92
|             /-----\
2922          40000          1e+06
INDEX: POPS   IXSCAN   TABLE: POPS

```

```

PERX1      ( 12)      CUSTOMER
Q1         2763.52    Q2
           |
           1
           |
           1e+06
INDEX: POPS
CUSTX1
Q2

```

TBSCAN(6) 演算子と TBSCAN(9) 演算子は、以下の情報を示します。

```

IS_TEMP_INDEX : True/False
スキャンによって、temp のランダム・アクセス用の索引が temp 上に作成されます。
(フラグが 'true' の場合)
スキャンによって、temp のランダム・アクセス用の高速整数ソート構造が作成されます。
(フラグが 'false' の場合)

```

TBSCAN(6) 演算子と TBSCAN(9) 演算子は、これら演算子に適用されるフィードバック述部に関する情報を、スタート・ストップ・キー条件の形で示します。

Predicates:

```

-----
5) Start Key Predicate,
Comparison Operator: Equal (=)
Subquery Input Required: No
Filter Factor: 0.000342231

```

```

Predicate Text:
-----
(Q1.PERKEY = Q3.PERKEY)

```

```

5) Stop Key Predicate,
Comparison Operator: Equal (=)
Subquery Input Required: No
Filter Factor: 0.000342231

```

```

Predicate Text:
-----
(Q1.PERKEY = Q3.PERKEY)

```

ZZJOIN(5) 演算子は、ジグザグ結合の処理で使用されるすべてのフィードバック述部の集合を示します。

```

Predicates:
-----
4) Feedback Predicate used in Join,
Comparison Operator: Equal (=)
Subquery Input Required: No
Filter Factor: 1e-06

```

```

Predicate Text:
-----
(Q3.CUSTKEY = Q2.CUSTKEY)

```

```

5) Feedback Predicate used in Join,
Comparison Operator: Equal (=)
Subquery Input Required: No
Filter Factor: 0.000342231

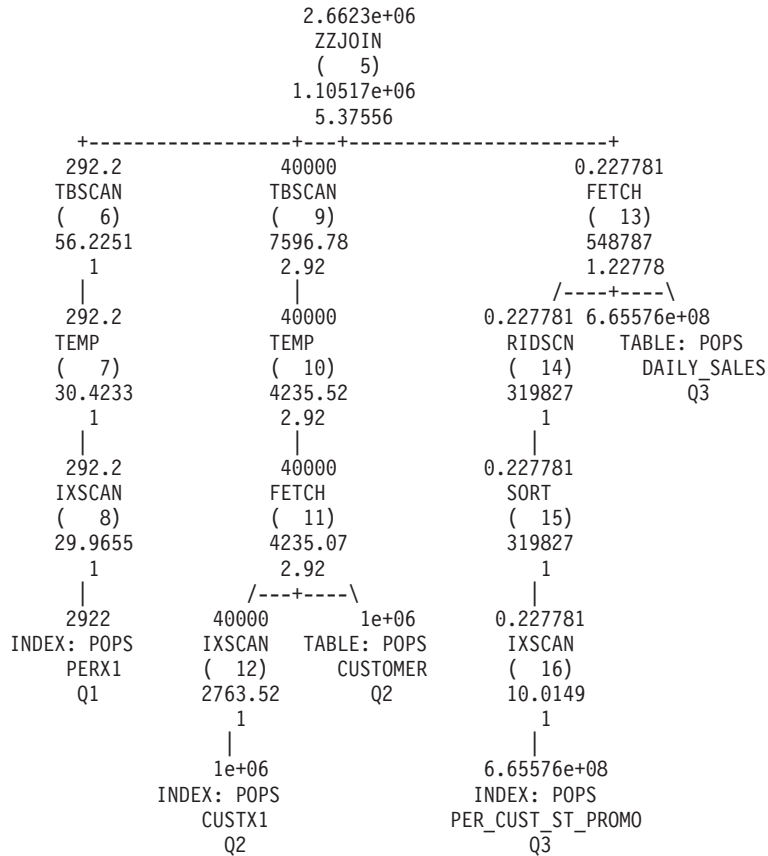
```

```

Predicate Text:
-----
(Q1.PERKEY = Q3.PERKEY)

```

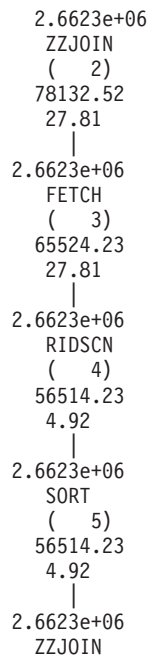
例: ファクト表の単一プローブ・リスト・プリフェッチ・アクセス



これで、索引スキャン・プランと単一プローブ・プランの違いが、ファクト表へのアクセス方法であることがわかります。

他の演算子はすべて、前述の例における演算子と同じ情報を表示します。

例: ファクト表の全プローブ・リスト・プリフェッチ・アクセス



```

              ( 6)
            7616.65
              4.92
+-----+-----+
292.2          40000          0.227781
TBSCAN        TBSCAN        IXSCAN
( 7)          ( 10)         ( 14)
56.2251       7596.78       9.93701
  1           2.92          1
  |           |             |
  292.2       40000         6.65576e+08
TEMP          TEMP          INDEX: POPS
( 8)          ( 11)         PER_CUST_ST_PROMO
30.4233       4235.52       Q3
  1           2.92
  |           |
  292.2       40000
IXSCAN        FETCH
( 9)          ( 12)
29.9655       4235.07
  1           2.92
  |           /-----\
  2922        40000        1e+06
INDEX: POPS   IXSCAN      TABLE: POPS
PERX1         ( 13)       CUSTOMER
Q1            2763.52     Q2
              1
              |
              1e+06
              INDEX: POPS
              CUSTX1
              Q2

```

他のアクセス・プランと比較すると、全プローブ・リスト・プリフェッチ・プランでは加算演算子 ZZJOIN (2) が表示されます。この演算子は、ファクト表とディメンション表との逆方向結合を実行するために使用されています。以下の情報が表示されます。

Backjoin = True

索引ギャップのあるジグザグ結合:

照会で結合述部が欠落しているか、ディメンションが欠落しているために複合索引にギャップがある場合でも、照会オプティマイザーは照会のスター型形状を認識して、ジグザグ結合を使用した適切なアクセス・プランを選択することができます。しかし、アクセス・プランのパフォーマンスが最適にならない可能性があります。**db2exfmt** コマンド出力を検討して、照会の中に索引ギャップがあるかどうかを調べ、パフォーマンスを改善するために新しい索引を追加するか、既存の索引を変更することを検討します。

例: ディメンションが欠落しているジグザグ結合

この例は次の照会に基づくものです。d1、d2、d3、d4、および d5 はディメンション、f1 はファクト表です。

```

select count(*)
from d1, d3, d4, d5, f1
where d1.pk = f1.fk1 and d3.pk = f1.fk3 and d4.pk = f1.fk4 and d5.pk = f1.fk5

```

ファクト表 f1 の索引は、次のコマンドを使用して作成されたものです。

```

create index i11 on f1(fk1, fk2, fk3, fk4, fk5, fk6);

```

この照会は、ディメンション d1、d3、d4、d5 をファクト表 f1 と結合します。ディメンション d2 は照会に含まれていないため、列 fk2 にはディメンション d2 との結合述部はありません。照会オプティマイザーはファクト列 fk2 を索引内のギャップとして認識し、この索引をジグザグ結合に使用できます。

db2exfmt コマンド出力では、JUMPSCAN=TRUE オプションを指示することにより、索引スキャンがジャンプ・スキャンであることが示されます。またこの出力では索引ギャップ情報も示されます。具体的には、2 番目の索引列に位置指定上のギャップがあり、他の列にはないということが示されます。

```

Rows
RETURN
( 1)
Cost
I/O
|
1
GRPBY
( 2)
1539.45
33
|
1000
ZZJOIN
( 3)
1529.44
33

+-----+-----+-----+-----+
1000      1000      1000      1000      1000
TBSCAN    TBSCAN    TBSCAN    TBSCAN    FETCH
( 4)      ( 9)      ( 14)     ( 19)     ( 24)
184.085   184.085   184.085   184.085   205.222
8         8         8         8         1
|         |         |         |         /-----\
1000      1000      1000      1000      1000      1000
TEMP      TEMP      TEMP      TEMP      RIDSCN    TABLE: STAR
( 5)      ( 10)     ( 15)     ( 20)     ( 25)     F1
184.003   184.003   184.003   184.003   55.5857   Q1
8         8         8         8         1
|         |         |         |         |
1000      1000      1000      1000      1000
TBSCAN    TBSCAN    TBSCAN    TBSCAN    SORT
( 6)      ( 11)     ( 16)     ( 21)     ( 26)
178.62    178.62    178.62    178.62    55.5342
8         8         8         8         1
|         |         |         |         |
1000      1000      1000      1000      1e-09
SORT      SORT      SORT      SORT      IXSCAN
( 7)      ( 12)     ( 17)     ( 22)     ( 27)
178.569   178.569   178.569   178.569   12.0497
8         8         8         8         1
|         |         |         |         |
1000      1000      1000      1000      1000
TBSCAN    TBSCAN    TBSCAN    TBSCAN    INDEX: STAR
( 8)      ( 13)     ( 18)     ( 23)     I11
135.093   135.093   135.093   135.093   Q1
8         8         8         8
|         |         |         |
1000      1000      1000      1000
TABLE: STAR  TABLE: STAR  TABLE: STAR  TABLE: STAR
D1          D3          D4          D5
Q5          Q4          Q3          Q2

JUMPSCAN: (Jump Scan Plan)
TRUE

Gap Info:          Status
-----

```



```

Index Column 0:    No Gap
Index Column 1:    Positioning Gap
Index Column 2:    No Gap
Index Column 3:    No Gap
Index Column 4:    No Gap

```

例: ファクト列の結合述部が欠落しているジグザグ結合

この例は、以下の照会に基づくものです。

```

select count(*)
from d2, d3, d4, f1
where d2.pk = f1.fk2 and d3.pk = f1.fk3 and d4.pk = f1.fk4 and fk1=10

```

この照会では、ディメンション d2、d3 および d4 がファクト表 f1 と結合します。ファクト列 fk1 には結合述部がなく、ローカル述部 fk1=10 しかありません。

照会オプティマイザはファクト列 fk1 に結合述部がないため、この列をギャップとして認識します。それでも、照会オプティマイザはジグザグ結合にこの索引を使用できます。

db2exfmt コマンド出力では、JUMPSCAN=TRUE オプションを指示することにより、索引スキャンがジャンプ・スキャンであることが示されます。またこの出力では索引ギャップ情報も示されます。具体的には、最初の索引列に位置指定上のギャップがあり、他の列にはないということが示されます。

```

      Rows
      RETURN
      (  1)
      Cost
      I/O
      |
      1
      GRPBY
      (  2)
      893.899
      25.12
      |
      40
      HSJOIN
      (  3)
      893.489
      25.12
      /-----+-----\
      1000                                40
      TBSCAN                                ZZJOIN
      (  4)                                (  5)
      135.093                              750.88
      8                                    17.12
      |
      1000                                40
      TABLE: STAR  TBSCAN                TBSCAN                FETCH
      D4              (  6)                ( 11)                ( 16)
      Q2              184.085              184.085              18.1845
      |              8                    8                    1.12004
      1000              |                  |                  /-----\
      TEMP              1000              1000              40              1000
      (  7)              TEMP              ( 12)              ( 17)              TABLE: STAR
      184.003            184.003            13.4358            13.4358            F1
      8                  |                  |                  1.12              Q1
      |                  |                  |
      1000              1000              40
      TBSCAN              TBSCAN              SORT

```

```

      ( 8)      ( 13)      ( 18)
      178.62    178.62    13.3843
        8         8         1.12
        |         |         |
        1000      1000      4e-05
      SORT      SORT      IXSCAN
      ( 9)      ( 14)      ( 19)
      178.569    178.569    12.5738
        8         8         1.12
        |         |         |
        1000      1000      1000
      TBSCAN    TBSCAN    INDEX: STAR
      ( 10)     ( 15)     I11
      135.093    135.093    Q1
        8         8
        |         |
        1000      1000
TABLE: STAR    TABLE: STAR
      D2         D3
      Q4         Q3

```

JUMPSCAN: (Jump Scan Plan)
TRUE

Gap Info:	Status
-----	-----
Index Column 0:	Positioning Gap
Index Column 1:	No Gap
Index Column 2:	No Gap

冗長な述部の回避:

冗長な述部を (とりわけ異なる表にまたがる場合に) 避けてください。場合によっては、オプティマイザは述部が冗長であることを検出できません。その結果、カーディナリティーの過小評価につながる可能性があります。

例えば、SAP ビジネス・インテリジェンス (BI) アプリケーション内で、ファクト表およびディメンション表を伴うスノーflake・スキーマを、照会最適化データ構造として使用するとします。ファクト表およびディメンション表で、冗長な時間特性列 (月について "SID_0CALMONTH" または年について "SID_0FISCPER") が定義されている場合があります。

SAP BI の OLAP プロセッサは、ディメンション表およびファクト表の時間特性列に関する冗長な述部を生成します。

これらの冗長な述部によって、照会実行時間が長くなる可能性があります。

以下は、SAP BI 照会の WHERE 条件で定義された、2 つの冗長な述部を含む例です。時間ディメンション (DT) およびファクト (F) 表について、同一の述部が定義されています。

```

      AND (
        "DT"."SID_0CALMONTH" = 199605
        AND "F"."SID_0CALMONTH" = 199605
        OR "DT"."SID_0CALMONTH" = 199705
        AND "F"."SID_0CALMONTH" = 199705 )
      AND NOT (
        "DT"."SID_0CALMONTH" = 199803
        AND "F"."SID_0CALMONTH" = 199803 )

```

DB2 オプティマイザーは、述部を同一のものとして認識せず、それぞれ独立したものとして扱います。これは、カーディナリティーの過小評価、最適ではない照会アクセス・プラン、および照会実行時間の増加につながります。

こうした理由で、冗長な述部は、DB2 データベース・プラットフォーム固有のソフトウェア層によって除去されます。

これらの述部は、以下のように書き換えられ、ファクト表の列「SID_0CALMONTH」の述部のみが残ります。

```
AND ( "F". "SID_0CALMONTH" = 199605
      OR "F". "SID_0CALMONTH" = 199705 )
AND NOT ( "F". "SID_0CALMONTH" = 199803 )
```

SAP ノート 957070 および 1144883 にある指示を適用して、冗長な述部を除去してください。

照会の最適化を改善するための制約の使用

ユニーク制約、チェック制約、および参照整合性制約を定義することを検討します。これらの制約によってセマンティック情報が提供されます。この情報により、DB2 オプティマイザーが、結合の除去、結合による集約のプッシュダウン、結合による FETCH FIRST *n* ROWS のプッシュダウン、不要な DISTINCT 操作の除去、および他の多くの最適化処理を行うために、照会を書き直すことが可能となります。

また、アプリケーション自体がリレーションシップを保証できる場合、インフォメーションナル制約をチェック制約および参照整合性制約の両方に使用できます。同様の最適化が可能です。行の挿入、更新、または削除時にデータベース・マネージャーが制約を施行すると、システムのオーバーヘッドが増加する場合があります。参照整合性制約を含む多数の行を更新する場合は、特にそう言えます。行の更新前に情報の検証がアプリケーションによって既に行われている場合は、通常の制約ではなく、インフォメーションナル制約を使用の方がより効率的かもしれません。このタイプのインフォメーションナル制約は、NOT ENFORCED TRUSTED 制約とも呼ばれます。

例として、2 つの表、DAILY_SALES および CUSTOMER を考慮してみましょう。CUSTOMER 表の各行には、ユニークなカスタマー・キー (CUST_KEY) があります。DAILY_SALES には CUST_KEY 列が含まれており、各行は CUSTOMER 表内のカスタマー・キーを参照しています。この CUSTOMER と DAILY_SALES 間の 1:N リレーションシップを表すために、参照整合性制約を作成できるかもしれません。アプリケーションがこのリレーションシップを施行する場合、インフォメーションナル制約として定義できます。すると、以下の照会では、CUSTOMER および DAILY_SALES 間の結合の実行を回避できます。なぜなら、CUSTOMER からは列を取得せず、DAILY_SALES のいずれの行についても、CUSTOMER 内に一致するものがあるからです。照会オプティマイザーは、自動的に結合を除去します。

```
SELECT AMT_SOLD, SALE PRICE, PROD_DESC
FROM DAILY_SALES, PRODUCT, CUSTOMER
WHERE
  DAILY_SALES.PROD_KEY = PRODUCT.PRODKEY AND
  DAILY_SALES.CUST_KEY = CUSTOMER.CUST_KEY
```

アプリケーションは、インフォメーション制約を施行しなければなりません。そうでないと、照会によって不正確な結果が返されるおそれがあります。この例の場合、CUSTOMER 表内の対応するカスタマー・キーを含まない行が、DAILY_SALES 内にある場合、照会によってこれらの行が誤って返されます。

もう 1 つのタイプのインフォメーション制約は、NOT ENFORCED NOT TRUSTED 制約です。表の行が制約に適合しているかどうかをアプリケーションで検証できない場合、このタイプのインフォメーション制約を指定すると便利です。DB2 オプティマイザーがデータを使用して統計ビューから統計を推定できる場合、NOT ENFORCED NOT TRUSTED 制約を使用して、照会の最適化を改善することができます。このような場合、外部キーの値と主キーの値を厳密に突き合わせる必要はありません。制約が NOT TRUSTED であり、照会の最適化に使用できるようになっている場合、これを使用して、制約に完全適合するデータに依存した最適化 (結合の除去など) を実行することはできません。

複雑な照会で、REOPT バインド・オプションを入力変数と共に使用する

入力変数は、オンライン・トランザクション処理 (OLTP) 環境において、適切なステートメント準備時間を得る上で不可欠です。こうした環境では、ステートメントは単純になる傾向があり、照会アクセス・プランの選択はより簡単です。

異なる入力変数を使用して、同じ照会を複数実行する場合、動的ステートメント・キャッシュ内のコンパイル済みアクセス・セクションを再使用できます。こうして、入力変数が変わるたびに、コストがかかる SQL ステートメント・コンパイルをする必要がなくなります。

しかし、複雑な照会ワークロードの場合は、入力変数によって問題が引き起こされることがあります。このような場合、照会アクセス・プランの選択はより複雑になり、オプティマイザーは適切な決定をするためにより多くの情報を必要とします。さらに、ステートメント・コンパイル時間は、多くの場合に合計実行時間のわずかな部分です。また、あまり繰り返されることがないビジネス・インテリジェンス (BI) 照会において、動的ステートメント・キャッシュは役立ちません。

複雑な照会ワークロードで入力変数を使用する必要があるなら、REOPT(ALWAYS) バインド・オプションの使用を検討してください。入力変数値が分かっている場合、REOPT バインド・オプションにより、ステートメント・コンパイルが、PREPARE から OPEN または EXECUTE 時間に先送りされます。値は SQL コンパイラーに渡されて、オプティマイザーが選択可能性の見積もりをより正確に計算できるようになります。REOPT(ALWAYS) は、毎回の実行ごとにステートメントを再コンパイルするように指定します。REOPT(ALWAYS) は、特殊レジスターを参照する複雑な照会 (例えば、WHERE TRANS_DATE = CURRENT DATE - 30 DAYS) でも使用できます。入力変数により、OLTP ワークロードにおけるアクセス・プランの選択が良くない結果となり、REOPT(ALWAYS) のゆえに、ステートメント・コンパイルによる過剰なオーバーヘッドが引き起こされているなら、選択された照会のために REOPT(ONCE) の使用を検討してください。REOPT(ONCE) により、最初の入力変数値がバインドされるまで、ステートメント・コンパイルが先送りされます。SQL ステートメントは、この最初の入力変数値を使用して、コンパイルおよび最適化されます。異なる値を使用する後続のステートメント実行では、最初の入力値に基づいてコンパイルされたアクセス・セクションが再使用されます。この方

法は、最初の変数値が後続の値を代表するものであるなら効果的です。また、変数値が不明な場合、デフォルト値に基づく照会アクセス・プランに比べ、より優れたプランが得られます。

以下のとおり、**REOPT** の指定方法は多数あります。

- C/C++ アプリケーションでの組み込み SQL の場合、**REOPT** バインド・オプションを使用します。このバインド・オプションは、静的 SQL と動的 SQL の両方における最適化のやり直し動作に影響を及ぼします。

- CLP パッケージの場合、**REOPT** バインド・オプションを使用して CLP パッケージを再バインドします。例えば、**REOPT ALWAYS** を使用して、分離レベル CS で使用する CLP パッケージを再バインドするには、以下のコマンドを指定します。

```
rebind nullid.SQLC2G13 reopt always
```

- CLI アプリケーションの場合、以下のいずれかの方法で **REOPT** の値を設定します。

- db2cli.ini 構成ファイル内の **REOPT** キーワード設定を使用する。各値および対応するオプションは、以下のとおりです。

- 2 = SQL_REOPT_NONE
- 3 = SQL_REOPT_ONCE
- 4 = SQL_REOPT_ALWAYS

- SQL_ATTR_REOPT 接続またはステートメント属性を使用する。

- SQL_ATTR_CURRENT_PACKAGE_SET 接続またはステートメント属性を使用して、NULLID、NULLIDR1、または NULLIDRA パッケージ・セットのいずれかを指定する。NULLIDR1 および NULLIDRA は、予約済みパッケージ・セット名です。使用時に、**REOPT ONCE** または **REOPT ALWAYS** が、それぞれ暗黙指定されます。これらのパッケージ・セットは、以下のコマンドを使用して明示的に作成される必要があります。

```
db2 bind db2clipk.bnd collection NULLIDR1
db2 bind db2clipk.bnd collection NULLIDRA
```

- IBM Data Server Driver for JDBC and SQLJ を使用する JDBC アプリケーションの場合、**DB2Binder** ユーティリティーの実行時に **-reopt** 値を指定します。

- SQL PL プロシージャの場合、以下の方法の 1 つを使用します。

- **SET_ROUTINE_OPTS** ストアード・プロシージャを使用して、バインド・オプションを設定する。このオプションは、現行セッション内における SQL PL プロシージャの作成に使用されるものです。例えば、以下を呼び出します。

```
sysproc.set_routine_opts('reopt always')
```

- **DB2_SQLROUTINE_PREOPTS** レジストリー変数を使用して、SQL PL プロシージャ・オプションをインスタンスのレベルで設定する。**SET_ROUTINE_OPTS** ストアード・プロシージャを使用して設定された値は、**DB2_SQLROUTINE_PREOPTS** で指定された値をオーバーライドします。

また、最適化プロファイルを使用して、静的および動的ステートメントのために **REOPT** を設定できます。以下はその例です。

```
<STMTPROFILE ID="REOPT example ">
  <STMTKEY>
    <![CDATA[select acct_no from customer where name = ? ]]>
  </STMTKEY>
```

```
<OPTGUIDELINES>
  <REOPT VALUE='ALWAYS' />
</OPTGUIDELINES>
</STMTPROFILE>
```

パラメーター・マーカーを使用した動的照会のコンパイル時間の削減

DB2 データ・サーバーは、動的ステートメント・キャッシュ内にアクセス・セクションおよびステートメント・テキストを格納することにより、前に実行した動的 SQL ステートメントの再コンパイルを回避できます。

このステートメントの後続する準備要求では、動的ステートメント・キャッシュ内にあるアクセス・セクションを検出して、コンパイルを回避しようとします。しかし、述部で使用されるリテラルのみが異なるステートメントは、不一致となります。例えば、以下の 2 つのステートメントは、動的ステートメント・キャッシュ内で異なっているとみなされます。

```
SELECT AGE FROM EMPLOYEE WHERE EMP_ID = 26790
SELECT AGE FROM EMPLOYEE WHERE EMP_ID = 77543
```

比較的単純な SQL ステートメントであっても、非常に頻繁に実行されるなら、ステートメント・コンパイルによってシステム CPU 使用量が過剰になります。ご使用のシステムでこの種のパフォーマンス上の問題が起きているなら、パラメーター・マーカーを使用するようにアプリケーションを変更することを検討してください。こうして、述部の値を明示的に SQL ステートメント内に含める代わりに、DB2 コンパイラーにそれらを渡すようにします。しかし、述部にパラメーター・マーカーを使用する複雑な照会の場合は、アクセス・プランが最適ではない可能性があります。詳しくは、『複雑な照会で、REOPT バインド・オプションを入力変数と共に使用する』を参照してください。

DB2_REDUCED_OPTIMIZATION レジストリー変数の設定

最適化クラスを設定してもアプリケーションのためにコンパイル時間を十分に削減できない場合は、**DB2_REDUCED_OPTIMIZATION** レジストリー変数を設定してみてください。

このレジストリー変数により、最適化クラスの設定を上回る仕方で、オブティマイザーの検索スペースを制御することが可能となります。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クラスに固定して使用するよう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。

最適化に必要な時間とリソースを削減できるかもしれませんが、一方で最適ではない照会アクセス・プランが生成されるリスクは増えます。

最初に、レジストリー変数を YES に設定してみます。最適化クラスが 5 (デフォルト) もしくは 5 未満の場合に、通常はより良い照会アクセス・プランを生成することがなく、相当量の準備時間とリソースを消費する、いくつかの最適化手法をオブティマイザーは使用不可にします。最適化クラスがちょうど 5 の場合、さらにいくつかの手法を削減または使用不可にします。その結果、オブティマイザーによる最適化に必要な時間とリソースをより削減できるかもしれませんが、一方で最適ではない照会アクセス・プランが生成されるリスクは増えます。最適化クラスが 5 未満の場合、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。

YES 設定でコンパイル時間を十分に削減できないなら、レジストリー変数を整数値に設定してみてください。YES と同じ効果があり、さらにクラス 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動作が追加されます。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、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欲張り型結合列挙に切り替えます。その結果、照会は最適化クラス 2 に類似したレベルで最適化されます。

挿入パフォーマンスの向上

データを表に挿入する前に、挿入検索アルゴリズムはフリー・スペース制御レコード (FSCR) を調べて、新しいデータのために十分なスペースがあるページを見つけます。

ただし、FSCR にフリー・スペースを十分に持つページが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削除操作によって予約されていると、このスペースを使用できない可能性もあります。

DB2MAXFSCRSEARCH レジストリー変数は、表へのレコードの追加時に、検索する FSCR の数を指定します。デフォルトでは、5 つの FSCR を検索します。この値の変更によって、スペース再利用で挿入速度の平衡が取れるようになります。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 FSCR を強制的に検索します。FSCR の検索時に十分なスペースが見つからないと、データは表の最後に追加されます。

ALTER TABLE ステートメントの **APPEND ON** オプションは、表データを追加すること、およびページ上のフリー・スペースに関する情報を保持しないことを指定します。そうした表がクラスタリング索引を持つてはなりません。このオプションによって、増加の一途をたどる表のパフォーマンスが向上します。

表においてクラスタリング索引が定義されている場合、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、周辺のページが考慮対象となります。そのようなページが適さないと、前述のように FSCR が検索されます。この場合、『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。最も大きさが異なるスペースを使用する方法では、フリー・スペースがより大きいページが選択される傾向があります。この方式により、同様のキー値を使って行の新しいクラスタ領域が確立されます。

表にクラスタリング索引を定義した場合、表のロードまたは再編成を行う前に、**ALTER TABLE** ステートメントで **PCTFREE** 節を使用します。PCTFREE 節は、ロードや再編成操作の後、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。

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る場合や、最適化のクラスが異なる場合は、パフォーマンスが変化する可能性があります。

効率的な SELECT ステートメントを作成するための、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と、不必要な処理が実行されます。
- 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、返される可能性がある行数の合計よりかなり少ない場合には、OPTIMIZE FOR 節を指定します。この節は、アクセス・プランの選択と通信バッファでブロック化される行数の両方に影響します。
- 行ブロッキングを利用してパフォーマンスを改善するには、FOR READ ONLY または FOR FETCH ONLY 節を指定します。検索された行には排他ロックがかかけられないので、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これらの節を指定すると、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して実行される照会のパフォーマンスが同様に向上する場合があります。
- 位置指定 UPDATE で使用されるカーソルについては、FOR UPDATE OF 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ク・レベルを選択したり、発生する可能性のあるデッドロックを回避したり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。
- 検索条件付き UPDATE で使用されるカーソルについては、FOR READ ONLY および USE AND KEEP UPDATE LOCKS 節を指定し、影響を受ける行を強制的に U ロックすることにより、デッドロックを回避しながらも、行ブロッキングを行えます。
- 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。値を比較するときは、同じデータ・タイプの項目を使うようにします。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、または DECFLOAT ではなく、整数
- 10 進数ではなく、DECFLOAT
- 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 などの節を、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、単一行の SELECT INTO 操作を実行します。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 <table-name>
```

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ます。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。
- 可能なら、結合述部と一緒に OR 節または式を使用しないようにします。
- パーティション・データベース環境では、結合される表が結合列上でパーティション化されるようにすることを推奨します。

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、SELECT ステートメントで指定されたすべての行をアプリケーションが必ず検索すると想定します。オンライン・トランザクション処理 (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、通常は特定の表示フォーマットに必要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 節を使用して、その後の位置指定 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 または FETCH ONLY 節を使用します。
- 全結果セットから最初の n 行の検索を優先させるには、OPTIMIZE FOR n ROWS 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 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。

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 節

FOR UPDATE 節は、その後の位置指定 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 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、FOR UPDATE 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、かつ FOR UPDATE 節で除外されていない場合

CLI アプリケーションの場合、CLI 接続属性 SQL_ATTR_ACCESS_MODE を同じ目的のために使用できます。

FOR READ または FETCH ONLY 節

FOR READ ONLY 節または FOR FETCH ONLY 節は、戻される結果を読み取り専用にします。更新と削除が許可されている結果表では、データベース・マネージャーが排他ロックを使用する代わりにデータのブロックを検索できる場合、FOR READ ONLY 節を指定すると、フェッチ操作のパフォーマンスが向上することがあります。位置指定 UPDATE または DELETE ステートメントで使用される照会では、FOR READ ONLY 節は指定しないでください。

CLI アプリケーションの場合、CLI 接続属性 SQL_ATTR_ACCESS_MODE を同じ目的のために使用できます。

OPTIMIZE FOR *n* ROWS 節

OPTIMIZE FOR 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・プランを選択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、*n* の値によって制限されます。したがって OPTIMIZE FOR 節は、サーバーが適格となる行をデータベースから検索する方法と、それらの行をクライアントに返す方法に影響を与えます。

例えば、最高給を得ている従業員を調べるために、次のようにして EMPLOYEE 表を定期的に照会するとします。

```
select lastname, firstnme, empno, salary
   from employee
  order by salary desc
```

SALARY 列には以前に降順索引を定義しましたが、従業員の順序は従業員番号順になっているため、この索引のクラスター化は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、適格となるすべての行の行 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 節を追加してください。

```
select lastname, firstnme, 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、SALARY 索引を直接使用することを選択すると考えられます。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 節が指定されている場合、オプティマイザーは、大量データ操作やフローの中断 (ソート操作による中断など) が発生しないアクセス・プランを使用しようとしています。OPTIMIZE FOR 1 ROW 節を使用すると、アクセス・パスに最も影響を与えることとなります。この節の使用には、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 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 節に必要な順序を指定している場合は、ORDER BY 節にある列で成る表が外部表として選ばれやすい。

OPTIMIZE FOR 節はすべての最適化レベルに適用されますが、3 より下のクラスは欲張り型結合列挙 検索ストラテジーを使用するので、最適化クラス 3 以上のクラスで最も効率的に処理されます。欲張り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。

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 (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、CLI に OPTIMIZE FOR 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時、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 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、この節はデータ・ソースに送られるリモート SQL 内で生成されます。データ・ソースでこの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 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 節によるパフォーマンスの向上もほとんどありません。

OPTIMIZE FOR 節と FETCH FIRST 節の両方が指定されている場合は、2 つの n 値のうち、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 節

FETCH FIRST *n* ROWS ONLY 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らず、*n* 行だけが検索されます。

FETCH FIRST 節と OPTIMIZE FOR 節の両方が指定されている場合は、2 つの *n* 値のうち、低い方の値が通信バッファ・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。

DECLARE CURSOR WITH HOLD ステートメント

WITH HOLD 節を含む DECLARE CURSOR ステートメントを使用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、現行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。

CLI アプリケーションの場合、CLI 接続属性 SQL_ATTR_CURSOR_HOLD を同じ目的のために使用できます。コール・レベル・インターフェース (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、CLI にすべての宣言されているカーソルについて WITH HOLD 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。

オーバーヘッド削減のための行ブロッキングの指定

行ブロッキングは、すべてのステートメントおよびデータ・タイプ (LOB データ・タイプを含む) においてサポートされます。行ブロッキングにより、単一の操作で複数の行からなるブロックを取り出して、カーソルにおけるデータベース・マネージャのオーバーヘッドを削減します。

このタスクについて

行のブロックとは、メモリー内のページ数を表します。これは、ディスク上のエクステンツに物理的にマップされるマルチディメンション・クラスタリング (MDC) 表ブロックや挿入時クラスタリング (ITC) 表ブロックではありません。

行ブロッキングは、**BIND** コマンドまたは **PREP** コマンドの以下のオプションによって指定されます。

BLOCKING ALL

FOR READ ONLY 節で宣言されているカーソルや、FOR UPDATE として指定されていないカーソルがブロック化されます。

BLOCKING NO

カーソルはブロック化されません。

BLOCKING UNAMBIG

FOR READ ONLY 節で宣言されているカーソルがブロック化されます。

FOR READ ONLY 節または FOR UPDATE 節で宣言されていない、未確定ではなく読み取り専用であるカーソルはブロック化されます。未確定カーソルはブロック化されません。

ブロック・サイズ計算の間、以下のデータベース・マネージャー構成パラメーターが使用されます。

- **aslheapsz** パラメーターにより、ローカル・アプリケーションのためのアプリケーション・サポート層ヒープのサイズを指定します。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。
- **rqrioblk** パラメーターにより、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のサイズを指定します。ブロック・カーソルがオープンされているとき、データ・サーバー・ランタイム・クライアントの入出力ブロック・サイズを決定するのにも使用されます。

LOB データ・タイプに対して行データのブロックングを有効にする前に、システム・リソースに対する影響について理解しておくことは重要です。LOB 列が戻されたときに、サーバーでは LOB 値への参照を各データ・ブロックに保管するために多くの共有メモリーが消費されます。こうした参照の数は、**rqrioblk** 構成パラメーターの値に従って変化します。

ヒープに割り当てられるメモリーの量を増やすには、以下のようにして **database_memory** データベース構成パラメーターを変更します。

- その値を AUTOMATIC に設定する。
- 現在、パラメーターがユーザー定義の数値に設定されている場合は、256 ページずつその値を増やす。

LOB 値を参照する既存の組み込み SQL アプリケーションのパフォーマンスを改善するには、**BIND** コマンドを使用し、**BLOCKING ALL** 節または **BLOCKING UNAMBIG** 節のいずれかを指定してブロックングを要求することにより、アプリケーションを再バインドします。組み込みアプリケーションは、行のブロックがサーバーから取り出されてから、一度に 1 行ずつ LOB 値を取り出します。ユーザー定義関数 (UDF) によって LOB の結果が戻る場合、サーバーで大量のメモリーが消費されているなら、DB2 サーバーは LOB データの単一行の取り出しに戻る可能性があります。

手順

行ブロック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。

$$\text{Rows per block} = \text{aslheapsz} * 4096 / \text{orl}$$

ページごとのバイト数は 4096 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。

$$\text{Rows per block} = \text{rqrioblk} / \text{orl}$$

2. 行ブロッキングを使用可能にするため、**BIND** または **PREP** コマンドの **BLOCKING** オプションに適切な値を指定します。

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 (CLP) およびコール・レベル・インターフェース (CLI) の場合、デフォルトの行ブロッキング・タイプは **ALL** です。

照会でのデータ・サンプリング

1 つの照会に関連したすべてのデータにアクセスすることは実際的ではなく、場合によっては不必要なことがあります。データのサブセットにおける全体の傾向やパターンを見つけるだけ十分な場合もあります。そのための方法の 1 つは、データベースのランダムなサンプルに対して照会を実行することです。

DB2 製品を使用すると、SQL および XQuery 照会のデータのサンプリングを効率的に行えます。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できる可能性があります。

サンプリングは、AVG、COUNT SUM などの集約照会に一般的に使用されます。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速度を上げるためにも、サンプリングを使用できます。

2 つのサンプリング方式を使用できます。行レベルのサンプリングと、ページ・レベルのサンプリングです。

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得します。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・クラスタリングの度合いにかかわらず、常に有効な無作為標本が生成されます。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。索引が使用可能な場合には、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、パフォーマンスは改善されます。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。ページがサンプルに組み込まれる確率は、 $P/100$ です。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桁違いに改善します。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。ページ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、その違いが最も顕著です。

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、TABLESAMPLE 節を使用できます。TABLESAMPLE 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。TABLESAMPLE 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。

アプリケーションの並列処理

DB2 製品は、特に対称型マルチプロセッサ (SMP) マシン上で並列環境をサポートします。

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間で分け合って実行できます。このパーティション内並列処理 では、単一のデータベース操作 (索引の作成など) が複数のパーツに再分割されてから、単一のデータベース・パーティション内で並列して実行されます。

アプリケーションのコンパイル時に並列処理の多重度を指定するには、CURRENT DEGREE 特殊レジスタまたは DEGREE BIND オプションを使用します。多重度とは、並行して実行できる照会のパーツの数を指します。プロセッサ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数よりも多い値または少ない値を指定することもできます。ユニプロセッサ・マシンの場合も、1 より高い多重度を設定してパフォーマンスを改善することができます。しかし、並列処理の多重度が高くなれば、システム・メモリーとプロセッサのオーバーヘッドが増えるという点にご注意ください。

MAXIMUM DEGREE ワークロード属性を使用して、ワークロードの並列処理の度合いを指定することもできます。影響を受けるワークロードでは、MAXIMUM DEGREE を使用して設定された値で、CURRENT DEGREE 特殊レジスタまたは DEGREE BIND オプションにより割り当てられた値がオーバーライドされます。

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。並列処理の多重度の高い環境では、共有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。

以下の構成パラメーターは、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、並列を有効にしたり無効にしたりします。

- **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。

パーティション内並列処理機能をデータベース・アプリケーションの中から有効化または無効化するために、ADMIN_SET_INTRA_PARALLEL プロシージャを呼び出すことができます。ADMIN_SET_INTRA_PARALLEL を設定すると、パーティション内並列処理機能が、自分のアプリケーションだけに適用されます。自分のアプリケーションでは、この値によって **intra_parallel** データベース・マネージャー構成パラメーターがオーバーライドされます。

パーティション内並列処理機能をワークロードの中から有効化または無効化するために、MAXIMUM DEGREE ワークロード属性を設定することができます。この方法を使用すると、パーティション内並列処理機能が、自分のワークロードにだけ適用されます。この値により、**intra_parallel** データベース・マネージャー構成パラメーターと、ADMIN_SET_INTRA_PARALLEL プロシージャにより割り当てられた値があればそれもオーバーライドされます。

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。多重度は、プロセッサの数や照会の特性などのいくつかの要素に基づいて選ばれます。これらの要素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の数よりも少ない場合があります。システムがビジー状態である場合には、並列処理の多重度は照会実行前に減らされる場合があります。

オブティマイザーによって選択された並列処理の多重度に関する情報を表示するには、DB2 Explain 機能を使用します。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。例えば、ユニプロセッサ・マシンで入出力制約の照会を実行する場合でも、並列処理の多重度は 2 以上で宣言しておいた方が有利な可能性があります。この場合、プロセッサは入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。load などのユーティリティーでは、入出力並列処理を独立して制御できます。

ロック管理

ロック管理は、アプリケーション・パフォーマンスに影響を与える要因の 1 つです。データベース・アプリケーションのパフォーマンスを最大化するうえで役立つロック管理上の考慮事項の詳細は、このセクションで調べてください。

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、データ・パーティション、表ブロック、または表の行をロックします。

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。

データベース・マネージャーは、以下に基づいて行レベルのロックングおよび表レベルのロック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。パフォーマンスの影響を最小化するには、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** 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)、TABLE (表ロックの場合)、または **BLOCKINSERT** (マルチディメンション・クラスタリング (MDC) 表のみのブロック・ロックの場合) を選択できます。MDC 表で **BLOCKINSERT** 節が使用される場合、挿入操作中 (ブロック・レベルのロックが行われる) を除いて、行レベルのロックが実行されます。トランザクションが非結合セルに大量の挿入を実行する予定の場合、MDC 表には ALTER TABLE ... **LOCKSIZE BLOCKINSERT** ステートメントを使用してください。読み取り専用の表には、ALTER TABLE ... **LOCKSIZE TABLE** ステートメント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が減ります。パーティション表では、表ロックがまず獲得され、次にアクセスされたデータの命令に従ってデータ・パーティション・ロックが獲得されます。
- ロック専用メモリーの量は、**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・エスカレーションが発生して、データベース内の共有オブジェクトの間の並行性が低くなるために、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**、**maxlocks**、あるいはその両方の値を増やしてください。同時に保持されるロックの数を減らすには、頻繁にトランザクションがコミットされるようにします。

バッファ・プールの作成、変更、またはドロップ時には、バッファ・プールのロック (排他) が設定されます。システムがモニターするデータの収集時にこのタイプのロックを検出することがあります。ロックの名前は、バッファ・プールそのものの識別子 (ID) です。

通常、以下のいずれかに該当する場合以外は、行レベルのロックングが使用されません。

- 分離レベルが非コミット読み取りの場合
- 分離レベルが反復可能読み取りで、アクセス・プランに、索引の範囲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、ロック・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明示的な表ロックが獲得された場合。このステートメントを使用すると、同時アプリケーション・プロセスによる表の変更や使用ができなくなります。

MDC 表の場合には、以下のときは行レベルのロックの代わりにブロック・レベルのロックが使用されます。

- 表の LOCKSIZE 属性が BLOCKINSERT である場合
- 分離レベルが反復可能読み取りで、アクセス・プランに述部が関係する場合
- 検索条件付き UPDATE または DELETE 操作にディメンション列の述部のみが関係する場合

行ロックングの期間は、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。一般的に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。ある場合には、CS スキャン中にロックが全く保持されない可能性があることに注意してください。
- RS スキャン。トランザクション中、適格となる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。

ロックの細分性

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 (ロックされているためにアクセスできないデータの量が最小限に抑えられる) は、ブロック・レベル、データ・パーティション・レベル、または表レベルのロックと比較して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 節は、行、データ・パーティション、ブロック、または表レベルでロックの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。表定義の中でこのオプションを使用しても、通常のロック・エスカレーションの発生に支障はありません。

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。

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 ステートメントで永続的な表ロックを定義する方が良いかもしれません。

- 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。

マルチディメンション・クラスタリング (MDC) 表の場合、挿入操作の間のみブロック・レベルのロックを使用するために、LOCKSIZE 節で BLOCKINSERT を指定できます。BLOCKINSERT が指定されると、他のすべての操作には行レベルのロックが実行されますが、挿入操作には最低限のロックしか実行されません。つまり、行の挿入時にはブロック・レベルのロックが使用されますが、レコード ID (RID) 索引の更新中にその中で反復可能読み取り (RR) スキャンが検出される場合には次のキーのロックに行レベルのロックが使用されます。BLOCKINSERT ロックは次のような場合に役立ちます。

- 異なるセルに大量の挿入を実行する複数のトランザクションがある
- 複数のトランザクションによって同じセルに対して並行挿入が発生しない場合、または個々のブロックへ挿入することを想定していない各トランザクションによってセルごとに十分なデータが挿入されて並行挿入が発生する場合

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、いくつかの基本属性があります。

属性には以下のものが含まれます。

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。データベース・マネージャーは、行、表スペースなど、他のタイプのリソースにもロックを設定します。マルチディメンション・クラスタリング (MDC) 表や挿入時クラスタリング (ITC) 表にはブロック・ロックも設定でき、パーティション表にはデータ・パーティション・ロックを設定できます。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。

ロック・カウント

ロックが保持される時間の長さ。照会が実行される分離レベルは、ロック・カウントに影響を与えます。

表 16 のリストは、ロック・モードとその効果を示しています。ここに示す順に、リソースへの制御が大きくなります。

表 16. 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、データ・パーティション	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 (意図的共有)	表スペース、ブロック、表、データ・パーティション	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ができませんが、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。
IX (意図的排他)	表スペース、ブロック、表、データ・パーティション	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。
NS (スキャン共有)	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ができますが、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。
NW (次キーの弱い排他)	行	行が索引に挿入される時、NW ロックが次の行に獲得されます。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、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。
S (共有)	行、ブロック、表、データ・パーティション	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ができますが、更新はできません。
SIX (意図的排他共有)	表、ブロック、データ・パーティション	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、データ・パーティション	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ができますが、更新はできません。
X (排他)	行、ブロック、表、バフパー・プール、データ・パーティション	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 (UR) アプリケーションだけです。
Z (超排他)	表スペース、表、データ・パーティション	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時、表の索引が作成またはドロップされる時、あるいは表の一部のタイプが再編成される時など、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。

ロッキングに影響を与える要因

いくつかの要素がデータベース・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す。

それらの要素には、次のものが含まれます。

- 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
- さまざまな構成パラメーターの値

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、変更を意図、変更、およびカーソル制御。

- 読み取り専用

この処理タイプには、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` 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` または `BIND` コマンドで指定された `BLOCKING` オプションの値が根拠となり照会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。このタイプは、共有ロック (`IS`、`NS`、または `S`) のみを必要とします。

- 変更を意図

この処理タイプには、`FOR UPDATE` 節、`USE AND KEEP UPDATE LOCKS` 節、`USE AND KEEP EXCLUSIVE LOCKS` 節を持つすべての `SELECT` ステートメント、または明示的ではないものの、変更が意図されていると照会コンパイラーが見なす `SELECT` ステートメントが含まれます。このタイプは、共有および更新ロック (行では `S`、`U`、または `X`。ブロックでは `IX`、`S`、`U`、または `X`。表では `IX`、`U`、または `X`) を使用します。

- 変更

この処理タイプには `UPDATE`、`INSERT`、および `DELETE` ステートメントが含まれますが、`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 (`IX` または `X`) が必要です。

- カーソル制御

この処理タイプには `UPDATE WHERE CURRENT OF` および `DELETE WHERE CURRENT OF` が含まれます。このタイプには排他ロック (`IX` または `X`) 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データを戻す表のロックは、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。

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。

索引スキャンを使ってある特定の行を見つける場合、通常オプティマイザーは表に関する行レベル・ロッキング (`IS`) を選択します。例えば、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、以下のステートメントを使用して、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、必要な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、オプティマイザーはおそらく、単一表レベル・ロック (S) を獲得することになります。例えば、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、更新や削除操作を行うときは必ず排他ロックが獲得されます。

範囲がクラスター化された表のロックインの作業は、標準キー・ロックの作業とは若干異なります。範囲がクラスター化された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・ロックの場合、既存のデータがある行だけがロックされます。

データ・ページへ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、それによりロック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 (RR) 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 (RS) 分離レベルおよびカーソル固定 (CS) 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。

ロック・タイプの互換性

ロックの互換性は、あるアプリケーションがあるオブジェクトのロックを保持しているときに、別のアプリケーションが同じオブジェクトのロックを要求する場合に問題になります。2 つのロック・モードに互換性があれば、オブジェクトに対する 2 番目のロックの要求は認可されます。

要求されたロックのロック・モードがすでに保持されているロックと互換性がないなら、ロック要求は認可されません。その場合、要求は最初のアプリケーションがロックを解放し、さらに他の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。

表 17 では、互換性のあるロック・タイプ (はい で表示) および互換性のないロック・タイプ (いいえ で表示) 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。

表 17. 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										
	None	IN	IS	NS	S	IX	SIX	U	X	Z	NW
なし	はい	はい	はい	はい	はい	はい	はい	はい	はい	はい	はい

表 17. ロック・タイプの互換性 (続き)

要求されている状態	保持されているリソースの状態										
	None	IN	IS	NS	S	IX	SIX	U	X	Z	NW
IN (意図なし)	はい	はい	はい	はい	はい	はい	はい	はい	はい	いいえ	はい
IS (意図的共有)	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
NS (スキャン共有)	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
S (共有)	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
IX (意図的排他)	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
SIX (意図的排他共有)	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
U (更新)	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
X (排他)	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
Z (超排他)	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
NW (次キーの弱い排他)	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

次キー・ロッキング

キーが索引に挿入される時、索引内で新しいキーの次のキーに対応する行が反復可能読み取り (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。このようになった場合、新しい索引キーの挿入は、RR スキャンを実行したトランザクションが完了するまで据え置かれます。

次キーロックで使用されるロック・モードは NW (次キーの弱い排他) です。この次キーロックは、キー挿入が実行される前、つまり行が表に挿入される前に解放されます。

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、キーをその場で更新することができ、次キー・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。パーティション表のパーティション索引の場合、索引の末尾のためのロックが 1 つだけ獲得されるのではなく、各索引パーティションの末尾をロックするために複数のロックが獲得されます。スキャン範囲の終わりの次に来るキーが削除済みとマークされている場合、以下のいずれかのアクションが起こります。

- スキャンが、削除済みとマークされていないキーを検出するまで、対応する行のロックを継続する
- スキャンが、そのキーの対応する行をロックする
- スキャンが、索引の末尾をロックする

標準の表のロック・モードおよびアクセス・プラン

標準の表で取得されるロックのタイプは、有効である分離レベル、および使用中のデータ・アクセス・プランによって決まります。

以下の表では、種々のアクセス・プランごとに、各分離レベルにおいて標準の表で取得されるロックのタイプをリストします。各項目には、表ロックおよび行ロックの 2 つの部分があります。ハイフンは、特定のロック細分性が使用できないことを示します。

表 7 から 12 で示すロックのタイプは、以下の場合に取得されるものです。すなわち、データ・ページの読み取りが据え置かれて、行のリストを複数の索引の使用によってさらに修飾すること、または効率的なプリフェッチのためにソートすることが可能になる場合です。

- 表 1. 述部なしの表スキャンのロック・モード
- 表 2. 述部での表スキャンのロック・モード
- 表 3. 述部なしの RID 索引スキャンのロック・モード
- 表 4. 単一修飾行での RID 索引スキャンのロック・モード
- 表 5. 開始述部と停止述部のみでの RID 索引スキャンのロック・モード
- 表 6. 索引と他の述部 (sargs、resids) のみでの RID 索引スキャンのロック・モード
- 表 7. 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: 述部なしでの RID 索引スキャン
- 表 8. 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: 述部なしでの RID 索引スキャン後
- 表 9. 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: 述部 (sargs、resids) での RID 索引スキャン
- 表 10. 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: 述部 (sargs、resids) での RID 索引スキャン後
- 表 11. 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: 開始述部と停止述部のみの RID 索引スキャン
- 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: 開始述部と停止述部のみの RID 索引スキャン後

注:

1. マルチディメンション・クラスタリング (MDC) 表および挿入時クラスタリング (ITC) 表では、ブロック・レベルのロックも使用可能です。
2. ロック・モードは、SELECT ステートメントの *lock-request-clause* を使用して明示的に変更できます。

表 18. 述部なしの表スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 19. 述部ありの表スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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: UR 分離レベルにおいて、索引の組み込み列に述部がある場合、分離レベルは CS にアップグレードされ、ロックは IS 表ロックまたは NS 行ロックにアップグレードされます。

表 20. 述部なしの RID 索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	S/-	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 21. 単一修飾行での RID 索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 22. 開始述部と停止述部のみでの RID 索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 23. 索引と他の述部 (sargs、resids) のみでの RID 索引スキンのロック・モード

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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

表 24. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部なしでの RID 索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
RR	IS/S	IX/S		X/-	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

表 25. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部なしでの RID 索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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 26. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部 (sargs、resids) での RID 索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
RR	IS/S	IX/S		IX/S	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

表 27. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部 (sargs, resids) での RID 索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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

表 28. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 開始述部と停止述部のみの RID 索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
RR	IS/S	IX/S		IX/X	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

表 29. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 開始述部と停止述部のみの RID 索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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

MDC 表および ITC 表と RID 索引スキンのロック・モード

表または RID 索引のスキン中に、マルチディメンション・クラスタリング (MDC) 表または挿入時クラスタリング (ITC) 表で取得されるロックのタイプは、有効になっている分離レベル、および使用中のデータ・アクセス・プランによって決まります。

以下の表では、種々のアクセス・プランごとに、各分離レベルにおいて MDC 表および ITC 表で取得されるロックのタイプをリストします。各項目には、表ロック、ブロック・ロック、および行ロックの 3 つの部分があります。ハイフンは、特定のロック細分性が使用できないことを示します。

表 9 から 14 で示すロックのタイプは、データ・ページの読み取りの据え置き時に RID 索引スキャンにおいて取得されるものです。UR 分離レベルにおいて、索引の組み込み列に述部がある場合、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、または NS 行ロックにアップグレードされます。

- 表 1. 述部なしの表スキャンのロック・モード
- 表 2. ディメンション列上の述部のみでの表スキャンのロック・モード
- 表 3. 索引と他の述部 (sargs、resids) での表索引スキャンのロック・モード
- 表 4. 述部なしの RID 索引スキャンのロック・モード
- 表 5. 単一修飾行での RID 索引スキャンのロック・モード
- 表 6. 開始述部と停止述部のみでの RID 索引スキャンのロック・モード
- 表 7. 索引述部のみでの RID 索引スキャンのロック・モード
- 表 8. 他の述部 (sargs、resids) での RID 索引スキャンのロック・モード
- 表 9. 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: 述部なしでの RID 索引スキャン
- 表 10. 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: 述部なしでの RID 索引スキャン後
- 表 11. 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: 述部 (sargs、resids) での RID 索引スキャン
- 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: 述部 (sargs、resids) での RID 索引スキャン後
- 表 13. 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: 開始述部と停止述部のみの RID 索引スキャン
- 表 14. 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: 開始述部と停止述部のみの RID 索引スキャン後

注: ロック・モードは、SELECT ステートメントの *lock-request-clause* を使用して明示的に変更できます。

表 30. 述部なしの表スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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/-

表 31. ディメンション列上の述部のみでの表スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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/-

表 32. 索引と他の述部 (sargs、resids) での表索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 33. 述部なしの RID 索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 34. 単一修飾行での RID 索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

表 35. 開始述部と停止述部のみでの RID 索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 36. 索引述部のみでの RID 索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 37. 他の述部 (sargs、resids) での RID 索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 38. 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: 述部なしでの RID 索引スキャン

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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/-	

表 39. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部なしでの RID 索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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 40. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部 (sargs、resids) での RID 索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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/-	

表 41. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部 (sargs、resids) での RID 索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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

表 42. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 開始述部と停止述部のみの RID 索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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/-	

表 43. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 開始述部と停止述部のみの RID 索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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

MDC ブロック索引スキンのロック・モード

ブロック索引スキン中に、マルチディメンション・クラスタリング (MDC) 表で取得されるロックのタイプは、有効である分離レベル、および使用中のデータ・アクセス・プランによって決まります。

以下の表では、種々のアクセス・プランごとに、各分離レベルにおいて MDC 表で取得されるロックのタイプをリストします。各項目には、表ロック、ブロック・ロック、および行ロックの 3 つの部分があります。ハイフンは、特定のロック細分性が使用できないことを示します。

表 5 から 12 で示すロックのタイプは、データ・ページの読み取りの据え置き時にブロック索引スキンにおいて取得されるものです。

- 表 1. 述部なしでの索引スキンのロック・モード
- 表 2. ディメンション列上の述部のみでの索引スキンのロック・モード
- 表 3. 開始述部と停止述部のみでの索引スキンのロック・モード
- 表 4. 述部での索引スキンのロック・モード
- 表 5. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部なしでのブロック索引スキン
- 表 6. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部なしでのブロック索引スキン後
- 表 7. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: ディメンション列上の述部のみでのブロック索引スキン
- 表 8. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: ディメンション列上の述部のみでのブロック索引スキン後
- 表 9. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 開始述部と停止述部のみでのブロック索引スキン
- 表 10. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 開始述部と停止述部のみでのブロック索引スキン後
- 表 11. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキン
- 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ンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキン後

注: ロック・モードは、SELECT ステートメントの *lock-request-clause* を使用して明示的に変更できます。

表 44. 述部なしでの索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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/--

表 45. ディメンション列上の述部のみでの索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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/-

表 46. 開始述部と停止述部のみでの索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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/-

表 47. 述部での索引スキャンのロック・モード

分離レベル	読み取り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 UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
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

表 48. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部なしでのブロック索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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/--	

表 49. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 述部なしでのブロック索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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/--

表 50. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: ディメンション列上の述部のみでのブロック索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

表 51. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: ディメンション列上の述部のみでのブロック索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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/--

表 52. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 開始述部と停止述部のみのブロック索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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/--	

表 53. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 開始述部と停止述部のみのブロック索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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/--	

表 54. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキン

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 55. 据え置きデータ・ページ・アクセスに使用される索引スキンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキン後

分離レベル	読み取り専用および未確定のスキン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキン	現在の場所	スキン	更新または削除
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

パーティション表でのロック動作

表全体のロックに加えて、パーティション表の各データ・パーティションごとのロックがあります。

これにより、非パーティション表と比べて、より良い細分性および並行性の増大が可能になります。データ・パーティション・ロックは、**db2pd** コマンド、イベント・モニター、管理ビュー、および表関数からの出力によって識別されます。

表がアクセスされると、まず表ロックが取得され、その後データ・パーティション・ロックが必要に応じて取得されます。アクセス方式および分離レベルは、結果セットに示されていないデータ・パーティションのロックを必要とする場合があります。これらのデータ・パーティション・ロックが取得されると、表ロックと同じだけ長く保持される可能性があります。例えば、カーソル固定 (CS) の索引のスキューンは、以前にアクセスされたデータ・パーティションでロックを維持し、後にデータ・パーティション・ロックを再取得するコストを削減する可能性があります。

さらにデータ・パーティション・ロックは、表スペースへのアクセスを確保するコストを担います。非パーティション表の場合、表スペースのアクセスは表ロックによって処理されます。表レベルに排他ロックまたは共有ロックがある場合でも、データ・パーティション・ロックは発生します。

より良い細分性によって、1 つのトランザクションは、他のトランザクションが他のデータ・パーティションにアクセスしている間に、特定のデータ・パーティションへ排他的にアクセスでき、行ロックを避けることができます。これは、大量更新用に選択されるプランの結果として、またはデータ・パーティション・レベルへのロックのエスカレーションによって生じることがあります。データ・パーティションが共有または排他モードでロックされている場合であっても、多数のアクセス方式の表ロックは、通常意図的ロックです。これにより、並行性が増大します。しかし、非意図的ロックがデータ・パーティション・レベルで必要とされ、すべてのデータ・パーティションがアクセスされる可能性があることをプランが示す場合には、並行トランザクション間のデータ・パーティション・デッドロックが発生しないようにするために、表レベルで非意図的ロックが選択されることがあります。

LOCK TABLE ステートメント

パーティション表の場合、LOCK TABLE ステートメントによって取得される唯一のロックは表レベル・ロックです。これにより、後続のデータ操作言語 (DML) ステートメントによって行がロックされることが回避され、行、ブロック、またはデータ・パーティションの各レベルでのデッドロックを避けられます。IN EXCLUSIVE MODE オプションを使用すると、索引を更新するときに排他的アクセスを保証するのに使用でき、大きな更新の間に索引が増大するのを制限するのに役立ちます。

ALTER TABLE ステートメントの LOCKSIZE TABLE オプションの影響

LOCKSIZE TABLE オプションを使用すると、意図的ロックを用いないで共有モードまたは排他モードで表をロックします。パーティション表の場合、このロック計画は、表ロックとデータ・パーティション・ロックの両方に適用されます。

行レベルおよびブロック・レベルのロックのエスカレーション

パーティション表の行レベルおよびブロック・レベルのロックは、データ・パーティション・レベルにエスカレートできます。これが生じる場合、データ・パーティションが共有、排他、または超排他モードにエスカレートされる場合にも、他のデータ・パーティションは影響を受けないので、表が他のトランザクションによりさらにアクセスできます。エスカレーションの通知ログ項目には、影響を受けるデータ・パーティションとその表の名前が含まれます。

ロック・エスカレーションによって非パーティション索引への排他的アクセスを確保することはできません。排他的アクセスの場合、以下のいずれかの条件が TRUE でなければなりません。

- ステートメントは排他表レベルのロックを使用する必要があります
- 明示的 LOCK TABLE IN EXCLUSIVE MODE ステートメントが発行される必要があります
- 表には LOCKSIZE TABLE 属性がなければなりません

パーティション索引の場合、データ・パーティションの排他的アクセス・モードまたは超排他アクセス・モードへのロック・エスカレーションにより、索引パーティションへの排他的アクセスを確保します。

ロック情報の解釈

SNAPLOCK 管理ビューは、パーティション表から戻されるロック情報を解釈する際に役立ちます。以下の SNAPLOCK 管理ビューは、オフラインでの索引再編成中にキャプチャーされたものです。

```
SELECT SUBSTR(TABNAME, 1, 15) TABNAME, TAB_FILE_ID, SUBSTR(TBSP_NAME, 1, 15) TBSP_NAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_ESCALATION FROM SYSIBMADM.SNAPLOCK where TABNAME like 'TP1' and LOCK_OBJECT_TYPE like 'TABLE_%' ORDER BY TABNAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE, TAB_FILE_ID, LOCK_MODE
```

TABNAME	TAB_FILE_ID	TBSP_NAME	DATA_PARTITION_ID	LOCK_OBJECT_TYPE	LOCK_MODE	LOCK_ESCALATION
TP1		32768 -	-1	TABLE_LOCK	Z	0
TP1	4	USERSPACE1	0	TABLE_PART_LOCK	Z	0
TP1	5	USERSPACE1	1	TABLE_PART_LOCK	Z	0
TP1	6	USERSPACE1	2	TABLE_PART_LOCK	Z	0
TP1	7	USERSPACE1	3	TABLE_PART_LOCK	Z	0
TP1	8	USERSPACE1	4	TABLE_PART_LOCK	Z	0
TP1	9	USERSPACE1	5	TABLE_PART_LOCK	Z	0
TP1	10	USERSPACE1	6	TABLE_PART_LOCK	Z	0
TP1	11	USERSPACE1	7	TABLE_PART_LOCK	Z	0
TP1	12	USERSPACE1	8	TABLE_PART_LOCK	Z	0
TP1	13	USERSPACE1	9	TABLE_PART_LOCK	Z	0
TP1	14	USERSPACE1	10	TABLE_PART_LOCK	Z	0
TP1	15	USERSPACE1	11	TABLE_PART_LOCK	Z	0
TP1	4	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	5	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	6	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	7	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	8	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	9	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	10	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	11	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	12	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	13	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	14	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	15	USERSPACE1	-	TABLE_LOCK	Z	0
TP1	16	USERSPACE1	-	TABLE_LOCK	Z	0

26 record(s) selected.

この例では、パーティション表 TP1 に対するアクセスと並行性の制御に、タイプ TABLE_LOCK のロック・オブジェクトと -1 の DATA_PARTITION_ID を使用しています。タイプ TABLE_PART_LOCK のロック・オブジェクトは、各データ・パーティションのほとんどのアクセスおよび並行性の制御に使用されます。

この出力では、他にもタイプ TABLE_LOCK のロック・オブジェクトがありますが (TAB_FILE_ID 4 から 16)、これらのロック・オブジェクトには DATA_PARTITION_ID の値がありません。この種のロック (オブジェクトがデータ・パーティションまたはパーティション表の索引に対応する TAB_FILE_ID と TBSP_NAME を持つ) は、オンライン・バックアップ・ユーティリティとの並行性を制御するのに使用される場合があります。

ロックの変換

既に保持しているロック・モードの変更は、**ロック変換** 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任意の一時点で 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、ブロック、またはデータ・パーティションにのみ適用されます。行またはブロックの場合、通常、X ロックが必要なときに S または U ロックを保持している場合に、変換が行われます。

IX および S ロックは、ロック変換での特殊ケースです。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、変換は結果として SIX (意図的排他共有) ロックになります。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があります。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。システム・モニター・エレメントの `lock_current_mode` および `lock_mode` は、データベースで発生しているロック変換に関する情報を提供することができます。

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・マネージャーのフィーチャーです。

例えば、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

セッ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ワークステーション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** データベース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。ロック要求がペー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にエラーが戻り、トランザクションがロールバックされます。例えば、APPL1 が、既に APPL2 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、APPL1 は SQLCODE -911 (SQLSTATE 40001) と理由コード 68 を受け取ります。**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無効になります。

アプリケーションは、CURRENT LOCK TIMEOUT 特殊レジスターの値を変更して、表、行、データ・パーティション、およびマルチディメンション・クラスタリング (MDC) ブロック・ロックに対する **locktimeout** 値をオーバーライドできます。

ロック・タイムアウトに関するレポート・ファイルを生成するには、**DB2_CAPTURE_LOCKTIMEOUT** レジストリー変数を ON に設定します。このロック・タイムアウト・レポートに含まれる情報には、ロックのタイムアウトの原因となったロック競合に関係した主なアプリケーション、およびロックに関する詳細 (ロック名、ロック・タイプ、行 ID、表スペース ID、および表 ID など) に関する情報が含まれます。この変数は推奨されておらず、将来のリリースで除去される可能性があることに注意してください。それは、ロック・タイムアウト・イベントを収集するための、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用する新しい方法があるためです。

db2diag ログ・ファイル内にロック要求タイムアウトに関するより多くの情報を記録するには、**diaglevel** データベース・マネージャー構成パラメーターの値を 4 に設定します。ログに記録される情報には、ロックされたオブジェクト名、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 または XQuery 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 または XQuery ステートメントは、**diaglevel** が 4 である場合にのみログに記録されます。

ロック待機およびロック・タイムアウトに関する追加情報を、ロック待機情報システム・モニター・エレメント、または **db.apps_waiting_locks** ヘルス・インディケーターから取得することができます。

ロック待機モードの方針を指定する

セッションで、ロック待機モードの方針を指定できます。この方針は、セッションが即時に取得できないロックを必要とするときに使用します。

方針は、セッションが次の処理をするかどうか示します。

- ロックを取得できないときに SQLCODE と SQLSTATE を戻す
- ロックを無限に待機する

- ロックを指定時間、待機する
- ロックを待機するときデータベース構成パラメーター **locktimeout** の値を使用する

ロック待機モードの方針は、`SET CURRENT LOCK TIMEOUT` ステートメントで指定されます。これは特殊レジスター `CURRENT LOCK TIMEOUT` の値を変更します。この特殊レジスターでは、ロックを取得できないことを示すエラーを戻す前にロックを待機する秒数を指定します。

伝統的なロッキング方法では、アプリケーションが互いにブロックする可能性があります。これは、あるアプリケーションが別のアプリケーションによるロックの解放を待機しているときに発生します。このようなブロッキングの影響を処理する方針では、通常ブロックの最大許容期間を指定するためのメカニズムを提供します。これは、ロックを取得しない場合でもアプリケーションが待機する時間です。以前は、データベース構成パラメーター **locktimeout** の値を変更することによってデータベース・レベルでのみこの値を指定できました。

locktimeout の値をすべてのロックに適用しますが、ロック待機モードの方針によって影響を受けるロック・タイプには、行、表、索引キー、およびマルチディメンション・クラスタリング (MDC) ブロック・ロックがあります。

デッドロック

デッドロックは、2 つのアプリケーションが他方が必要とするデータをロックし、その結果、いずれのアプリケーションも実行を継続できないときに作成されます。

例えば、245 ページの図 23 ではアプリケーション A とアプリケーション B の 2 つのアプリケーションが同時に実行されています。アプリケーション A の最初のトランザクションが表 1 の最初の行を更新しようとし、2 番目のトランザクションは表 2 の 2 行目を更新しようとしています。アプリケーション B は表 2 の 2 行目を最初に更新し、その後に表 1 の 1 行目を更新します。T1 で、アプリケーション A は表 1 の 1 行目をロックします。同時にアプリケーション B が表 2 の 2 行目をロックします。T2 で、アプリケーション A が表 2 の 2 行目のロックを要求します。しかし、アプリケーション B は同時に表 1 の 1 行目をロックしようとしています。アプリケーション A は表 2 の 2 行目の更新を完了できるまでは表 1 の 1 行目のロックを解放しませんし、アプリケーション B は表 1 の 1 行目の更新が完了できるまでは表 2 の 2 行目のロックを解放しないので、デッドロックが生じます。これらのアプリケーションは、他方がデータのロックを解放するまで待機します。

デッドロックの概念

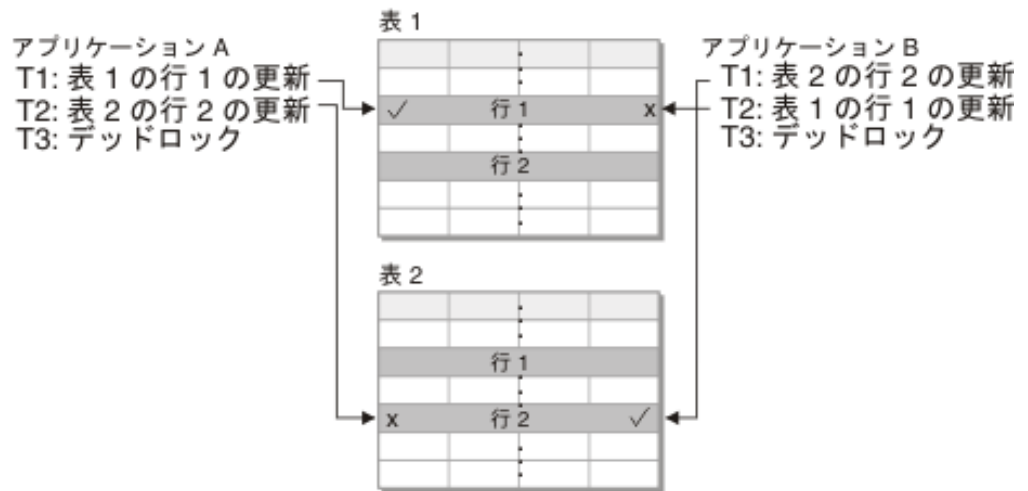


図 23. アプリケーション間のデッドロック

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するためのデッドロック検出プロセスが必要です。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、**dlchktime** データベース構成パラメーターで指定された間隔で実行されます。

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任意に選択します。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から非コミット・トランザクションを自動的にロールバックします。ロールバック操作が完了すると、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。

適切なパフォーマンスを確保するには、**dlchktime** に適切な値を選択してください。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がいつまでも続いてしまいます。

パーティション・データベース環境では、**dlchktime** の値はカタログ・データベース・パーティションにのみ適用されます。大量のデッドロックが生じる場合には、**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。

アプリケーションがその後に更新する予定のデータを読み取っている際にデッドロックが生じないようにするには、以下のようにします。

- 選択操作を実行するときには **FOR UPDATE** 文節を使用します。この文節によって、プロセスがデータを読み取ろうとするときに **U** ロックが設定され、行ブロッキングが不可能になります。
- 照会で **WITH RR** または **WITH RS**、および **USE AND KEEP UPDATE LOCKS** 文節を使用します。これらの文節によって、プロセスがデータを読み取ろうとするときに **U** ロックが設定され、行ブロッキングが可能になります。

フェデレーテッド・システムでは、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生じ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 サーバーはデータ・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 サーバーはデータ・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より多くの情報をログに記録するには、**diaglevel** データベース・マネージャー構成パラメーターの値を 4 に設定します。ログに記録される情報には、ロックされたオブジェクト名、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 および XQuery 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。

照会の最適化

照会の最適化は、アプリケーション・パフォーマンスに影響を与える要因の 1 つです。データベース・アプリケーションのパフォーマンスを最大化するうえで役立つ照会の最適化に関する考慮事項の詳細は、このセクションで調べてください。

SQL および XQuery コンパイラーの処理

SQL および XQuery 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。

照会グラフ・モデル とは、247 ページの図 24 で示されているステップで処理される照会を表す、メモリー内の内部データベースです。一部のステップは、フェデレーテッド・データベースに対して実行される照会のみ該当することに注意してください。

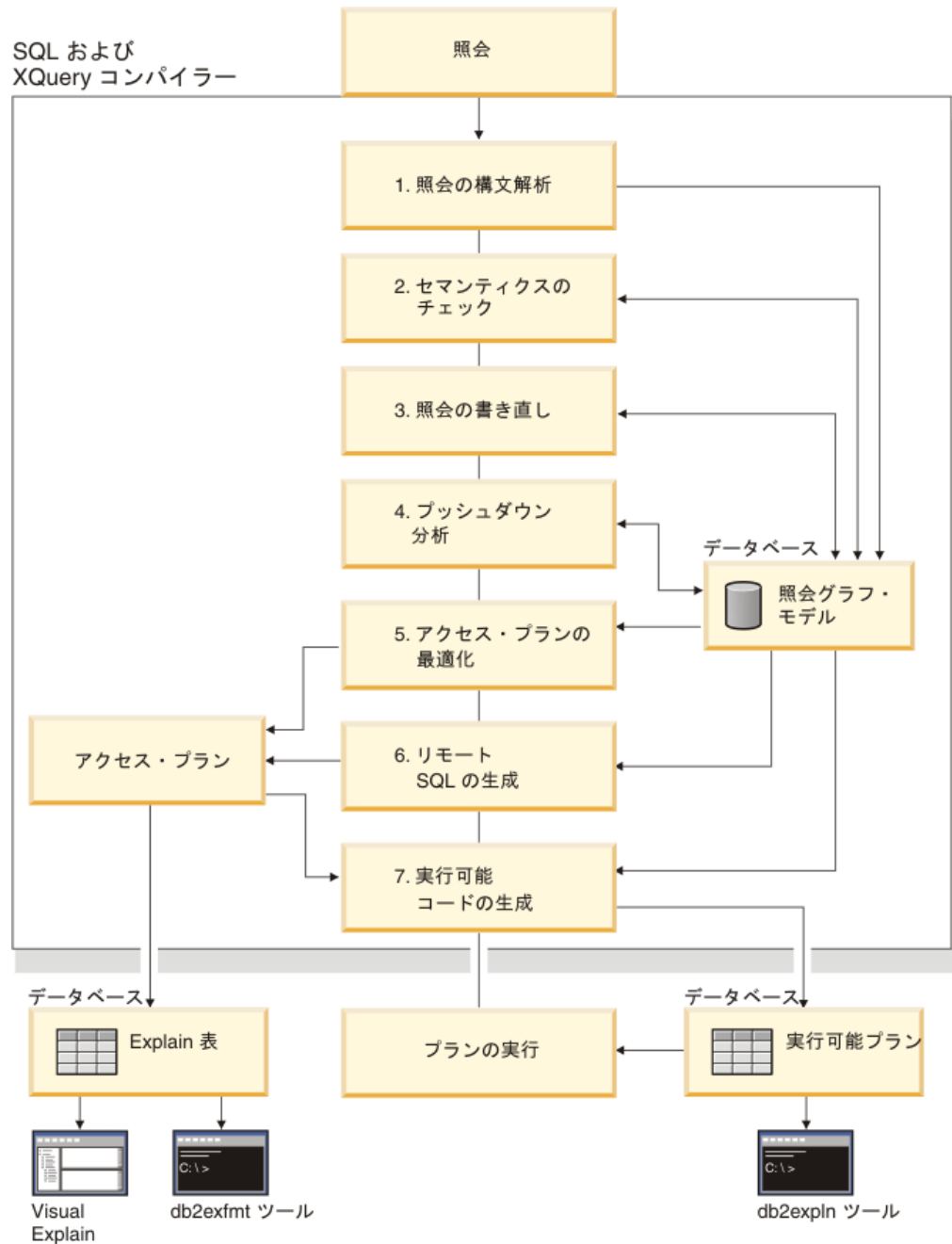


図 24. SQL および XQuery コンパイラーによって実行されるステップ

1. 照会の構文解析

SQL および XQuery コンパイラーは、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、照会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. セマンティクスのチェック

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。例えば、コンパイラーは、YEAR スカラー関数用に指定した列が日時データ・タイプで定義されていることを検査します。

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、および分散キーを含む、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トランスフォームします。その後、その結果を照会グラフ・モデルに保管します。

例えば、コンパイラーは述部を移動して、適用されるレベルを変更することにより、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・データベース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・データベースのみ)

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。

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。これらの各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。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 (DMS) 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・サイズを選択時に、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。データベースがパーティショ

ン化されているかどうか、または対称型マルチプロセッサ (SMP) 環境で照会内並列処理の可能性があるかどうかを考慮されます。この情報は、オプティマイザ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。

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・プランに関する詳細は、`Explain` 表にキャプチャーされます。アクセス・プランを生成するために使用される情報は、`Explain` スナップショットで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・データベースのみ)

オプティマイザ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、そのデータ・ソースの SQL ダイアレクトに基づく有効な SQL ステートメントが作成されます。

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 回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。この種の最適化は、コード・ページ変換の場合やホスト変数が使用される場合に可能です。

ホスト変数、特殊レジスタ、またはパラメータ・マーカーを持つ静的または動的 SQL あるいは XQuery ステートメントの照会最適化または再最適化を有効にするには、`REOPT BIND` オプションを使用してパッケージをバインドします。そのようなパッケージに属し、かつホスト変数、特殊レジスタ、またはパラメータ・マーカーを含んだステートメントのアクセス・パスは、コンパイラが選択するデフォルトの見積もりではなく、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 および XQuery ステートメントのアクセス・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・マネージャはシステム・カタログ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、`db2exp1n` ツールによって使用されます。

注: たびたび変更される表に対しては、`RUNSTATS` コマンドを適切な間隔で実行してください。オプティマイザが最も効率的なアクセス・プランを作成するには、表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、空かほぼ空の表に対してこのコマンドが実行されたとおプティマイザが想定している) 場合には、オプティマイザは、デフォルト値を使用するか、あるいはディスク上に表を保管するために使用されるファイル・ページ数に基づいて特定の統計を導き出そうとします。『自動統計収集』も参照してください。

照会書き直しの方法とその例

照会書き直し段階では、照会コンパイラは SQL および XQuery ステートメントをより最適化しやすい形式に変換します。これによりアクセス・プランの候補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 または XQuery 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。照会書き直しプロセスの結果をいくつか表示させるには、`EXPLAIN` 機能を使用します。

照会の書き直しは、以下のいずれの方法でも行えます。

- 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 および XQuery 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 - 共有集約

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、最適化で `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。場合によっては、`DISTINCT` 操作は完全に除去できます。

- 例 - 一般的な述部プッシュダウン

照会書き直し時に、最適化は述部を適用する順序を変更して、選択範囲のより狭い述部ができるだけ早い機会に照会に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・データベース環境では、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) を減らすことが、照会書き直しプロセスの目標です。

- 述部の変換

SQL および XQuery コンパイラーは照会を書き直して、既存の述部をより最適化された形式に変換します。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

照会を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、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 および XQuery コンパイラーでそのような変換をすることもできます。

コンパイラー書き直しの例: 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。

EMPLOYEE 表に基づく次の 2 つのビューへのアクセス権限があるとします。一方は高学歴の従業員を表示するビューで、他方は \$35,000 を超える年収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, firstnme, lastname, salary) as
select empno, firstnme, lastname, salary
from employee
where salary > 35000
```

次のユーザー作成照会を使用すると、高学歴でかつ \$35,000 を超える年収がある従業員がリストされます。

```
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 および XQuery 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

照会には不要な結合が含まれる場合があります。

```
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
```

SQL および XQuery 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 表と DEPARTMENT 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るにもかかわらず、基礎表へのアクセス権限がないため書き直すことができない場合がある点に注意してください。アクセス権限は、上記のビューに対するものしかない可能性があります。したがって、このタイプの最適化は、データベース・マネージャーで実行する必要があります。

次のような場合、参照整合性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

例 - 共有集約

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。必要な計算の数を減らすことによって、プランが改善されます。コンパイラーが、以下のような複数の関数を使用する照会を処理するとします。

```
select sum(salary+bonus+comm) as osum,
       avg(salary+bonus+comm) as oavg,
       count(*) as ocount
from employee
```

これが次のようにトランスフォームされます。

```
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、sum と count の必要数が半分になります。

コンパイラー書き直しの例: 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。場合によっては、DISTINCT 操作は完全に除去できます。

例えば、EMPLOYEE 表の EMPNO 列を主キーとして定義した場合に、次の照会について考えます。

```
select distinct empno, firstnme, lastname
from employee
```

これは、DISTINCT 節を除去することで書き直せます。

```
select empno, firstnme, lastname
from employee
```

この例では、主キーが選択されているため、コンパイラーは、返される各行がユニークであると認識しています。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必要な処理 (ソートなど) を含むプランを作成して、列値が必ずユニークになるようにしなければなりません。

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。例えば、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ものとします。

```
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'
```

このビューに対する以下の照会は、期待したようには効率的ではありません。

```
select firstnme, phoneno
from d11_employee
where lastname = 'BROWN'
```

照会書き直しの際、コンパイラーは、述部 lastname = 'BROWN' をビュー D11_EMPLOYEE にプッシュダウンします。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、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。この照会は、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するために設計されています。

```
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、各データベース・パーティションに各プロジェクトをブロードキャストできます。さらに、この副照会の評価を何回も行わなければならない。

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%')
```

- DIST_PROJS を EMPLOYEE 表と結合して、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)
```

- 書き直された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_proj 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
```

この照会では、プロジェクトごとの avg_comp (avg_per_proj) が計算されます。その後、EMPLOYEE 表が含まれるすべてのデータベース・パーティションに結果をブロードキャストできます。

コンパイラ書き直しの例: 結合された SQL/XQuery ステートメントの述部プッシュダウン:

リレーショナル SQL 照会を最適化するための基本的な手法の 1 つは、括弧で囲まれた照会ブロックの WHERE 文節の述部を、括弧で囲まれた下位の照会ブロック (例えば、ビュー) に移動するという方法です。それにより、データの事前のフィルタリングが可能になり、索引をより良い仕方で使用できる場合があります。

これはパーティション・データベース環境では一層重要です。これにより、データベース・パーティション間で送受信される必要のあるデータ量を事前のフィルター処理により減らすことができる可能性があるためです。

XQuery 内の述部または XPath フィルターを移動するために類似の手法を使用できます。可能な限りデータ・ソースに近いフィルタリング式を常時移動するのが、基本的な方針です。この最適化手法は SQL では述部プッシュダウン、XQuery では抽出プッシュダウン (フィルターおよび XPath 抽出用) と呼ばれています。

SQL と XQuery で採用されているデータ・モデルには違いがあるため、これら 2 つの言語間の垣根を越えて述部、フィルター、または抽出を移動する必要があります。SQL 述部を、意味的に等しいフィルターにトランスフォームして、それを XPath 抽出にプッシュダウンする際には、データ・マッピングとキャストに関する規則を考慮する必要があります。以下の例は、関連述部の XQuery 照会ブロックへのプッシュダウンを取り上げています。

カスタマー情報が含まれる、以下の 2 つの XML 文書について検討してください。

Document 1 <customer> <name>John</name> <lastname>Doe</lastname> <date_of_birth> 1976-10-10 </date_of_birth> <address> <zip>95141.0</zip> </address> </customer> <customer> <name>Jane</name> <lastname>Doe</lastname> <date_of_birth> 1975-01-01 </date_of_birth> <address> <zip>95141.4</zip> </address> </customer>	Document 2 <customer> <name>Michael</name> <lastname>Miller </lastname> <date_of_birth> 1975-01-01 </date_of_birth> <address> <zip>95142.0</zip> </address> </customer> <customer> <name>Michaela</name> <lastname>Miller</lastname> <date_of_birth> 1980-12-23 </date_of_birth> <address> <zip>95140.5</zip> </address> </customer>
--	--

例 - INTEGER 述部のプッシュ

次の照会を考えてみましょう。

```
select temp.name, temp.zip
  from xmltable('db2-fn:xmlcolumn("T.XMLDOC")'
    columns name varchar(20) path 'customer/name',
    zip integer path 'customer/zip'
  ) as temp
 where zip = 95141
```

T.XMLDOC で可能な索引を使用し、不要なレコードを事前にフィルターに掛けて除去するには、zip = 95141 述部を以下の同等の XPATH フィルター式に内部的に変換します。

```
T.XMLCOL/customer/zip[. >= 95141.0 and . < 95142.0]
```

コンパイラーは XML フラグメントのスキーマ情報を使用しないので、ZIP に整数のみが含まれるとは想定できません。この特定の XPath 抽出に、小数部分を持つ他の数値と、対応する double XML 索引が存在する可能性があります。XML2SQL キャストは、値を INTEGER にキャストする前に小数部分を切り捨てることによって、このトランスフォーメーションを処理します。プッシュダウン・プロシージャーでこの動作を反映する必要があり、述部は引き続き意味的に正しくなるように変更しなければなりません。

例 - DECIMAL(x,y) 述部のプッシュ

次の照会を考えてみましょう。

```
select temp.name, temp.volume
  from xmltable('db2-fn:xmlcolumn("T.XMLDOC")'
    columns name varchar(20) path 'customer/name',
    volume decimal(10,2) path 'customer/volume'
  ) as temp
 where volume = 100000.00
```

T.XMLDOC で可能な DECIMAL または DOUBLE 索引を使用し、INTEGER 型での処理と同様に不要なレコードを事前にフィルターに掛けて除去するには、volume = 100000.00 述部を以下の XPATH 範囲フィルター式に内部的に変換します。

```
T.XMLCOL/customer/volume[.>=100000.00 and .<100000.01]
```

例 - VARCHAR(n) 述部のプッシュ

次の照会を考えてみましょう。

```
select temp.name, temp.lastname
from xmltable('db2-fn:xmlcolumn("T.XMLDOC")'
  columns name varchar(20) path 'customer/name',
  lastname varchar(20) path 'customer/lastname'
) as temp
where lastname = 'Miller'
```

T.XMLDOC で可能な索引を使用し、不要なレコードを事前にフィルターに掛けて除去するには、lastname = 'Miller' 述部を同等の XPATH フィルター式に内部的に変換します。この式のハイレベルな表現は以下のとおりです。

```
T.XMLCOL/customer/lastname[.>= rtrim("Miller") and .<
  RangeUpperBound("Miller", 20)]
```

SQL の場合、末尾ブランクの処理方法が XPath または XQuery とは異なります。元の SQL 述部では、2 人の顧客の姓が『Miller』の場合、その一人 (Michael) に末尾ブランクがあるとしても区別されません。その結果、2 人の顧客が戻り、これは未変更の述部がプッシュダウンされる場合とは異なります。

これを解決するには、述部を範囲フィルターにトランスフォームします。

- RTRIM() 関数を使用して、比較値からすべての末尾ブランクを切り捨てると、最初の境界が作成されます。
- 2 番目の境界は、「Miller」以上の可能なすべてのストリングを調べて、「Miller」で始まるすべてのストリングが見つかるようにすることによって作成されます。したがって、元のストリングは、この 2 番目の境界を表す上限のストリングに置き換えられます。

コンパイラー書き直しの例: 暗黙の述部:

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、追加の表結合についても検討できるようになります。

以下の照会は、部門 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返すものです。

```
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
```

この照会は、*推移的閉包の述部* と呼ばれる以下の暗黙の述部を使用して書き直すことができます。

```
dept.mgrno = proj.respemp
```

これによって、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に考慮できる結合の種類が増えました。

照会書き直し段階の際、等価述部によって暗黙のうちに示される推移に基づいて、追加のローカル述部が得られます。例えば、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を返します。

```
select empno, lastname, firstname, deptno, deptname
  from employee emp, department dept
  where
    emp.workdept = dept.deptno and
    dept.deptno > 'E00'
```

この照会は、以下の暗黙の述部を使って書き直すことができます。

```
emp.workdept > 'E00'
```

このように書き直すと、結合する必要がある行の数が減ります。

例 - OR から IN への変換

OR 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 節の代わりに IN 述部を使用すると、照会がより効率的に処理されます。

```
select *
  from employee
  where deptno in ('D11', 'D21', 'E21')
```

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 節に変換して、索引 ORing を実行できるようにすることがあります。

述部の分類とアクセス・プラン

述部は、比較操作を明示する、または暗黙のうちに示す検索条件の 1 つの要素です。述部は、通常、照会の WHERE 節で指定され、照会によって返される結果セットの範囲を減らすために使用されます。

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。以下のリストは、それらのカテゴリーをパフォーマンスの高いものから順に示しています。

1. 範囲区切り述部
2. 索引 SARGable 述部
3. データ SARGable 述部
4. Residual 述部

SARGable (検索引数述部) は、検索引数 (search argument) として使用できる述部です。

表 56 は述部カテゴリーの特性の要約です。

表 56. 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	はい	いいえ	いいえ	いいえ
データ・ページ入出力の低減	はい	はい	いいえ	いいえ
内部的に渡される行数の低減	はい	はい	はい	いいえ
修飾行の数の低減	はい	はい	はい	はい

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ンの範囲を限定します。それは索引検索の開始および停止キー値を提供します。索引 SARGable 述部は、検索の範囲を限定できませんが、述部が参照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ことができます。例えば、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 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 and dept = :hv2) は範囲区切り述部であり、years > :hv5 は索引 SARGable 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、基本表を読み取るのではなく、索引データを使用します。索引 SARGable 述部によって、表から読み取る必要のある行数は少なくなりますが、アクセスする索引ページの数はありません。

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ものの、データ管理サービス (DMS) 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、表の個々の行へのアクセスを必要とします。必要なら、DMS は、述部を評価するために必要な列に加え、SELECT リストで必要なその他の列のうち索引から獲得できなかったものを取り出します。

例えば、PROJECT 表に定義されている次の索引を考えます。

```
INDEX IX0: PROJNO ASC
```

次の照会では、deptno = 'D11' はデータ SARGable 述部と見なされます。

```
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。このタイプの述部は以下の特性を持つ場合があります。

- 相関副照会を使用する
- ANY、ALL、SOME、または IN 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 (RDS) によって評価されません。

索引のみに適用される述部の中には、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ならないものもあります。例えば、索引 ORing 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。

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

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・データベースに対して実行される照会では、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・ソースで特定の操作を実行できるかどうかを判別します。

操作は、関係演算子、システム関数、ユーザー関数などの関数である場合もあり、例えば ORDER BY、GROUP BY などの SQL 演算子の場合もあります。

DB2 照会コンパイラーがリモート・データ・ソースにある SQL サポートについての正確な情報にアクセスできるようにするため、必ずローカル・カタログ情報を定期的に更新してください。カタログを更新するときには、DB2 データ定義言語 (DDL) ステートメント (例えば、CREATE FUNCTION MAPPING や ALTER SERVER など) を使用します。

関数リモート・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 サーバーにリモート・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演算子も、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。例えば、GROUP BY 演算子でリモート・データ・ソースをローカルに集約するなら、DB2 サーバーにリモート・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。

例えば、DB2 for z/OS のデータ・ソースにあるデータ・ソース表 EMPLOYEE を参照するニックネーム N1 について考えます。この表には 10 000 の行があり、列の 1 つには従業員の姓が、そして別の列には給与が入っています。オプティマイザーが次のステートメントを処理するとき、ローカルとリモートの照合シーケンスが同じであるかどうかに応じて、いくつかのオプションがあります。

```
select lastname, count(*) from n1
  where
    lastname > 'B' and
    salary > 50000
  group by lastname
```

- 照合シーケンスが同じ場合、照会の述部が DB2 for 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、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、両方の述部をデータ・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 をプッシュダウンする方法をとる場合があります。その場合でも、範囲の比較はローカルに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 操作をローカルに実行するのが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リモート・データ・ソース上で関数や演算子を評価させることにあります。関数や SQL 演算子をリモート・データ・ソースで評価できるかどうかは、次のような多くの要素に左右されます。

- 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 製品で豊富な SQL ダイアレクトがサポートされているためです。DB2 データ・サーバーは別のデータ・サーバーで使用できる機能の不足を補正することができます、そのようにすると、その操作は DB2 サーバー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。例えば、ほとんどのデータ・ソースは GROUP BY 演算子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 サーバーは 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。例えば一部のデータ・ソースでは、パラメーター・マーカをリモート 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 サーバー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 サーバーでは、リモート・データ・ソースで許可されている整数よりも大きい整数を使用できる場合がありますが、リモートでの限度を超える値をデータ・ソースに送られるステートメントに組み込むことはできませんし、影響を受ける関数または演算子はローカルに評価する必要があります。

- 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。例えば、データ・ソースにある NULL 値のソート方法が DB2 サーバーのソート方法と異なる場合には、NULL 可能な式での ORDER BY 操作はリモートでは評価できません。

- 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。データ・ソースが使用するも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てから、COLLATING_SEQUENCE サーバー・オプションを Y に設定した場合、オプティマイザーは、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っていると、異常な結果が発生する可能性があります。比較の際に大/小文字を区別しないデータ・ソースにステートメントをサブミットすると、予期しない結果が返される場合があります。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 サーバー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。例えば、DB2 for z/OS では、ORDER BY 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 節でソートされた DB2 for 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 サーバー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。

ユーザーは、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に従って配列された照会結果を必要としているため、データをローカルに配列することによってこの必要に応えます。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

COLLATING_SEQUENCE、VARCHAR_NO_TRAILING_BLANKS、PUSHDOWN などのいくつかのサーバー・オプションが、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。

- DB2 タイプ・マッピングおよび関数マッピングの要因

DB2 サーバー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・マッピングをカスタマイズすることもできます。例えば、Oracle データ・ソース列の 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 サーバー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 関係演算子が挙げられます。ほとんどのデータ・ソースはこの演算子をサポートしていますが、列名は IS NULL 演算子の左辺にしか使用できないなど、制限が存在する場合があります。
- 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、より大きい (>) 演算子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、「より大きい」演算子が DB2 サーバー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。例えば、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・ソースでは行えません (結合列が Oracle 整数列より短いため)。ただし、この Oracle 整数列の領域を DB2 INTEGER データ・タイプに合わせられる場合は、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、ALTER NICKNAME ステートメントを使用します。

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。その後コンパイラのプッシュダウン分析ステップで、そのような列に対して実行するすべての操作を検査するときに、この情報が考慮されます。DB2 サーバーは、データ・ソースに送信される SQL ステートメントで使用するための、異なっても等価な形式の述部を生成する場合があります。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。

表 57 は、これらのオプションについて説明しています。

表 57. 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	<p>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。列データのソートを妨げる可能性があるブランク文字は含まれません。このオプションは、データ・ソースの照合シーケンスが DB2 サーバー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。この列に、末尾ブランク文字が付いた数値ストリングしか含まれない場合は、Y は指定しないでください。</p> <p>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。</p>	N

表 57. 列オプションとその設定値 (続き)

オプション	有効な設定値	デフォルト設定
VARCHAR_NO_TRAILING_BLANKS	<p>Y: このデータ・ソースが、DB2 データ・サーバーのように、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ことを指定します。末尾ブランク文字を含んでいない可変長文字ストリングについては、一部のデータ・サーバー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 表またはビューの列に末尾ブランク文字が含まれていないことが明らかである場合は、この値を指定してください。</p> <p>N: このデータ・ソースが、DB2 データ・サーバーのように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ことがないことを指定します。</p>	N

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 演算子を参照できます。セット演算子 (例えば UNION) などの 1 つの演算子を使用して、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 サーバーで実行する必要があります。この演算子は、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。

フェデレーテッド照会がどこで評価されるかを判別するためのガイドライン:

db2expln コマンドを呼び出すことにより開始できる DB2 Explain ユーティリティにより、照会がどこで評価されるかが示されます。各演算子の実行位置は、コマンド出力に含まれています。

- 照会をプッシュダウンする場合は、DB2 の標準的な演算子である RETURN 演算子が示されるはずですが、SELECT ステートメントがニックネームからデータを検索する場合は、フェデレーテッド・データベースの操作だけで使用される SHIP 演算子も示されます。この演算子はデータ・フローのサーバー・プロパティを変更し、リモート演算子とローカル演算子を区別します。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。
- INSERT、DELETE、または UPDATE ステートメントが完全にリモート・データ・ソ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 演算子が示され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、または DELETE ステートメントは、RETURN 演算子で示されます。ただし、照会全体をプッシュダウンすることができない場合は、SHIP 演算子に、どの操作がリモート側で実行されたかが示されます。

照会が DB2 サーバーによってではなく、データ・ソースで評価される理由

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?

選択範囲が非常に狭い述部を使用して行をフィルターに掛ければネットワーク・トラフィックを減らせる場合に、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 演算子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演算子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? 関係演算子は、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? そのようにすると、このデータ・ソースでの SQL 制限に違反しませんか?
- 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 演算子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。

- GROUP BY 演算子への入力の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、この演算子についての何らかの制約事項がありますか? 例えば、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 演算子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 節には文字式が含まれているか? 含まれている場合は、大/小文字の区別がリモート・データ・ソースと DB2 サーバーで同じであることを確認してください。

- セット演算子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。

- それぞれのオペランドはどちらも、同じリモート・データ・ソースでオペランド全体が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、このセット演算子についての何らかの制約事項がありますか? 例えば、ラージ・オブジェクト (LOB) または LONG フィールド・データは、この特定のセット演算子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?

調べる必要のある分野は、以下のものがあります。

- ORDER BY 演算への入力の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 節には文字式が含まれていますか? 含まれている場合は、照合シーケンスまたは大/小文字の区別がリモート・データ・ソースと DB2 サーバーで異なっていませんか?
- そのリモート・データ・ソースには、この演算子についての何らかの制約事項がありますか? 例えば、ORDER BY 項目の数が限定されていませんか? リモート・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?

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。そのようにリモート SQL を生成すると、照会のためのグローバルで最適化されたアクセス戦略を作成するのに役立ちます。

オブティマイザーは、プッシュダウン分析の出力を使用して、各操作が DB2 サーバー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 サーバーとリモート・データ・ソースの間でデータやメッセージを送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、以下の要素はグローバルな最適化に大きな影響があります。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・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・サーバーのオプションには、次のようなものがあります。

- 処理速度の相対比率

データ・ソースの処理速度が DB2 サーバーの処理速度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。比率が低い場合、データ・ソースの処理速度は DB2 サーバーの処理速度より早いことを示します。その場合、DB2 オプティマイザーは、プロセッサへの負荷が大きい操作をデータ・ソースにプッシュダウンすることを考慮する可能性が高くなります。

- 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 サーバーのシステム入出力速度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、IO_RATIO サーバー・オプションを使用します。比率が低い場合、データ・ソースの入出力速度は DB2 サーバーの入出力速度より早いことを示します。その場合、DB2 オプティマイザーは、入出力集中の操作をデータ・ソースにプッシュダウンすることを考慮する可能性が高くなります。

- DB2 サーバーとデータ・ソース間の通信速度

ネットワーク容量を指定するには、COMM_RATE サーバー・オプションを使用します。比率が低い (DB2 サーバーとデータ・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 に設定しなかった場合、DB2 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 サーバーはプラン・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・ソースにあるオプティマイザーに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・ソースにある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。例えば、Oracle オプティマイザーへ送信される、プラン・ヒントを使用するステートメントは、次のようになります。

```
select /*+ INDEX (table1, t1index)*/
      coll
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 サーバー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、特定のデータベース管理システム (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 サーバーは、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 サーバーで使用することができます。この理由から、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仕様の作成

ニックネームのための索引の仕様を作成できます。索引の仕様は、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。索引の仕様を作成するには `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仕様を作成する構文は、ローカル表で索引を作成する構文と似ています。索引の仕様の作成は、次のような場合に考慮してください。

- DB2 サーバーが、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、ネスト・ループ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、結合列に対して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。ただし、2 つの表を結合させて作成したビューのような、単純な `SELECT` ステートメントではないビューでローカルに索引を作成すると、照会のパフォーマンスは低下する場合があります。例えば、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、ネスト・ループ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くなります。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てから、両方のニックネームを参照するローカル・ビューを DB2 サーバー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 サーバーでは、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、そのデータに対する更新を自動的に検出することはできません。さらに DB2 サーバーは、データ・ソースのオブジェクトの定義に加えられた変更を自動的に検出することはできません。オブジェクトの統計データまたは定義に変更がある場合は、以下のようにできます。

- データ・ソースで **RUNSTATS** コマンドと同等の機能を実行し、現在のニックネームをドロップしてから、ニックネームを再作成します。この方法は、オブジェクトの定義が変更された場合に使用してください。
- **SYSSTAT.TABLES** カatalog・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、オブジェクトの定義が変更された場合には機能しません。

フェデレーテッド・データベース照会のグローバル分析:

db2expln コマンドを呼び出すことにより開始できる DB2 Explain ユーティリティにより、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・オブティマイザーで生成されるアクセス・プランが示されます。各演算子の実行位置は、コマンド出力に含まれています。

照会のタイプによって、SHIP または RETURN 演算子で各データ・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。各演算子の詳細を調べるなら、DB2 オプティマイザーが各演算子に関する入出力として見積もる行数が分かります。

DB2 最適化の決定

パフォーマンスを向上させる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。

- 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。リモート・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。

- **GROUP BY** 演算子がリモートに評価されていないのはなぜか?

その演算子の構文が、リモート・データ・ソースで評価できることを調べてください。

- 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、各演算子はリモート・データ・ソースで評価できることが示された場合でも、オプティマイザーはそれ自体が行ったコスト見積もりを利用してグローバル最適化プランを生成します。そのプランに参与する要因は非常にたくさんあ

ります。例えば、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るとしても、その処理速度が DB2 サーバーの処理速度よりはるかに遅いため、DB2 サーバー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 カタログ・ビュー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。このステートメントが元の照会と同一であることを確認します。述部の順序に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ないはずですが。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。DB2 サーバー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も調べることができます。適切な照会オプティマイザーであれば、述部が等価となるように置き換えても影響されることはないはずですが。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。例えば、オプティマイザーによっては、述部の推移的閉包ステートメントを生成できないものもあります。

- その他の関数。リモート SQL ステートメントに、元の照会には存在しない関数が含まれていませんか？ そのような関数の中には、データ・タイプを変換するために使用されるものがあるかもしれません。それらの関数が必要であることを確認してください。

データ・アクセス方式

SQL または XQuery ステートメントをコンパイルするとき、照会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。

これらの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 または XQuery 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 および XQuery 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 および XQuery ステートメントのアクセス・プランは、実行時に作成されます。

表のデータにアクセスするには次の 3 つの方法があります。

- 表全体を順番にスキャンする方法
- 表の索引にアクセスして特定の行を見つける方法
- スキャン・シェアリングによる方法

行は、通常 WHERE 節で記述される、述部で定義された条件にしたがってフィルター操作できます。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が生成されます。この出力データをさらにグループ化したりソートしたりして処理する場合もあります。

DB2 9.7 からは、スキャン時に別のスキャンのバッファ・プール・ページを使用する機能であるスキャン・シェアリングがデフォルトの動作になりました。スキャン・シェアリングによって、ワークロードの並行性とパフォーマンスが改善されます。スキャン・シェアリングを使用すると、システムはさらに多くの同時アプリケーションをサポートでき、照会のパフォーマンスが良くなり、システムのスループットが向上する可能性があるため、スキャン・シェアリングに関係しない照会さえも益が及びます。スキャン・シェアリングは、表スキャンまたは大規模な表のマルチディメンション・クラスタリング (MDC) ブロック索引スキャンなどのスキャンを実行するアプリケーションが含まれる環境では、特に有効です。コンパイラーは、特定のスキャンがスキャン・シェアリングに含めるのに適格であるかどうかを、スキャンのタイプ、その目的、分離レベル、レコードごとに実行される処理の量などに基づいて決定します。

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、基本表にアクセスする前に (索引の指定範囲内にある行をスキャンすることによって) 適格となる行の集合の範囲を狭めるため、出力を並べ替えるため、または要求した列データを直接取得するため (索引のみのアクセス)、データベース・マネージャーが索引にアクセスするときです。

索引の指定範囲にある行をスキャンするとき、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。索引のみのアクセスの場合には、要求されたデータはすべて索引内にあるため、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。

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。

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、またはパーティション索引 (特定の場面に順序を保持できない) が使用されるときに追加のソートが必要な場合です。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判別するには、DB2 Explain 機能を使用します。

範囲を制限す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 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 節内で比較操作を明示する、または暗黙のうちに示す検索条件の 1 つのエレメントです。以下の場合に、述部を使用して索引スキャンの有効範囲を制限できます。

- IS NULL または IS NOT NULL であるかどうかのテスト
- 狭義および広義の不等比較のテスト

- 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テストされます。
- 基本的な副照会 (つまり、ANY、ALL、および SOME のいずれも含まれない副照会) に対する等価性のテスト。この副照会には、その即時親照会ブロック (つまり、この副照会が副選択となる選択) への相関列参照を含めることはできません。

以下の例では、スキャンの範囲を制限するために索引を使用する方法を示します。

- 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
```

以下の述部を使用すると、索引 IX1 を使用するスキャンの範囲を制限できます。

```
where
  name = :hv1 and
  dept = :hv2
```

または

```
where
  mgr = :hv1 and
  name = :hv2 and
  dept = :hv3
```

2 番目の WHERE 節は、索引内でキー列が出現する順序で述部を指定する必要はないことを例示しています。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、式、または定数などの他の変数を代わりに使用することもできます。

以下の WHERE 節では、NAME および DEPT を参照する述部だけが索引スキャンの範囲を制限するために使用されます。

```
where
  name = :hv1 and
  dept = :hv2 and
  salary = :hv4 and
  years = :hv5
```

これらの列を最後の 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ります。しかし、name = :hv1 および dept = :hv2 の述部によって範囲が決まった後は、その他の述部を残りの索引キー列に対して評価できるようにする必要があります。

- 次のように ALLOW REVERSE SCANS オプションを使用して作成された索引を考えてみます。

```
create index iname on tname (cname desc) allow reverse scans
```

この場合、索引 (INAME) は、CNAME 列の降順の値に基づいています。索引のスキャンが降順で実行され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。索引の使用は、アクセス・プランを作成および考慮する時にオプティマイザーにより制御されます。

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制限できます。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

範囲を制限する述部として使用される狭義の不等比較演算子は、より大きい (>) とより小さい (<) です。

索引スキャンの範囲を制限す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制限することはできますが、その目的で MGR 列に対して述部を使用することはできません。

```
where
  name = :hv1 and
  dept > :hv2 and
  dept < :hv3 and
  mgr < :hv4
```

ただし、ジャンプ・スキャンを使用するアクセス・プランでは、索引スキャンの範囲を制限するために、狭義の不等比較述部が含まれている複数の列を考慮することができます。上記の例 (ジャンプ・スキャンを使用するアクセス・プランをオプティマイザーが選択していることを想定しています) では、DEPT における狭義の不等比較述部、および MGR 列を使用して範囲を制限することができます。この例では狭義の不等比較述部に焦点を当てていますが、範囲を制限するために NAME 列に対する等価述部も使用されていることに注意してください。

- 広義の不等比較述部

範囲を制限する述部として使用される広義の不等比較演算子を以下に示します。

- >= と <=
- BETWEEN
- LIKE

索引スキャンまたはジャンプ・スキャンの範囲を制限するために、広義の不等比較述部が使用されている複数の列を考慮できます。次の例では、すべての述部を使用して範囲を制限することができます。

```
where
  name = :hv1 and
  dept >= :hv2 and
  dept <= :hv3 and
  mgr <= :hv4
```

:hv2 = 404、:hv3 = 406、および :hv4 = 12345 であると想定します。データベース・マネージャーは部門 404 と 405 の索引をスキャンしますが、従業員番号 (MGR 列) が 12345 より大きい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。

データ並べ替えのための索引スキャン

照会がソートされた出力を必要としている場合、並べ替える列が、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、データを並べ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、および UNION などの操作の結果得られます。これについての例外は、以下のとおりです。

- 索引がパーティション化される場合、この索引をデータの並べ替えに使用できるのは、索引キー列に表パーティション・キー列の接頭部が付く場合か、パーティションの除去によって 1 つを除き、すべてのパーティションが除去される場合に限ります。
- 索引キー列が「定数値」、または評価結果が定数になる式に等しいかどうかをテストするとき、列の順序は、最初の索引キー列とは異なる場合があります。

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ト済みになるため、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 節と ORDER BY 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 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 が固有になるため、PROJNAME 列でさらに順序付けを行う必要はありません。各 PROJNO 値には、PROJNAME 値が 1 つしかありません。

ジャンプ・スキャン

複合 (複数列) 索引を持つ表に対する照会は、表の索引を設計する際に特に困難があります。理想としては、照会の述部が表の複合索引と整合していることが望ましい形です。このことは、それぞれの述部をスタート・ストップ・キーとして使用できることと、それによって、検索が必要な索引の範囲が狭められることを意味しています。照会の中に、複合索引とは不整合な述部が含まれている場合、これを索引ギャップといいます。そのように索引ギャップは、照会が表の索引とどの程度一致しているかを示す特性です。

制約のない索引ギャップ

このトピックの前の部分で定義した IX1 複合索引を持つ表に対して、次の照会を行う場合を考えてみます。

```
where
  name = :hv1 and
  dept = :hv2 and
  mgr = :hv3 and
  years = IS NOT NULL
```

この照会には、複合索引の SALARY キー部分に索引ギャップが含まれています (これは、アクセス・プランに、複合索引の索引スキャンが含まれていることを前提としています)。SALARY 列をスタート・ストップ述部として含めることはできません。SALARY 列は、制約のない索引ギャップ の一例です。

注: 照会によっては、索引ギャップがあるかどうかを評価するのが困難な場合があります。索引ギャップが存在するかどうかを調べるには、EXPLAIN 出力を使用してください。

制約のある索引ギャップ

このトピックの前の部分で定義した IX1 複合索引を持つ表に対して、次の照会を行う場合を考えてみます。

```
where
  name = :hv1 and
  dept = :hv2 and
  mgr = :hv3 and
  salary < :hv4 and
  years = :hv5
```

この照会には、複合索引の SALARY キー部分に索引ギャップが含まれています (これは、アクセス・プランに、複合索引の索引スキャンが含まれていることを前提としています)。照会に含まれる SALARY 列は等価述部ではなく、この列に対しては開始値/停止値を生成することはできません。SALARY キー部分は、制約のある索引ギャップ を表します。

索引ギャップがある照会でのパフォーマンスの低下を避けるために、オプティマイザはジャンプ・スキャン 操作を実行できます。ジャンプ・スキャン操作では、索引マネージャーがギャップのある複合索引の小さいセクションに対する修飾キーを識別し、これらのギャップに修飾キーを埋め込みます。その結果、索引マネージャーは、索引の中の、結果が得られない部分をスキップします。

ジャンプ・スキャンの制約事項

ジャンプ・スキャンが予期される照会を実行している場合、ターゲット表に適切な複合索引があること、および照会述部によって索引ギャップが生じていることを確認してください。DB2 オプティマイザは、索引ギャップがない場合にはジャンプ・スキャンを使用したプランを作成しません。

ジャンプ・スキャンは、以下のタイプの索引をスキャンしません。

- 範囲がクラスター化された表索引
- 拡張索引 (例えば、空間索引など)
- XML 索引
- テキスト索引 (Text Search 用)

ジャンプ・スキャンを使用すると、IN-LIST 述部のあるギャップ列は制約のないギャップ列として扱われることがあります。また、ユニコード照合アルゴリズム (UCA) を採用したデータベースでは、ジャンプ・スキャンは、ホ

スト変数またはパラメーター・マーカーを使用して、LIKE 述部を制約のないギャップとして扱うことがあります。

注: 照会を評価する際は、索引ギャップが存在する場合でも、オプティマイザーはジャンプ・スキャン操作を含まないアクセス・プランを選択する場合があります。これは、オプティマイザーにより、ジャンプ・スキャンを使用しないほうがより効率的だと判断された場合に発生します。

索引アクセスのタイプ

照会が必要としているデータすべてが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。範囲がクラスター化された表の場合は、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。例えば、次の索引定義を考えてみてください。

```
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、必要な列が索引内がないことがよくあります。それらの列からデータを取り出すには、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、**include** 列を持つユニーク索引を作成します。例えば、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** 列をユニークとし、さらに **DEPT**、**MGR**、**SALARY**、および **YEARS** 列のデータも保管して保守します。このようにして、以下の照会は索引にアクセスするだけで達成できます。

```
select name, dept, mgr, salary
from employee
where name = 'SMITH'
```

include 列を含める場合には、追加のストレージ・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慮してください。 **include** 列を使用する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。

複数の索引へのアクセス

WHERE 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。例えば、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 述部を満たすレコード 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去されます。これは、索引 *ORing* と呼ばれます。

索引 *ORing* 操作は、次の例のように IN 節で指定されている述部でも使用することもできます。

```
where
  dept in (:hv1, :hv2, :hv3)
```

索引 *ORing* 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するアプリケーションが、複数の AND 述部がある照会をこの表に対して実行する場合に行われることがあります。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。1 番目のビットマップを厳密に調べるために 2 番目のビットマップが使用され、最終的な結果セットに対して適格となる行が生成されます。例えば、以下のような索引があるとします。

```
INDEX IX4: SALARY  ASC
INDEX IX5: COMM    ASC
```

これを使用して次の述部を解決できます。

```
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プを行うと、両方の述部を満たす、適格となる RID のリストが生成されます。これは、動的ビットマップ *ANDing* と呼ばれます。これは、表に十分なカーディナリティがあるか、その列の適格となる範囲内に十分な値があるか、または、等価述部が使用される場合に十分な重複があるときにのみ行われます。

複数の索引をスキャンするとき動的ビットマップを使用してパフォーマンス上の利点を実現するためには、**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ります。

るかもしれません。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、並行照会の割合が 2、3 回の係数よりも少ない) 場合、複数索引アクセスに伴う動的ビットマップは、最適化ierer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。

最適化iererは、単一の表にアクセスするのに、索引 ANDing 操作と索引 ORing 操作を組み合わせることはしません。

範囲がクラスター化された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、範囲がクラスター化された表には不要です。その代わりに、列ドメインが持つ順次性を利用し、表内の特定の行の位置を生成するために関数マッピングを使用します。このタイプのマッピングの最も単純な例では、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、それ以降も同様になります。

最適化iererは表の範囲クラスター・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。その際のコストは範囲クラスター関数の計算だけです。範囲がクラスター化された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。

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、最適化iererはディスクからページをバッファ・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・プール使用率の予測も含まれています。既にバッファ・プールの中にあるページから行を読み取るために追加の入出力が必要ないためです。

索引スキャンの場合、最適化iererは、システム・カタログの情報を使用して、データ・ページをバッファ・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。それは、SYSCAT.INDEXES ビュー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、最適化ierer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** 列には **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・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、**DETAILED** 節を指定した **RUNSTATS** コマンドを索引に対して実行した場合だけ集められます。

索引のクラスターリング統計が使用不可である場合、最適化ierer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ーリング率が低いことを想定しています。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、表に対して定義する索引の 1 つについ

ては、クラスター化が 100% 近くに維持されるようにしてください。一般的に、索引のキーがクラスター索引のキーのスーパーセットを表す場合と 2 つの索引のキー列間に実際の相関がある場合を除いて、100% クラスターリングできるのは 1 つの索引だけです。

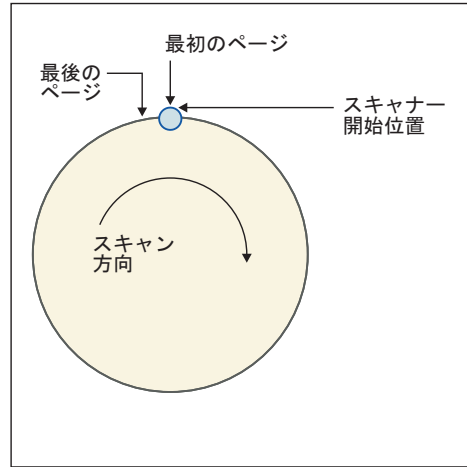
表を再編成するときに、行をクラスター化するために使用する索引を指定し、挿入処理中にそのクラスター化を保持することができます。更新および挿入操作を行うと索引に対する表のクラスター化が低下する場合がありますため、定期的に表を再編成することが必要な場合があります。挿入、更新、または削除操作が頻繁に発生する表に対する再編成の回数を少なくするには、ALTER TABLE ステートメントで PCTFREE 節を指定します。

スキャン・シェアリング

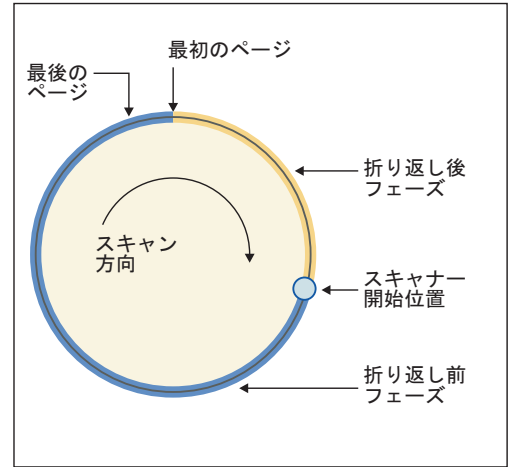
スキャン・シェアリングとは、他のスキャンによる処理を利用するスキャン機能のことです。共有作業の例としては、ディスク・ページ読み取り、ディスク・シーク、バッファ・プール内容の再利用、圧縮解除などがあります。

大きな表の表スキャンまたはマルチディメンション・クラスターリング (MDC) ブロック索引スキャンなどの負荷の大きなスキャンの場合、他のスキャンとページ読み取りを共有するのが適していることがあります。そうした共有スキャンは表内の任意の点から開始して、既にバッファ・プール内にあるページを活用できます。共有スキャンが表の最後に到達すると、先頭から継続され、開始点に到達すると完了します。これは、ラッピング・スキャンと呼ばれます。281 ページの図 25 は、表と索引の両方に関する通常のスキャンとラッピング・スキャンの違いを示しています。

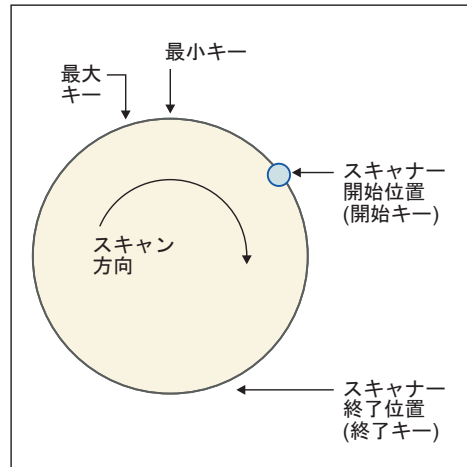
通常の表スキャン



折り返し表スキャン



通常の索引スキャン



折り返し索引スキャン

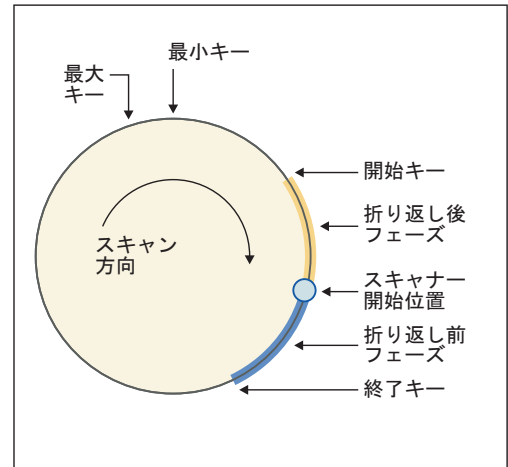


図 25. 通常のスキャンとラッピング・スキャンの概念ビュー

デフォルトでスキャン・シェアリング・フィーチャーは有効にされていて、スキャン・シェアリングと折り返しが適格かどうかは SQL コンパイラーによって自動的に判別されます。実行時に、コンパイル時には把握されていなかった要因に基づいて、適格なスキャンが共有または折り返しを行うこともあれば、行わないこともあります。

共有スキャナーは、共有グループで管理されます。こうしたグループは、共有の利点を最大化できるように、可能な限り長期間に渡りそのメンバーを一緒に維持します。他のスキャンより速いスキャンがあると、ページ共有の利点が失われる可能性があります。そのような場合、最初のスキャンがアクセスするバッファ・プール・ページが、その共有グループ内の他のスキャンがアクセスできるようになる前に、バッファ・プールから消去されてしまうことがあります。データ・サーバーは、同じ共有グループ内にある 2 つのスキャンの距離を、それらの間にあるバッファ・プール・ページ数によって測定します。またデータ・サーバーは、スキャンの速度をモニターします。同じスキャン・グループ内にある 2 つのスキャンの距離が長くなり過ぎる場合には、バッファ・プール・ページを共有できない恐れがあります。こうした可能性を減らすには、速度の速いスキャンをスロットル化して、データ・ページが消去される前に遅いスキャンがそうしたページにアクセスできる

ようにします。図 26 は、1 つは表用、もう 1 つはブロック索引用の 2 つの共有セットを示しています。共有セットは、同じアクセス・メカニズム (例えば、表スキャンまたはブロック索引スキャン) を使用して同じオブジェクト (例えば、表) にアクセスしている共有グループの集合です。表スキャンの場合、ページ読み取り順序はページ ID の昇順で、ブロック索引スキャンの場合、ページ読み取り順序はキー値の昇順です。

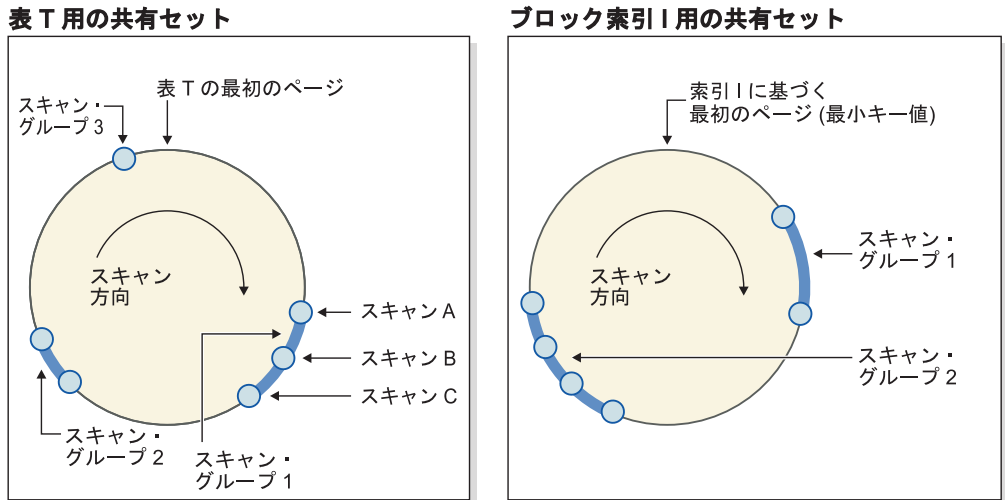


図 26. 表とブロック索引のスキャン・シェアリングの共有セット

またこの図は、バッファ・プール内容を複数のグループで再利用する方法も示しています。グループ 1 の先行スキャンであるスキャン C について考えます。後続スキャン (A と B) は C にグループ化されます。それらのスキャンは近接していて、C がバッファ・プールに入れたページを再利用する可能性が高いからです。

優先度の高いスキャナーは優先度の低いスキャナーによってスロットルされることはなく、その代わりに別の共有グループに移動する可能性があります。優先度の高いスキャナーは、グループ内の優先度の低いスキャナーが実行中の作業から益を受けられるグループ内に配置できます。それは、その益が受けられる限りはそのグループ内に留まります。速度の速いスキャナーをスロットルするか、(スキャナーが検出する場合に) より速い共有グループに移動して、データ・サーバーはその共有が最適化された状態を保つように共有グループを調整します。

スキャン・シェアリングに関する情報を表示するには、**db2pd** コマンドを使用できます。例えば、個々の共有スキャンの場合、**db2pd** 出力にはスキャン速度、およびスキャンがスロットルされた時間などのデータが表示されます。共有グループの場合、このコマンド出力にはグループ内のスキャン数、グループによって共有されるページ数が表示されます。

EXPLAIN_ARGUMENT 表には、表スキャンと索引スキャンについてのスキャン・シェアリング情報が含まれる新しい行があります。この表の内容をフォーマット設定および表示するには **db2exfmt** コマンドを使用できます。

オプティマイザ・プロファイルを使用すると、コンパイラがスキャン・シェアリングに関して下した決定をオーバーライドできます (『アクセスのタイプ』を参照してください)。このようなオーバーライドは、特別な必要性が生じた場合のみ

使用します。例えば、結果セット内のレコードの反復可能順序が必要なものの、ORDER BY 文節 (ソートを起動する必要がある) を使用しない場合には、折り返しヒントが役立ちます。その他の場合には、DB2 Service によって要求されない限りは、こうした最適化プロファイルを使用しないようお勧めします。

照会の述部の処理

述部とは、比較演算を表す、あるいは暗黙のうちに示す検索条件の 1 つの要素です。述部は、その述部が評価プロセスにおいていつ、どのように用いられるかによって 4 つのカテゴリーに分類できます。そのカテゴリーは以下のリストに示すとおりです (パフォーマンスの高いものから順に配列しています)。

- 範囲区切り述部は、索引スキンの範囲を限定するために使用される述部です。これらは、索引検索のための開始キーまたは停止キーの値を指定します。これらの述部は、索引マネージャーによって評価されます。
- 索引検索指数述部は、検索の範囲を限定するためには使用されず、選択された場合に索引から評価されます。これは、この述部に関係する列が索引キーの一部であるためです。これらの述部も、索引マネージャーによって評価されます。
- データ検索指数述部は、索引マネージャーでは評価できない述部ですが、データ管理サービス (DMS) によって評価できます。通常、これらの述部には、基本表の個々の行へのアクセスが必要となります。必要に応じて DMS は、述部を評価するために必要な列を取り出し、さらに SELECT リストの列を満たすもののうち索引からは取得できなかったものを取り出します。
- 残余述部は、基本表の単純アクセスの域を越えた入出力が必要となる述部です。残余述部の例には、定量化された副照会 (ANY、ALL、SOME、または IN を含む副照会) を使用する述部や、表とは別に保管されているラージ・オブジェクト (LOB) のデータを読み取る述部などがあります。これらの述部はリレーショナル・データ・サービス (RDS) によって評価され、述部の 4 つのカテゴリーの中で最もコストがかかります。

次の表では、さまざまな述部の例と、使用されるコンテキストに基づくそれらのタイプを示しています。

注: この例では、(c1, c2, c3) に複数列の昇順の索引が存在し、それが必要に応じて述部の評価に使用されることを想定しています。索引内のいずれかの列が降順である場合は、範囲区切り述部の開始キーと停止キーが入れ替わる可能性があります。

表 58. 各種の照会の述部の処理

述部	列 c1	列 c2	列 c3	コメント
c1 = 1 and c2 = 2 and c3 = 3	範囲区切り (開始-停止)	範囲区切り (開始-停止)	範囲区切り (開始-停止)	索引のすべての列についての等号述部を、開始-停止キーとして適用できます。
c1 = 1 and c2 = 2 and c3 ≥ 3	範囲区切り (開始-停止)	範囲区切り (開始-停止)	範囲区切り (開始)	列 c1 と c2 は等号述部によってバインドされ、c3 についての述部のみが開始キーとして適用されます。

表 58. 各種の照会の述部の処理 (続き)

述部	列 c1	列 c2	列 c3	コメント
c1 ≥ 1 and c2 = 2	範囲区切り (開始)	範囲区切り (開始-停止)	該当なし	先行列 c1 は ≥ 述部を持つ ので、開始キーとして使用で きます。後続の列 c2 は等 号述部を持つので、これも開 始-停止キーとして適用でき ます。
c1 = 1 and c3 = 3	範囲区切り (開始-停止)	該当なし	索引検索指数 述部	c2 についての述部がないの で、c3 についての述部は開 始-停止キーとして使用でき ません。しかし、索引検索指 数述部として適用できます。
c1 = 1 and c2 > 2 and c3 = 3	範囲区切り (開始-停止)	範囲区切り (開始)	索引検索指数 述部	前の列が > 述部を持つてい るので、c3 についての述部 は開始-停止述部として適用 できません。仮にこれが ≥ であったなら、これは開始- 停止キーとして使用できま す。
c1 = 1 and c2 ≤ 2 and c4 = 4	範囲区切り (開始-停止)	範囲区切り (停止)	データ検索指 数述部	この例では、c2 についての 述部が ≤ 述部です。これは 停止キーとして使用できま す。c4 についての述部はこ の索引には適用できず、 FETCH の間にデータ検索指 数述部として適用されます。
c2 = 2 and UDF_with_ external_ action(c4)	該当なし	索引検索指数 述部	残余	先行列 c1 は述部を持たない ので、c2 の述部は索引検索 指数述部として適用でき、こ の場合は索引全体がスキャン されます。外部アクションを 含むユーザー定義関数が関係 する述部は、残余述部として 適用されます。
c1 = 1 or c2 = 2	索引検索指数 述部	索引検索指数 述部	該当なし	OR があるため、この複数列 索引は開始-停止キーとして 使用できません。仮に 2 つ の索引があつて、一方の先行 列が c1、もう一方の先行列 が c2 であり、なおかつ DB2 オプティマイザーが 「索引の ORing」プランを 選択していたとすれば、その ように使用することもできま す。しかし、この例では、2 つの述部は索引検索指数述部 として扱われます。

表 58. 各種の照会の述部の処理 (続き)

述部	列 c1	列 c2	列 c3	コメント
c1 < 5 and (c2 = 2 or c3 = 3)	範囲区切り (停止)	索引検索指数 述部	索引検索指数 述部	この例では、先行列 c1 を利用し、停止キーによって索引スキャンによる述部の使用を停止しています。c2 と c3 についての OR 述部は、索引検索指数述部として適用されます。

DB2 オプティマイザーは、多数の複雑なユーザー作成述部を、以下の表で示すような、よりパフォーマンスの良い照会にトランスフォームするための照会書き直し機構を採用しています。

表 59. 照会書き直しの述部

元の述部または照会	最適化された述部	コメント
c1 between 5 and 10	c1 ≥ 5 and c1 ≤ 10	BETWEEN 述部は、それと同等の範囲区切り述部に書き直されて、ユーザーが範囲区切り述部を指定した場合と同様に内部で使用できるようになります。
c1 not between 5 and 10	c1 < 5 or c1 > 10	OR 述部がある場合は、DB2 オプティマイザーが索引の ORing のプランを選択していない限り、開始-停止キーは使用できません。
SELECT * FROM t1 WHERE EXISTS (SELECT c1 FROM t2 WHERE t1.c1 = t2.c1)	SELECT t1.* FROM t1 EOJOIN t2 WHERE t1.c1= t2.c1	副照会は結合にトランスフォームされる可能性があります。
SELECT * FROM t1 WHERE t1.c1 IN (SELECT c1 FROM t2)	SELECT t1.* FROM t1 EOJOIN t2 WHERE t1.c1= t2.c1	これは、上の行の EXISTS 述部のトランスフォーメーションの例と類似しています。
c1 like 'abc%'	c1 ≥ 'abc X X X ' and c1 ≤ 'abc Y Y Y'	索引の先行列が c1 の場合、DB2 はこれらの述部を生成して、範囲区切りの開始-停止述部として適用できるようにします。この例で、文字 X と Y は、最低と最高の照合文字を表しています。
c1 like 'abc%def'	c1 ≥ 'abc X X X ' and c1 ≤ 'abc Y Y Y' and c1 like 'abc%def'	これは前の例と類似していますが、さらに元の述部を索引検索指数述部として適用する必要があるという点が異なります。そうすることにより、「def」の文字が正しく検出されるようにします。

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のデータを組み合わせるプロセスのことです。1つの表の行は、対応する行にある情報が結合基準（結合述部）に基づいて合致する場合に、別の表の行と組になります。

例えば、次の2つの表を考えてみてください。

TABLE1		TABLE2	
PROJ	PROJ_ID	PROJ_ID	名前
A	1	1	Sam
B	2	3	Joe
C	3	4	Mary
D	4	1	Sue
		2	Mike

表の PROJ_ID 列が同じ値になるように 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
```

この場合、適切な結合述部は where x.proj_id = y.proj_id です。

照会により、次の結果セットが生成されます。

PROJ	PROJ_ID	名前
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

結合述部の特性、また表と索引の統計に基づいて判別されるコストに応じて、オプティマイザーは以下の結合方式のいずれかを選択します。

- 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つの表は結合の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、結合の種類および使用できる索引によって異なります。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演算子を指定して、結合で表をどう使用するかを決定することができます。ただし、この方法で照会を変更する前に、表の結合方法をオプティマイザーで判別してみると良いでしょう。それから、照会のパフォーマンスを分析して結合演算子を追加するかどうかを決定します。

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、ネスト・ループ結合、マージ結合、またはハッシュ結合という 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

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

例えば、表 T1 の列 A と表 T2 の列 A の値が次のようになっています。

外部表 T1: 列 A	内部表 T2: 列 A
2	3
3	2
3	2
	3
	1

表 T1 と T2 の間のネスト・ループ結合を完了するために、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 の最初の行を読み取ります。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. 表の終わりに達するまで、ステップ 2 を繰り返します。
 4. T1 に戻って、次の行 (3) を読みます。
 5. 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。
 6. 表の終わりに達するまで、ステップ 5 を繰り返します。
 7. T1 に戻って、次の行 (3) を読みます。
 8. 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使用できます。

```
expr(outer_table.column) relop inner_table.column
```

ここで、relop は比較演算子 (例えば、=、>、>=、<、または <=) であり、expr は外部表で有効な式です。例えば、

```
outer.c1 + outer.c2 <= inner.c1  
outer.c4 < inner.c3
```

この方法を使用すると、その効果の程度は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ソートするならば、ページが既にバッファー・プール内に存在している可能性が高くなるため、デ

ディスク上のページにアクセスするための内部表に対する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、アクセスする索引ページの数をもっと減らすことができます。

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、またはマージ結合操作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

マージ結合 (マージ・スキャン結合 あるいはソート・マージ結合 と呼ばれる) 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。

例えば、表 T1 の列 A と表 T2 の列 A の値が次のようになっています。

外部表 T1: 列 A	内部表 T2: 列 A
2	1
3	2
3	2
	3
	3

表 T1 と T2 の間のマージ結合を完了するために、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 の最初の行を読み取ります。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. 列が一致する間は T2 のスキャンを続け、行を結合します。
4. T2 内で 3 を読んだら、T1 に戻って次の行を読み取ります。
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、行を結合します。
7. T2 の終わりに達したら T1 に戻り、次の行を取得します。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` という形式の述部が 1 つ以上必要で、それらの列はいずれも LONG フィールド列や LOB 列にすることはできません。

ハッシュ結合は、次のようにして実行されます。まず指定された内部表をスキャンし、**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セクションに分割されています。内部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、選択したセク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、外部表) 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、内部表からの行と突き合わされます。外部の行の列のハッシュ値が内部の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表の部分に対応した外部表の行は、メモリー内の内部表の行と直ちに突き合わされます。対応する内部表の部分が一時表に書き込まれている場合は、外部の行も一時表に書き込まれます。最後に、一致している表の部分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。

ハッシュ結合処理では、フィルターを使用してパフォーマンスを改善することもできます。オプティマイザーがフィルターの使用を選択し、かつ十分なメモリーを使用できる場合、結合による処理を必要とする外部表の行数が、フィルターの使用により少なくなることがしばしばあります。

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・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最高になります。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。

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。照会の最適化クラスによって決定されるこうした方式の中には、いくつかの検索方法、スター・スキーマ結合、アーリーアウト結合、および複合表があります。

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。

- 貪欲型結合列挙
 - スペースおよび時間の所要量の観点から効率的です。
 - 単一方向列挙を使用します。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、欲張り型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の推移的閉包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
 - スペースおよび時間の所要量の観点から効率的ではありません。結合する表の数が増えると、それらも幾何級数的に増えます。
 - 最適なアクセス・プランの検索には、効率的で包括的な方法です。
 - DB2 for z/OS が使用する方法与似ています。

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 つの表のデカルト積が形成されます。デカルト積では、最初の表の適格となる各行が 2 番目の表の適格となる各行に結合されます。これにより、通常は非常に大きい結果表が作成されます。そのサイズは 2 つのソース表のサイズの乗積となるからです。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、オブティマイザーはこのタイプのアクセス・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他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デカルト積を計算してから、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 データ・サーバーは、少なくとも 2 つのディメンション表を持つスター・スキーマを使って設計されたデータベースに対する照会を認識することができ、ディメンション表のデカルト積を計算するプランの候補が含まれるように検索スペースを大きくすることができます。ジグザグ結合は、デカルト積がマテリアライズされる前であっても考慮されます。複数列索引を使用してファクト表を厳密に調べるため、ファクト表は 2 つ以上のディメンション表を使って同時にフィルターされます。ジグザグ結合が使用されると、ファクト表の索引から次に使用できる値の組み合わせが返されます。次に使用できるこの値の組み合わせはフィードバックと呼ばれ、ファクト表で一致が検出されないディメンション表のデカルト積を使って求められるプローブ値をスキップオーバーするために使用されます。2 つ以上のディメ

ンション表でファクト表を同時にフィルターすると共に、非生産的であると分かっているプローブをスキップすることにより、ジグザグ結合が大きなファクト表の照会に対して効率的な方法となります。

このスター・スキーマ結合方式では、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系列索引で、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・スキーマ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
 - レコード 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 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。ファクト表とディメンション表の間の明示的な参照整合性制約は必要ではありませんが、設定することが推奨されています。

スター・スキーマ結合技法で作成され使用される動的ビットマップにはソート・ヒープ・メモリーが必要ですが、そのサイズは **sortheap** データベース構成パラメーターで指定します。

アーリーアウト結合

オプティマイザーは、表の 1 つからの各行が、他の表からの最大でも 1 行と結合する必要しかないことを検出する場合、アーリーアウト結合を選択できます。

アーリーアウト結合は、表の 1 つのキー列に結合述部がある場合に可能です。例えば、従業員とその直接の上司の名前を返す以下の照会を考慮してください。

```
select employee.name as employee_name,  
       manager.name as manager_name  
from employee as employee, employee as manager  
where employee.manager_id = manager.id
```

ID 列が EMPLOYEE 表内のキーであり、すべての従業員に最大 1 人の上司がいると想定すると、この結合により MANAGER 表内の後続の一致行の検索が回避されます。

アーリーアウト結合は、照会内に DISTINCT 節がある場合も可能です。例えば、\$30000 を超える価格のモデルを販売する自動車メーカーの名前を返す以下の照会を考えます。

```

select distinct make.name
  from make, model
 where
   make.make_id = model.make_id and
   model.price > 30000

```

自動車メーカーごとに、その製造モデルのいずれかの価格が \$30000 を超えるかどうかを判別する必要があるだけです。自動車メーカーと、価格が \$30000 を超えるそのメーカーのすべての製造モデルとの結合は、照会結果の正確さに貢献しないため、不必要です。

アーリーアウト結合も、結合が GROUP BY 節に MIN または MAX 集約関数をフィードする場合は可能です。例えば、以下の照会について考えてみましょう。これは、2000 年より前に特定の株の終値がその始値よりも 10% 以上高くなった銘柄記号と、そのような日付のうち最も後のものものを戻します。

```

select dailystockdata.symbol, max(dailystockdata.date) as date
  from sp500, dailystockdata
 where
   sp500.symbol = dailystockdata.symbol and
   dailystockdata.date < '01/01/2000' and
   dailystockdata.close / dailystockdata.open >= 1.1
 group by dailystockdata.symbol

```

適格となる集合は、日付と株価の要件を満たし、SP500 表の特定の銘柄記号と結合される DAILYSTOCKDATA 表の行の集合です。DAILYSTOCKDATA 表の適格となる集合 (SP500 表の各銘柄記号行について) が DATE に基づいて降順で配列されている場合、最初の行が特定の記号の最新の日付であるので、必要なのは各記号の適格となる集合の最初の行を返すことだけです。適格となる集合に含まれる他の行は必要ありません。

複合表

一对の表を結合した結果が新規表 (複合表 と呼ばれる) になる場合、通常この表は別の内部表との結合の外部表になります。これは、複合外部結合 と呼ばれます。ある場合、特に欲張り型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 と呼ばれます。例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)、次に T3 と T4 を結合し (T3xT4)、最後に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) が複合内部結合 表となります。照会最適化クラスに従って、オブ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。

パーティション・データベース環境の複製されたマテリアライズ照会表

複製マテリアライズ照会表 (MQT) 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。

この文脈での複製された MQT とは、データベース内複製に関連した用語である点に注意してください。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。

次の前提条件で以下の例を考えます。

- SALES 表は、REGIONTABLESPACE という名前の複数パーティション表スペースにあり、REGION 列で分割されています。
- EMPLOYEE 表および DEPARTMENT 表が単一パーティションのデータベース・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 MQT 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
```

複製された MQT の内容を次のように更新します。

```
refresh table r_employee
```

REFRESH ステートメントを使用した後は、複製表に対して (他の表に対して行う場合と同様に) runstats ユーティリティを実行する必要があります。

次の照会は、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
```

データベース・マネージャーは、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、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 MQT である R_EMPLOYEE を使用します。結合を行うときに、ネットワークを介して従業員の情報を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。

コロケートッド結合における複製マテリアライズ照会表

複製 MQT は結合のコロケーションでも助けになります。例えば、スター・スキーマに 20 のデータベース・パーティション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ト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コロケート結合では使用できません。それは、ファクト表の結合列がファクト表の分散キーに対応していないためです。

C1 で分割された FACT (C1、C2、C3、...) という表、C1 で分割された DIM1 (C1、dim1a、dim1b、...) という表、C2 で分割された DIM2 (C2、dim2a、dim2b、...) という表 (以下同様に続く) がある場合を考えます。この場合、述部 $\text{dim1.c1} = \text{fact.c1}$ は連結できるので、FACT と DIM1 の間の結合は完全です。これらの表は両方とも C1 列で分割されています。

しかし、DIM2 と述部 $\text{dim2.c2} = \text{fact.c2}$ が関係する結合は連結できません。これは FACT が、C2 列ではなく C1 列で分割されているからです。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、データベース・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。

複製 MQT 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・グループの単一パーティション表でも複数パーティション表でも構いません。ほとんどの場合、複製される表は小さく、単一パーティション・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適格となる行数を制限することにより、複製されるデータを制限できます。

ソース表のコピーをすべてのデータベース・パーティションに作成するため、複数パーティションのデータベース・パーティション・グループに複製 MQT を作成することもできます。すべてのデータベース・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、複製された表に対してはユニーク索引の作成および制約の定義は行えません (同じ制約がソース表にある場合でも定義できません)。

複製された表は照会で直接参照できますが、複製された表で DBPARTITIONNUM スカラー関数を使用して特定のデータベース・パーティション上の表データを参照することはできません。

複製された MQT 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るには、DB2 Explain 機能を使用します。複製された MQT がオプティマイザーで選択されたアクセス・プランで使用されるかどうかは、結合されるデータによって異なります。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・グループの他のデータベース・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、複製された MQT が使用されない場合があります。

パーティション・データベースでの結合ストラテジー

パーティション・データベース環境での結合ストラテジーは、非パーティション・データベース環境でのストラテジーとは異なる場合があります。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。

頻繁に結合が行われる表では、表コロケーションを考慮する必要があります。パーティション・データベース環境では、表コロケーションとは、互換性のあるパーティション・キーの数が同じである 2 つの表が、同じデータベース・パーティション・グループに保管されている場合に生じる状態のことです。この状態になると、結合処理はそのデータが保管されているデータベース・パーティションで実行できるようになり、結果セットをコーディネーター・データベース・パーティションに移動するだけで済みます。

表キュー

パーティション・データベース環境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 (TQ と呼ばれることもある) は、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間) で行を転送するための機構です。
- 指示表キュー (DTQ と呼ばれることもある) は、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です。
- ブロードキャスト表キュー (BTQ と呼ばれることもある) は、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です。

表キューは、次の方法で表データを渡す場合に使用されます。

- 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・パーティションから別のデータベース・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、データベース・パーティション内で渡す
- 単一パーティション・データベースの使用時に、データベース・パーティション内で渡す

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、アプリケーションからフェッチ要求が出される前に、行の読み取りを行うためです。FETCH ステートメント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 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 ステートメント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 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、順序を保持しません。

- Listener 表キュー (LTQ と呼ばれることもある)

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。

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・データベース環境では、コロケートッド結合、外部表のブロードキャスト結合、外部表の指示結合、内部表および外部表の指示結合、内部表のブロードキャスト結合、内部表の指示結合といった、いくつかの結合方式を使用できます。

以下の図中の q1、q2、および q3 は、表キューを指しています。参照される表は 2 つのデータベース・パーティションに分割されていて、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・データベース・パーティションはデータベース・パーティション 0 です。

コンパイラーによって選択された結合方式がハッシュ結合の場合、リモート・データベース・パーティションごとに作成されるフィルターを使用して、ハッシュ結合が処理されるデータベース・パーティションにタプルが送信される前にタプルを除去することができ、これによってパフォーマンスが向上します。

コロケートッド結合

コロケートッド結合はデータがあるデータベース・パーティションでローカルに発生します。結合の完成後、そのデータベース・パーティションはデータを他のデータベース・パーティションに送信します。 옵ティマイザーがコロケートッド結合を処理するためには、結合される表は連結され、対応する分散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。297 ページの図 27 に例を示します。

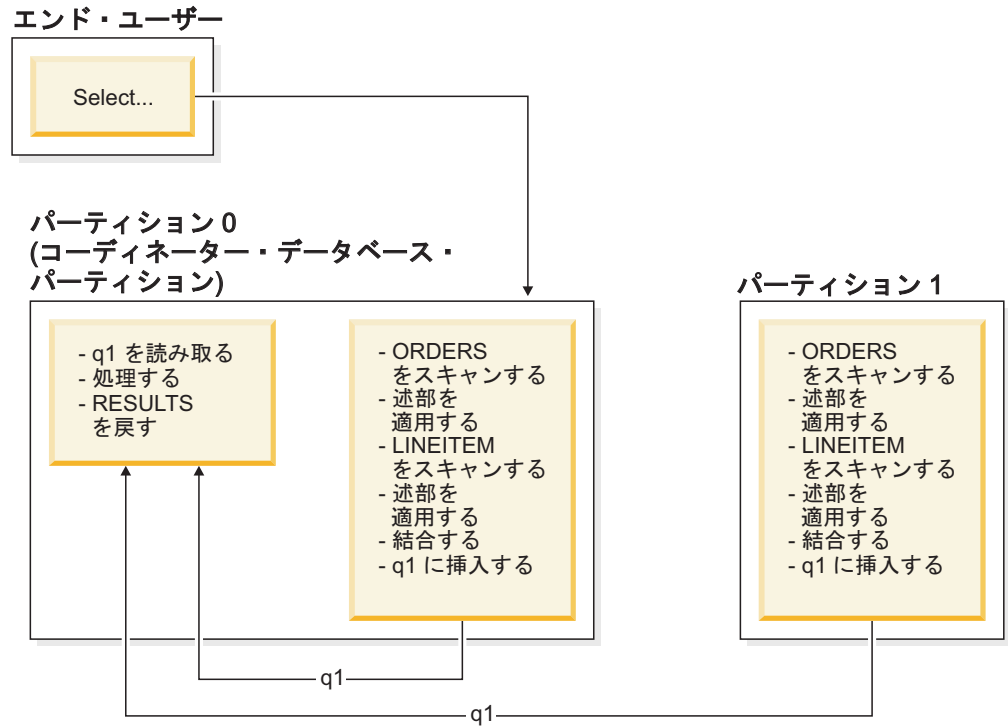


図 27. コロケートッド結合の例

LINEITEM および 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

複製マテリアライズ照会表 (MQT) はコロケートッド結合の可能性を高めます。

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を示します。また、この結合方式は、コスト面での効果においてこれが最良の結合方式となるその他の状況でも使用できます。例えば、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分割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。 298 ページの図 28 に例を示します。

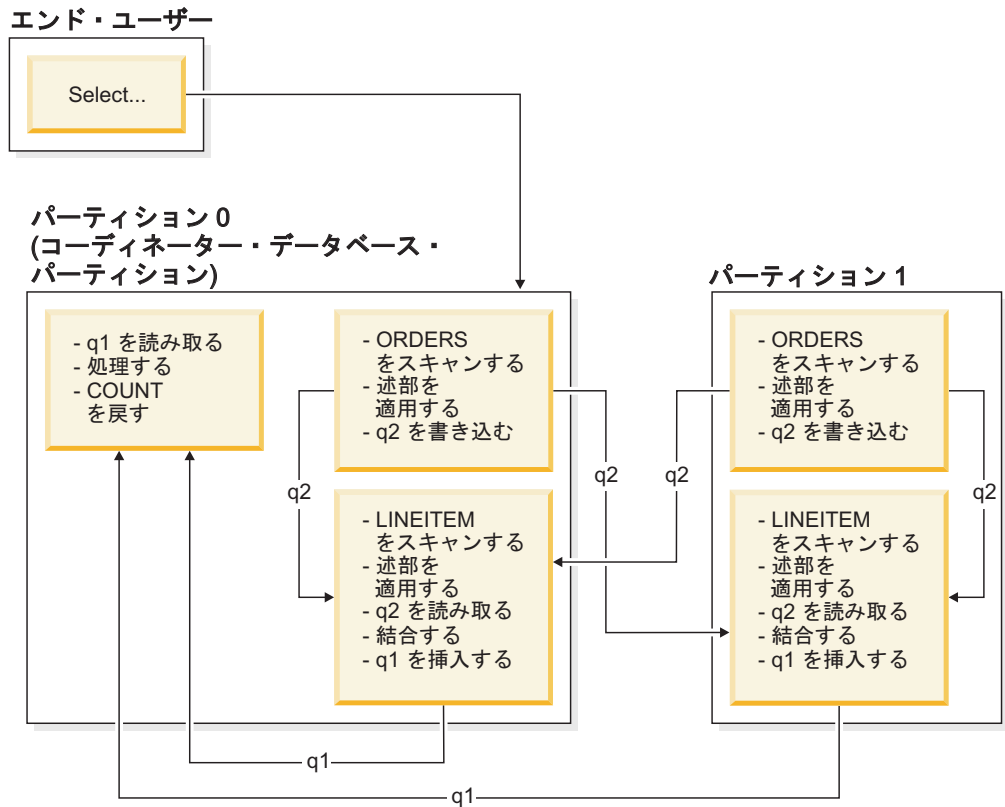
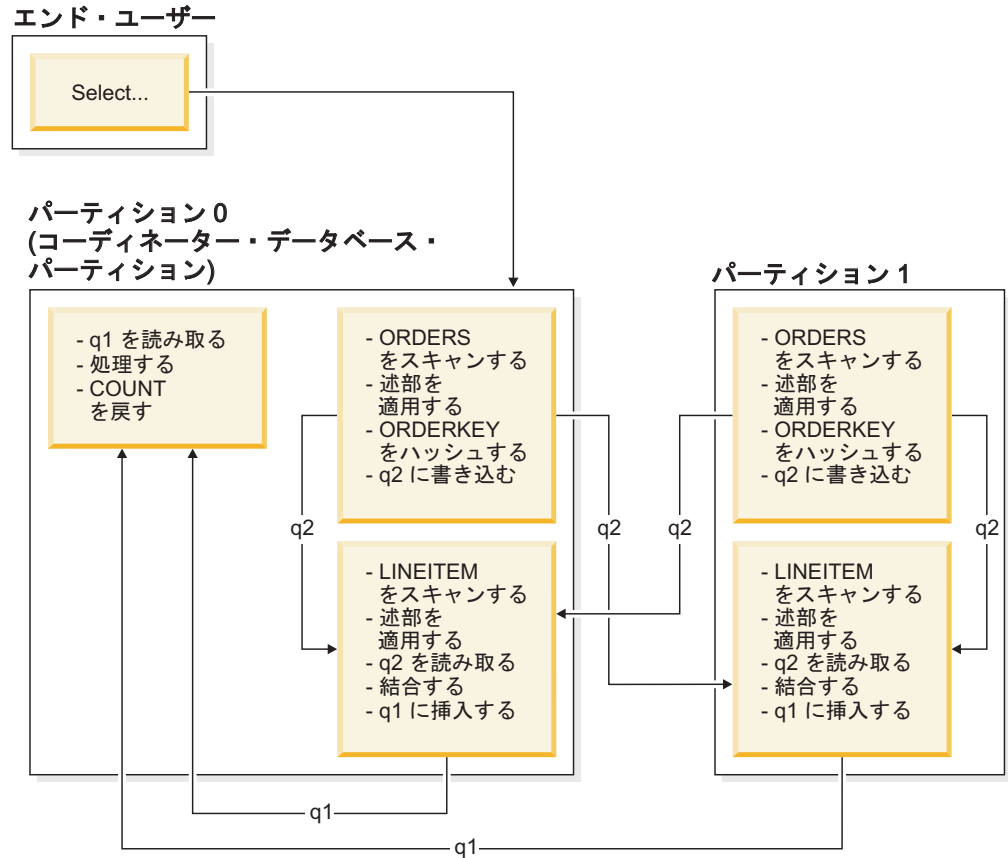


図28. 外部表のブロードキャスト結合の例

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 q2 は、内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。

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分割属性に基づいて内部表の一部に送ります。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。299 ページの図 29 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 29. 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。300 ページの図 30 に例を示します。

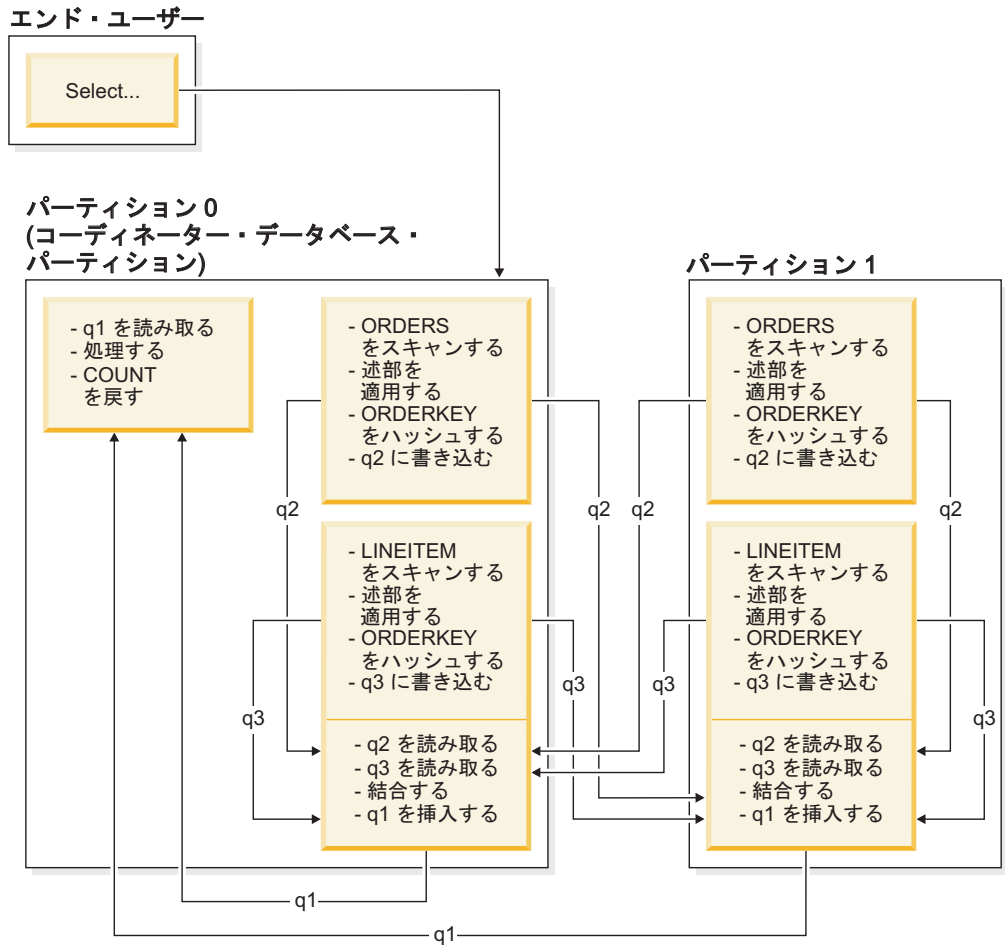
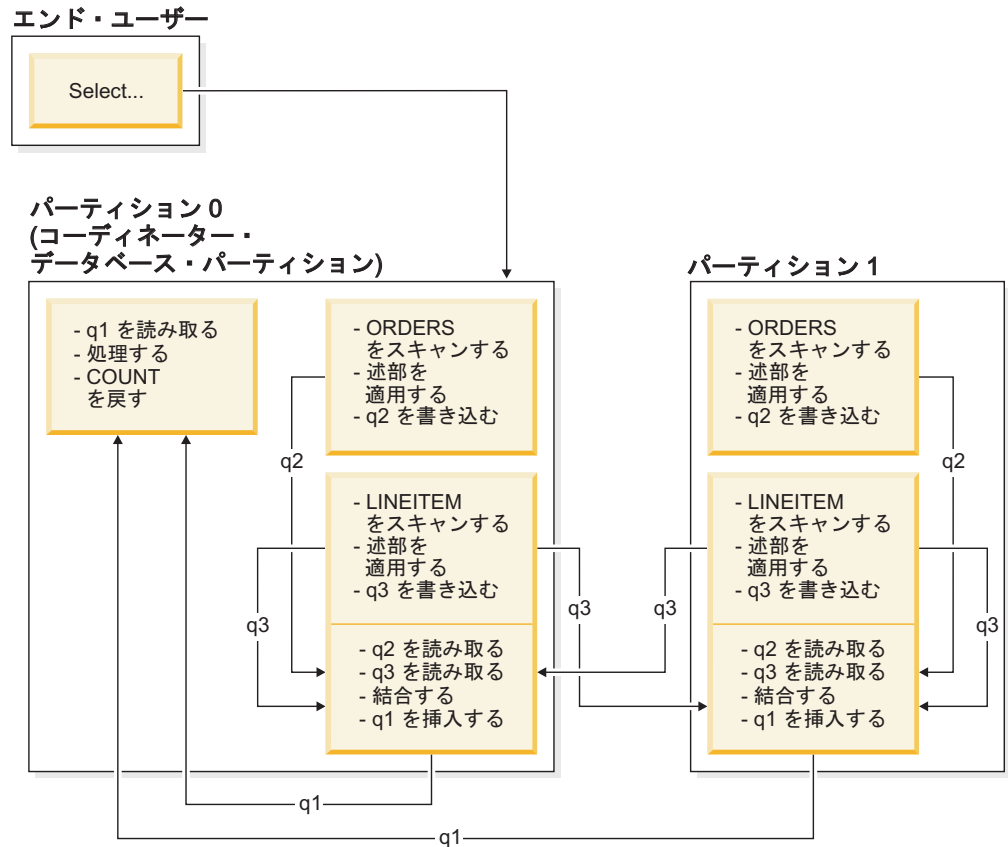


図 30. 内部表および外部表の指示結合の例

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・パーティションに送られて、そこで結合される。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

内部表のブロードキャスト結合

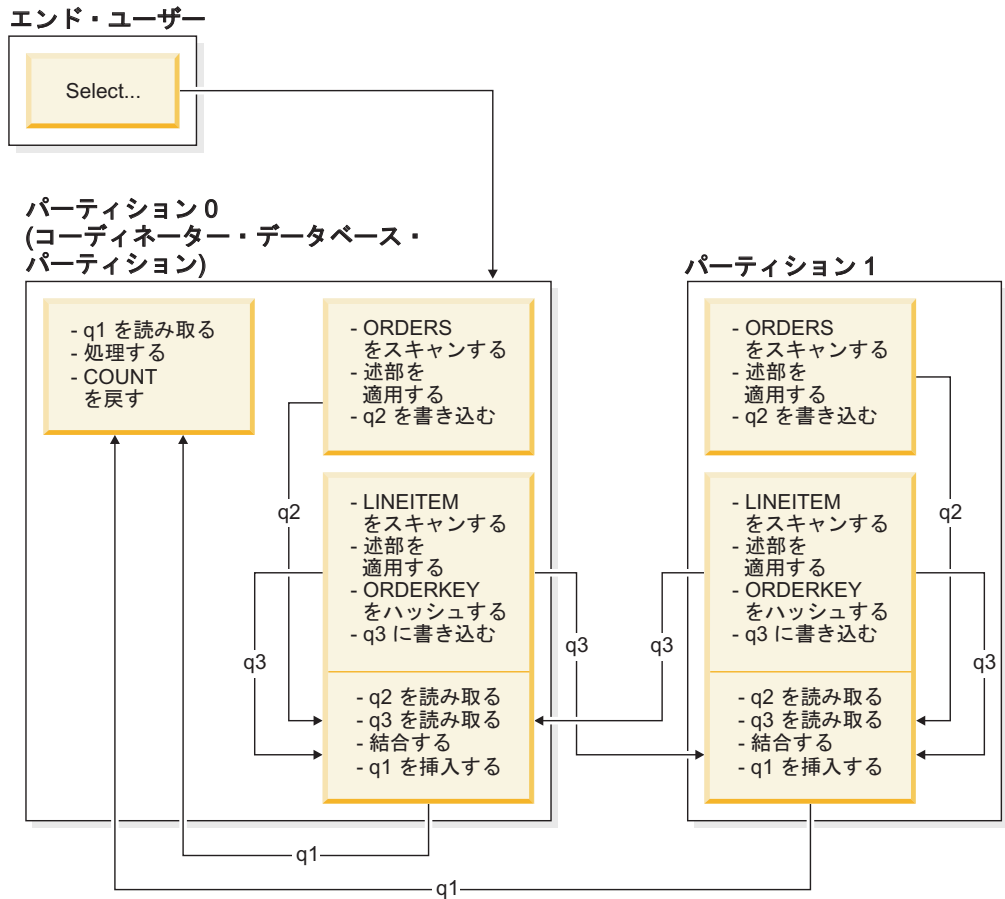
内部表のブロードキャスト結合方式では、内部表が外部表のすべてのデータベース・パーティションに対してブロードキャストされます。301 ページの図 31 に例を示します。



LINEITEM 表は、ORDERS 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 31. 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、外部表の分割属性に基づいて外部表のデータベース・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。302 ページの図 32 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 32. 内部表の指示結合の例

照会最適化における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ソート済みデータを処理します。

- 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・マネージャーを使用して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・パスが実行されます。それぞれのパスの実行中に行セット全体のうちの 1 つのサブセットがソートされます。各パスはバッファ・プール内の一時表に記憶されます。バッファ・プールに十分な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。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 (RDS) に渡されます。(RDS は、データベースの内容にアクセスまたは操作するための要求を処理する DB2 コンポーネントです。)

グループ化およびソート・プッシュダウン演算子

場合によっては、オプティマイザーは、RDS のデータ管理サービス (DMS) 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。(DMS は、データベース内の表および表データの作成、除去、保守、およびアクセスを制御する DB2 コンポーネントです。) これらの操作をプッシュダウンにすると、DMS 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、DMS 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。例えば、次の照会にはこのタイプ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、集約の以下の 3 つのステージのうち、1 つ以上のステージが必要です。

- 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。部分集約の際、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書き出します。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番目の集約の段階である**中間集約**は、書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分散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、複数のデータベース・パーティションにわたってグループを分けている分散キーのサブセットであって、集約を完了するために再分散が必要な場合に、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、各エージェントが書き出されたソート実行のマーヅを終了するときに、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。
- 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** 演算子で常に生じます。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。集約のこの方式は、通常、既に正しい順序になっているデータをグループ化するために使用されます。

最適化ストラテジー

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・データベース環境では、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、それらは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、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、複数の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

ランと同様に、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。

並列表スキャンは範囲パーティション表に対して実行できます。同様に、並列索引スキャンは、パーティション索引に対して実行できます。並列索引スキャンの場合には、パーティション索引は、索引キー値およびキー値あたりのキー項目数に基づいて、レコード範囲に分割されます。並列スキャンが開始すると、サブエージェントにレコード範囲が割り当てられ、サブエージェントは 1 つ範囲を完了するたびに新しい範囲が割り当てられます。索引パーティションは任意の時点で、予約されていない索引パーティションをスキャンしている可能性があるサブエージェントを使用して順次スキャンされます。これらのサブエージェントは、互いが完了するのを待ちません。データ・パーティションの除去分析に基づく照会に関連した索引パーティションのサブセットのみがスキャンされます。

オブティマイザーは、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。オブティマイザーは、サブエージェント間で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

オブティマイザーは、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、共有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。例えば、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、対応する表の部分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分割されるかに応じて、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分割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・エージェントにより集約が完全に行われます。
- 部分的に集約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、グループ化列上で分割されるため、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、部分的に集約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。例えば、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。

データのフィルタリングとデータの偏りによって、照会の実行中にサブエージェント間のワークロードに不均衡が生じる可能性があります。ワークロードの不均衡による効率の悪さは、結合その他、コンピューターに負荷を掛ける操作によって拡大します。オプティマイザーは照会のアクセス・プランの中で不均衡の原因を探し、バランシング方針を適用して、作業がサブエージェント間で均等に分割されるようにします。順序付けられていない外部データ・ストリームの場合、オプティマイザーはこの外部データ・ストリームに REBAL 演算子を使用して、結合のバランスをとります。順序付けられているデータ・ストリームの場合 (順序付けられているデータは索引のアクセスやソートにより生成されます)、オプティマイザーは共有ソートを使用してデータのバランスをとります。ソートが一時表にオーバーフローした場合、ソート・オーバーフローによる負担が大きいため、共有ソートは使用されません。

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、複数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。

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間並列処理によるパフォーマンス上の利点も活用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。
- ブロック索引はレコード ID (RID) 索引よりも小さいので、参照はより高速になります。
- 索引の ANDing 操作および ORing 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。
- ロールアウトを使用できるときに行の削除が高速になります。

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、こうしたブロックのみをスキャンして、結果セットを迅速にフェッチできます。

ロールアウト削除

ロールアウトを使用する削除が許可される条件になると、MDC 表から行を削除するための、さらに効果的なこの方法が使用されます。必要な条件は以下のとおりです。

- DELETE ステートメントは検索条件付き DELETE であって、位置指定 DELETE ではない (このステートメントは WHERE CURRENT OF 節を使用しない)。
- WHERE 節がない (すべての行が削除される) または WHERE 節の条件だけがディメンションに適用される。
- 表が DATA CAPTURE CHANGES 節で定義されていない。
- 表が参照整合性リレーションシップで親でない。
- 表に ON DELETE トリガーが定義されていない。
- 表が即時に更新される MQT で使用されない。
- カスケード削除操作がロールアウトに適切になる可能性がある (その外部キーがその表のディメンション列のサブセットである場合)。
- DELETE ステートメントは、(CREATE TRIGGER ステートメント上の OLD TABLE AS 節により指定される) トリガー SQL 操作の前に、影響を受ける一連の行を識別する一時表に対して実行される SELECT ステートメントには入れることができません。

ロールアウト削除の際、削除レコードはログに記録されません。その代わりに、レコードの入ったページはページのパーツを再フォーマットすることによって空にされます。再フォーマットされたパーツに対する変更はログに記録されますが、レコード自体はログに記録されません。

デフォルトの動作である即時クリーンアップ・ロールアウト は、削除時に RID 索引をクリーンアップするためのものです。このモードは、DB2_MDC_ROLLOUT レジストリー変数を IMMEDIATE に設定するか、または IMMEDIATE を SET CURRENT MDC ROLLOUT MODE ステートメントで指定することによって、指定することもできます。標準の削除操作と比較して、索引の更新のロギングに変更がない場合、パフォーマンスの向上は RID 索引の数によって異なります。RID 索引が少ないと、合計時間およびログ・スペースの割合が改善されます。

節約されるログ・スペースの見積もりは、以下の公式によって作成できます。

$$S + 38*N - 50*P$$

ここで、 N は削除されるレコードの数であり、 S は削除されるレコードの合計サイズ (NULL 標識および VARCHAR の長さなどのオーバーヘッドを含む) であり、 P は削除されるレコードの入ったブロックのページ数です。この数値は、実際のログ・データを縮約したものです。アクティブ・ログ・スペースで必要な節約量はその値の倍です。ロールバック用に予約されたスペースの節約量があるためです。

代替方法として、据え置きクリーンアップ・ロールアウト を使用して、トランザクション・コミット後に RID 索引を更新できます。このモードは、DB2_MDC_ROLLOUT レジストリー変数を DEFER に設定するか、または DEFERRED を SET CURRENT MDC ROLLOUT MODE ステートメントで指定することでも指定できます。据え置きロールアウトで、RID 索引は削除コミット後に、バックグラウンドで非同期にクリーンアップされます。このロールアウトのメソッドは、非常に大規模な削除の場合、または表に多数の RID 索引が存在する場合は結果としてかなり高速な削除となります。即時索引クリーンアップでは索引内の各行は 1 つずつクリーンアップされるのに対し、据え置き索引クリーンアップ中に索引は並列でクリーンアップされるので、クリーンアップ操作全体の速度は向上します。さらに、非同期索引クリーン

アップでは、索引キーではなく索引ページにより索引更新をログ記録するので、DELETE ステートメントのトランザクション・ログスペース所要量は大幅に削減されます。

注: 据え置きクリーンアップ・ロールアウトには、データベース・ヒープから取られる追加のメモリー・リソースが必要です。データベース・マネージャーが必要とするメモリー構造を割り振れない場合、据え置きクリーンアップ・ロールアウトは失敗し、メッセージが管理通知ログに書き込まれます。

据え置きクリーンアップ・ロールアウトを使用する状況

削除のパフォーマンスが最も重要な要因であり、RID 索引が表に定義されている場合は、据え置きクリーンアップ・ロールアウトを使用してください。索引クリーンアップの前に、ロールアウトされたブロックの索引ベースのスキューは、ロールアウトされたデータの量に応じてパフォーマンスがやや低下します。即時索引クリーンアップと据え置き索引クリーンアップとを決定するときは、以下の問題についても考慮してください。

- 削除操作のサイズ

非常に大規模な削除に対しては、据え置きクリーンアップ・ロールアウトを選択します。ディメンション DELETE ステートメントが数多くの小さな MDC 表に対して発行される場合は、非同期に索引オブジェクトをクリーンアップするためのオーバーヘッドが、削除操作中の時間の節約という利点を上回ってしまう可能性があります。

- 索引の数およびタイプ

表に行レベルの処理を必要とする数多くの RID 索引が含まれている場合は、据え置きクリーンアップ・ロールアウトを使用します。

- ブロック可用性

DELETE ステートメントのコミット直後に、その削除操作により解放されるブロック・スペースを使用できるようにするには、即時クリーンアップ・ロールアウトを使用します。

- ログ・スペース

ログ・スペースが制限されている場合、大規模削除の据え置きクリーンアップ・ロールアウトを使用します。

- メモリー制約

据え置きクリーンアップ・ロールアウトは、据え置きクリーンアップが保留中のすべての表の追加のデータベース・ヒープ・スペースを消費します。

削除でロールアウト動作を無効にするには、**DB2_MDC_ROLLOUT** レジストリー変数を OFF に設定するか、または NONE を SET CURRENT MDC ROLLOUT MODE ステートメントに指定します。

注: DB2 バージョン 9.7 以降のリリースでは、パーティション化された RID 索引を含むデータ・パーティション MDC 表において据え置きクリーンアップ・ロールアウトはサポートされません。サポートされるのは、NONE モードと IMMEDIATE モードだけです。 **DB2_MDC_ROLLOUT** レジストリー変数が DEFER に設定されている

場合、または **DB2_MDC_ROLLOUT** の設定をオーバーライドするために **CURRENT MDC ROLLOUT MODE** 特殊レジスターが **DEFERRED** に設定されている場合、クリーンアップ・ロールアウト・タイプは **IMMEDIATE** になります。

MDC 表に対して非パーティション化された RID 索引のみ存在する場合には、据え置き索引クリーンアップ・ロールアウトがサポートされます。

パーティション表の最適化ストラテジー

データ・パーティションの除去とは、照会述部に基づき、照会に応答するためにアクセスする必要があるのは表のデータ・パーティションのサブセットのみであると判断するデータベース・サーバーの機能のことです。データ・パーティションの除去は、パーティション表に対して意思決定支援照会を実行する際に特に役立ちます。

パーティション表は、データ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに表データを分割するというデータ編成スキームを使用します。分割は、表の 1 つ以上の表パーティション・キー列の値に従って行われます。表のデータは、**CREATE TABLE** ステートメントの **PARTITION BY** 節で提供された仕様に基づいて、複数のストレージ・オブジェクトにパーティション化されます。このストレージ・オブジェクトは異なる表スペース、同じ表スペース内、またはその両方に配置することができます。

以下の例は、データ・パーティションの除去のパフォーマンス上の利点について示しています。

```
create table custlist(
  subdate date, province char(2), accountid int)
partition by range(subdate) (
  starting from '1/1/1990' in ts1,
  starting from '1/1/1991' in ts1,
  starting from '1/1/1992' in ts1,
  starting from '1/1/1993' in ts2,
  starting from '1/1/1994' in ts2,
  starting from '1/1/1995' in ts2,
  starting from '1/1/1996' in ts3,
  starting from '1/1/1997' in ts3,
  starting from '1/1/1998' in ts3,
  starting from '1/1/1999' in ts4,
  starting from '1/1/2000' in ts4,
  starting from '1/1/2001'
  ending '12/31/2001' in ts4)
```

2000 年の顧客情報にのみ関心があるとします。

```
select * from custlist
where subdate between '1/1/2000' and '12/31/2000'
```

311 ページの図 33 に示されているように、データベース・サーバーはこの照会を解決するために、表スペース **TS4** の 1 つのデータ・パーティションのみにアクセスする必要があると判断します。

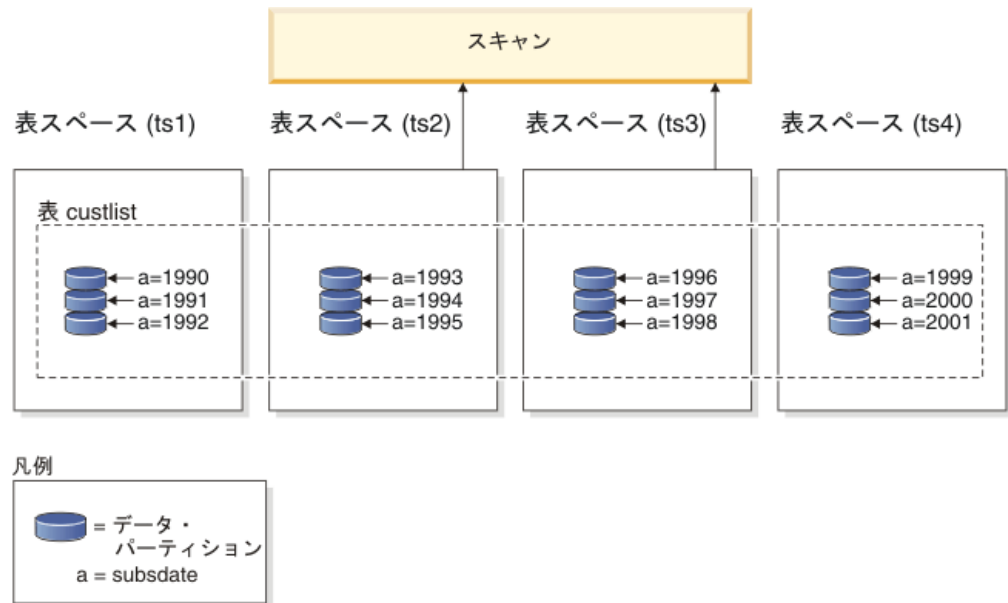


図 33. データ・パーティションの除去のパフォーマンス上の利点

データ・パーティションの除去の別の例は、以下のスキームに基づいています。

```
create table multi (
  sale_date date, region char(2))
partition by (sale_date) (
  starting '01/01/2005'
  ending '12/31/2005'
  every 1 month)

create index sx on multi(sale_date)

create index rx on multi(region)
```

以下の照会を発行するとします。

```
select * from multi
  where sale_date between '6/1/2005'
         and '7/31/2005' and region = 'NW'
```

表のパーティション化を使用しないで考えられる 1 つのプランは、索引 ANDing です。索引 ANDing は、以下のタスクを実行します。

- 各索引から関連するすべての索引項目を読み取る
- 行 ID (RID) の両方のセットを保管する
- RID を突き合わせて、両方の索引のどちらで生じたかを判別する
- RID を使用して行を取り出す

312 ページの図 34 で示されているように、表のパーティション化を使用すると、REGION と SALE_DATE の両方での一致を検出するために索引が読み取られ、それにより一致する行を素早く取得することができます。

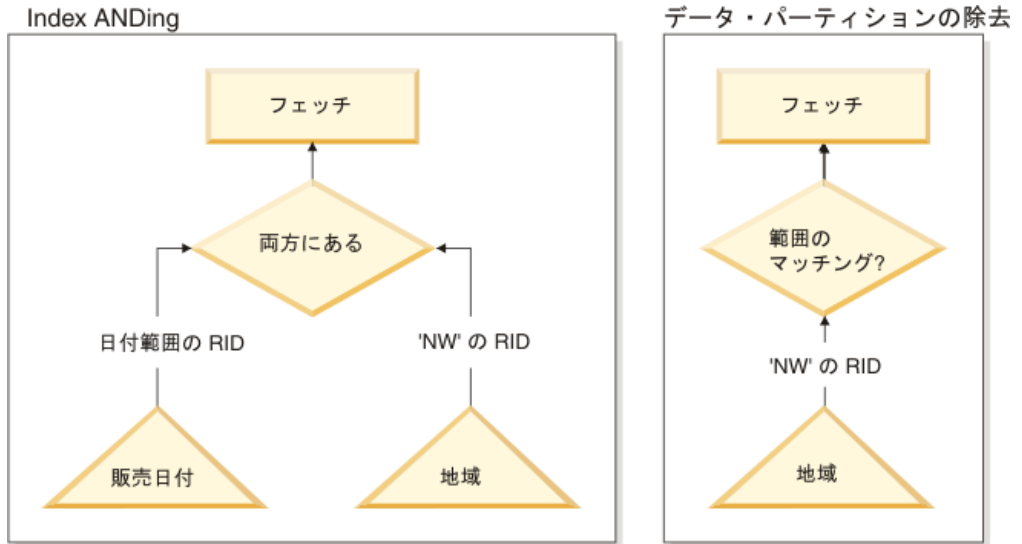


図 34. 表のパーティション化と索引 ANDing の両方のオプティマイザーの決定パス

DB2 Explain

さらに、Explain 機能を使用して、照会オプティマイザーによって選択されたデータ・パーティションの除去プランを判別できます。『DP Elim Predicates』情報には、以下の照会を解決するためにどのデータ・パーティションがスキャンされるかが示されます。

```
select * from custlist
  where subdate between '12/31/1999' and '1/1/2001'
```

Arguments:

```
-----
DPESTFLG: (Number of data partitions accessed are Estimated)
  FALSE
DPLSTPRT: (List of data partitions accessed)
  9-11
DPNUMPRT: (Number of data partitions accessed)
  3
```

DP Elim Predicates:

```
-----
Range 1)
  Stop Predicate: (Q1.A <= '01/01/2001')
  Start Predicate: ('12/31/1999' <= Q1.A)
```

Objects Used in Access Plan:

```
-----
Schema: MRSRINI
Name:    CUSTLIST
Type:    Data Partitioned Table
Time of creation:    2005-11-30-14.21.33.857039
Last statistics update:    2005-11-30-14.21.34.339392
Number of columns:    3
Number of rows:    100000
Width of rows:    19
Number of buffer pool pages:    1200
Number of data partitions:    12
Distinct row values:    No
Tablespace name:    <VARIOUS>
```

複数列のサポート

データ・パーティションの除去は、表パーティション・キーとして複数列が使用されている場合に向いています。例えば、

```
create table sales (  
  year int, month int)  
  partition by range(year, month) (  
    starting from (2001,1)  
    ending at (2001,3) in ts1,  
    ending at (2001,6) in ts2,  
    ending at (2001,9) in ts3,  
    ending at (2001,12) in ts4,  
    ending at (2002,3) in ts5,  
    ending at (2002,6) in ts6,  
    ending at (2002,9) in ts7,  
    ending at (2002,12) in ts8)  
  
select * from sales where year = 2001 and month < 8
```

照会オプティマイザーは、この照会を解決するために TS1、TS2 および TS3 内のデータ・パーティションのみにアクセスする必要があると推論します。

注: 表パーティション・キーが複数列から構成されている場合、複合キーの先頭列に述部がある場合に限ってデータ・パーティションの除去が可能です。表パーティション・キーとして使用される非先頭列は独立していないからです。

複数範囲のサポート

複数範囲 (OR で結ばれたもの) を持つデータ・パーティションでデータ・パーティションの除去を行うことが可能です。前述の例で作成された SALES 表を使用して、以下の照会を実行します。

```
select * from sales  
  where (year = 2001 and month <= 3)  
         or (year = 2002 and month >= 10)
```

データベース・サーバーは、2001 年の最初の四半期と 2002 年の最後の四半期のデータのみにアクセスします。

生成列

生成列を表パーティション・キーとして使用できます。例えば、

```
create table sales (  
  a int, b int generated always as (a / 5))  
  in ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10  
  partition by range(b) (  
    starting from (0)  
    ending at (1000) every (50))
```

この場合、生成列の述部がデータ・パーティションの除去に使用されます。加えて、列を生成するために使用される式が単調の場合、データベース・サーバーはソース列上の述部を生成列上の述部に変換し、生成列でデータ・パーティションの除去を可能にします。例えば、

```
select * from sales where a > 35
```

データベース・サーバーは、a (a > 35) から b (b > 7) に追加の述部を生成し、データ・パーティションの除去が可能になります。

結合述部

結合述部が表アクセス・レベルに下がると、結合述部をデータ・パーティションの除去に使用することもできます。結合述部は、ネストされたループ結合 (NLJN) の内部結合の表アクセス・レベルにのみ下がります。

次の表を考慮してください。

```
create table t1 (a int, b int)
  partition by range(a,b) (
    starting from (1,1)
    ending (1,10) in ts1,
    ending (1,20) in ts2,
    ending (2,10) in ts3,
    ending (2,20) in ts4,
    ending (3,10) in ts5,
    ending (3,20) in ts6,
    ending (4,10) in ts7,
    ending (4,20) in ts8)

create table t2 (a int, b int)
```

次の 2 つの述部を使用できます。

```
P1: T1.A = T2.A
P2: T1.B > 15
```

この例では、結合の外部値が不明なため、コンパイル時にアクセスされるデータ・パーティションを正確に判別することはできません。この場合、ホスト変数またはパラメーター・マーカが使用される場合と同様に、必要な値が結合される実行時にデータ・パーティションの除去が生じます。

実行時に T1 が NLJN の内部にある場合、述部に基づいて、T2.A のすべての外部値に対してデータ・パーティションの除去が動的に生じます。実行時に、述部 T1.A = 3 および T1.B > 15 が外部値 T2.A = 3 に対して適用され、表スペース TS6 のデータ・パーティションにアクセスする資格を与えます。

表 T1 と T2 の列 A に以下の値があるとします。

外部表 T2: 列 A	内部表 T1: 列 A	内部表 T1: 列 B	内部表 T1: データ・パーティション・ロケーション
2	3	20	TS6
3	2	10	TS3
3	2	18	TS4
	3	15	TS6
	1	40	TS3

(内部表に対する表スキャンを前提として) ネスト・ループ結合を実行するために、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. T2 から最初の行を読み取ります。A の値は 2 です。
2. T2.A の値 (2) を結合述部 T1.A = T2.A で列 T2.A にバインドします。述部は T1.A = 2 となります。

3. 述部 T1.A = 2 および T1.B > 15 を使用して、データ・パーティションの除去を適用します。これにより、表スペース TS4 のデータ・パーティションが資格を得ます。
4. T1.A = 2 および T1.B > 15 の適用後に行が検出されるまで、表 T1 の表スペース TS4 のデータ・パーティションをスキャンします。資格にかなう最初の検出行は、T1 の行 3 です。
5. 一致した行を結合します。
6. 次の一致 (T1.A = 2 および T1.B > 15) が検出されるまで、表 T1 の表スペース TS4 のデータ・パーティションをスキャンします。それ以上、行は見つかりません。
7. T2 のすべての行が処理されるまで、T2 の次の行 (A の値を 3 に置換する) に対してステップ 1 から 6 までを繰り返します。

XML データの索引

DB2 バージョン 9.7 フィックスパック 1 以降では、パーティション表の XML データに対する索引を、パーティション索引と非パーティション索引のどちらとしても作成できます。デフォルトはパーティション索引です。

パーティションおよび非パーティション XML 索引は、表の挿入、更新、および削除の各操作時にデータベース・マネージャーによって保守され、その方法はパーティション表の他のリレーショナル索引の保守方法と同じです。パーティション表における XML データの非パーティション索引は、非パーティション表の XML データの索引と同様の仕方で照会処理の速度を速めるために使用されます。照会述部を使用すると、照会に応答するために、パーティション表のデータ・パーティションのサブセットのみにアクセスする必要があることを判別できます。

XML 列のデータ・パーティションの除去および索引は、連携して照会パフォーマンスを向上させることができます。以下のパーティション表について考慮してください。

```
create table employee (a int, b xml, c xml)
  index in tbspx
  partition by (a) (
    starting 0 ending 10,
    ending 20,
    ending 30,
    ending 40)
```

次に以下の照会について考慮します。

```
select * from employee
  where a > 21
  and xmlexist('$doc/Person/Name/First[.="Eric"]'
    passing "EMPLOYEE"."B" as "doc")
```

オプティマイザーは、述部 a > 21 に基づいて最初の 2 つのパーティションを直ちに除去できます。照会プランでオプティマイザーが列 B の XML データの非パーティション索引を選択すると、XML データの索引を使用する索引スキャンでは、オプティマイザーが実行したデータ・パーティションの除去を活用でき、リレーショナル・データ・パーティションの除去述部によって除去されなかったパーティションに属する結果だけを戻すことができます。

マテリアライズ照会表による照会最適化の改善

マテリアライズ照会表 (MQT) は、複雑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。

これは、とりわけ以下の操作が 1 つ以上必要な照会ではそう言えます。

- 1 ディメンション以上の集約データ
- 表のグループを対象とする結合および集約データ
- 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」水平データベース・パーティションまたは垂直データベース・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からの再パーティション・データ、または表の一部

MQT の情報は、SQL および XQuery コンパイラーに組み込まれています。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。この場合、ユーザーには、転送された MQT ではなく、基本表に対するアクセス権が必要です。

MQT は多くの点で REGULAR 表のような働きをするため、表スペース定義および索引を使用したデータ・アクセスの最適化、および runstats ユーティリティを起動してのデータ・アクセスの最適化に関する指針は、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、以下の例では、マルチディメンション分析照会でどのように MQT を活用できるかが示されています。一連の顧客と一連のクレジット・カード口座が入っているデータベース・ウェアハウスについて考えます。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。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示す表) が含まれるため、このスキーマはマルチスター・スキーマ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 (地域または販売区域の場合もある) の情報が含まれ、単一の非正規化された表に保管されます。時刻階層には、日、月、および年情報が含まれ、単一の日付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。

以下の階層の各レベルで、売上に関して MQT が作成されます。

- 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せん。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、および国（地域または販売区域の場合もある）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データ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 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 pname,
         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.pg_salessum;
```

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
- 特定の製品グループまたは製品ラインの売上の合計
- 1995 年と 1996 年における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、地域、または販売区域の売上の合計

これらの照会の正確な答はこの MQT にはありませんが、答の一部が既に計算されているので、MQT を使用して答を計算するコストは、大きな基本表を使用する場合のコストよりはるかに少なくなる場合があります。MQT を使用すると、コストのかかる基本データの結合、ソート、および集約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。

最初の照会は、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 番目の照会は、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;

```

基本表が大きくなれば大きくなるほど、MQT を使用した場合の応答時間向上の可能性は大きなものになります。MQT を使用すると、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。計算は MQT が作成されるたびに 1 回と、リフレッシュされるたびに 1 回行われるだけですが、その内容は多くの照会が実行されているときに再利用できます。

Explain 機能

DB2 の Explain 機能により、SQL または XQuery ステートメントでオプティマイザ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。

提供される情報は、そのアクセス・プランを選ぶのに使われた決定基準を記述しています。この情報は、パフォーマンスを向上させるためにステートメントまたはインスタンス構成をチューニングする上で役立ちます。さらに具体的に説明すると、Explain 情報は以下のタスクを行う上で役立ちます。

- 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。ステートメントを変更したり構成変更を行ったりした後は、新しい Explain 情報を調べて、行ったアクションによるパフォーマンスへの影響を調べます。

キャプチャーされる情報には、以下の情報が含まれます。

- 照会を処理するために使用された操作の順序
- 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 情報がキャプチャーされた時点の、SQL または XQuery ステートメントで参照された全オブジェクトに関する統計
- SQL または XQuery ステートメントを再最適化するために使用された、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 機能は EXPLAIN ステートメントを発行することにより呼び出されます。これにより、特定の Explain 可能ステートメントのために選択されたアクセス・プランに関する情報がキャプチャーされ、この情報が Explain 表に書き込まれます。EXPLAIN ステートメントを発行する前に Explain 表を作成しておく必要があります。

す。さらに、Explain 機能の動作を制御する特殊レジスターである CURRENT EXPLAIN MODE または CURRENT EXPLAIN SNAPSHOT も設定できます。

Explain ユーティリティーを使用するために必要な特権と権限については、EXPLAIN ステートメントの説明を参照してください。Explain 情報にアクセスする必要はあるが、データベースに格納されているデータにはアクセスする必要がないユーザーには、EXPLAIN 権限を付与できます。この権限はデータベース管理者権限のサブセットで、表に格納されたデータにアクセスする固有の特権はありません。

Explain 情報を表示するために、コマンド行ツールを使用できます。使用するツールによって、Explain 機能の動作を制御する特殊レジスターをどのように設定するかが決まります。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、あるいはカスタム SQL または XQuery ステートメントを使用して詳細な分析を実行したい場合には、すべての Explain 情報をキャプチャーします。

IBM Data Studio バージョン 3.1 以降、SQL ステートメントまたは XPATH ステートメントの現行アクセス・プランのダイアグラムを生成することができます。詳細については、Visual Explain を使ったアクセス・プランのダイアグラミングを参照してください。

Explain 機能を使用して SQL ステートメントをチューニングする

Explain 機能は、SQL ステートメントを実行するために照会オプティマイザーによって選択された、照会アクセス・プランを表示するのに使用します。

それには、SQL ステートメントの実行に使用する関係演算についての広範な詳細が含まれており、その中にはプラン演算子、それらの引数、実行の順番、およびコストなどがあります。照会アクセス・プランは照会パフォーマンスにおける最も重要な要素の 1 つであるため、照会パフォーマンス上の問題を診断する際、Explain 機能出力について理解しておくことは重要です。

通常、以下の目的で Explain 情報を使用します。

- 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。

パフォーマンスの変化の分析

照会パフォーマンスが変化した理由を理解するには、以下のステップを実行して、「前と後」の Explain 情報を取得します。

1. 変更前の照会の Explain 情報をキャプチャーし、結果の Explain 表を保存します。あるいは、**db2exfmt** ユーティリティーからの出力を保存することもできます。しかし、Explain 情報を Explain 表に入れておくなら、それらを SQL で照会するのが容易になり、より洗練された分析を行う上でも役立ちます。その上、リレーショナル DBMS にデータを入れておくことで得られる、メンテナンス上の明白なメリットがすべてもたらされます。**db2exfmt** ツールは、いつでも実行することができます。
2. 現行のカタログ統計を保存または印刷します。**db2look** コマンドを使用して、このタスクの実行に役立てることもできます。DB2 バージョン 9.7 では、Explain 表にデータを設定する際に Explain スナップショットを収集できます。

Explain スナップショットには、ステートメントの Explain 時における関係する統計のすべてが含まれます。 **db2exfmt** ユーティリティーは、スナップショットに含まれる統計を自動的にフォーマットします。これは特に、自動またはリアルタイムの統計収集を使用している場合に重要です。なぜなら、照会の最適化で使用された統計は、システム・カタログ表にはまだ存在していない可能性があるためです。あるいはそれらが、ステートメントの Explain 時とシステム・カタログからの統計取得時の間に変更されている可能性があるためです。

3. データ定義言語 (DDL) ステートメント (CREATE TABLE、CREATE VIEW、CREATE INDEX、および CREATE TABLESPACE のステートメントを含む) を保存または印刷します。 **db2look** コマンドもこのタスクを実行します。

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行うときに、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。この情報を収集するのが特に重要なのは、システムの大きな変更 (新しいサービス・レベルまたは DB2 リリースのインストールなど)、または構成の重大な変更 (データベース・パーティションの追加や除去、およびデータの再配分など) の前です。なぜなら、この種のシステム変更を行うと、結果としてアクセス・プランに不都合な変化が生じるおそれがあるからです。アクセス・プランの退行はめったに起きないはずですが、この情報を使用可能にしておけば、パフォーマンスの退行をより迅速に解決する助けになるでしょう。パフォーマンスの変化を分析するために、前に収集した情報を、分析の開始時に収集する照会および環境に関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・プラン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。 **db2exfmt** によって表示されるカタログ統計情報を使用すると、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が付きます。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る可能性があります。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試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・チューニング努力の評価

構成パラメーターの調整、コンテナの追加、または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た後、Explain 機能を使用すると、変更によって選択したアクセス・プランに影響が及んでいる場合に、それを判別することができます。例えば、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。

Explain 表および Explain 情報の編成

`Explain` インスタンスは 1 つまたは複数の SQL または XQuery 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る `Explain` 情報には、コンパイル環境に関する情報、ならびにコンパイル対象となっている SQL ステートメントまたは XQuery ステートメントを満たすために選ばれるアクセス・プランに関する情報が含まれます。

例えば、`Explain` インスタンスは、以下のいずれかの項目で構成される場合があります。

- 静的照会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 または XQuery ステートメント。SQL ステートメント (XML データを照会するステートメントを含む) の場合、`CALL`、コンパウンド SQL (動的)、`DELETE`、`INSERT`、`MERGE`、`REFRESH TABLE`、`SELECT`、`SELECT INTO`、`SET INTEGRITY`、`UPDATE`、`VALUES`、および `VALUES INTO` の各ステートメントに関する `Explain` 情報をキャプチャーできます。XQuery ステートメントの場合、`XQUERY db2-fn:xmlcolumn` および `XQUERY db2-fn:sqlquery` ステートメントに関する `Explain` 情報を取得することができます。

注: `REFRESH TABLE` および `SET INTEGRITY` ステートメントは、動的にのみコンパイルされます。

- 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` ステートメント (動的または静的)。

`EXPLAIN` ステートメントを発行するかセクション `Explain` インターフェースを使用することにより呼び出すことができる `Explain` 機能は、特定の `Explain` 可能ステートメントのために選択されるアクセス・プランに関する情報をキャプチャーし、その情報を `Explain` 表に書き込みます。 `EXPLAIN` ステートメントを発行する前に `Explain` 表を作成しておく必要があります。この表を作成するには、以下のいずれかの方法を使用します。

- `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` スクリプトを実行します。
- `SYSPROC.SYSINSTALLOBJECTS` プロシージャを使用します。このプロシージャを使用して、`Explain` 表のドロップおよび妥当性検査を行うこともできます。

この表は、特定のスキーマおよび表スペースの下に作成することができます。サンプルは、`EXPLAIN.DDL` ファイル内にあります。

`Explain` 表は、複数のユーザーに共通にすることができます。あるユーザー用の表を定義してから、その定義した表を指す別名を各追加ユーザー用に作成できます。別の方法として、`SYSTOOLS` スキーマの下に `Explain` 表を定義することもできます。ユーザーのセッション ID (動的 SQL ステートメントまたは動的 XQuery ステート

メントの場合)、またはステートメントの許可 ID (静的 SQL ステートメントまたは静的 XQuery ステートメントの場合) の下に Explain 表も別名もない場合、 Explain 機能は SYSTOOLS スキーマをデフォルトとして使用します。共通の Explain 表を共有する各ユーザーは、それらの表に対する INSERT 特権を保持している必要があります。

次の表に、各 Explain 表の目的を要約します。

表 60. Explain 表のサマリー

表名	説明
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、照会コンパイラまたは db2adv コマンドを使用して設定することもできますし、ユーザーが自分で設定することもできます。この表は、推奨索引の取得および提案された索引の評価を行うために使用されます。
ADVISE_INSTANCE	db2adv コマンド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。この表には、 db2adv コマンドの実行ごとに 1 行が入ります。
ADVISE_MQT	以下の情報が格納されます。 <ul style="list-style-type: none"> • 推奨される各マテリアライズ照会表 (MQT) を定義する照会 • 各 MQT の列統計。COLSTATS (XML 形式) や NUMROWS など • 各 MQT の詳細な統計を取得するためのサンプリング照会
ADVISE_PARTITION	db2adv コマンドによって生成および評価される仮想データベース・パーティションを保管します。
ADVISE_TABLE	MQT、マルチディメンション・クラスタリング表 (MDC)、およびデータベース・パーティションに関する設計アドバイザーの最終的な推奨値を使った、表作成のためのデータ定義言語 (DDL) ステートメントを保管します。
ADVISE_WORKLOAD	ワークロード内の SQL ステートメントまたは XQuery ステートメントごとに 1 つの行が入ります。 db2adv コマンドは、この表を使ってワークロード情報を収集し、保管します。
EXPLAIN_ACTUALS	Explain セクション実行時統計情報を含みます。
EXPLAIN_ARGUMENT	各演算子に固有の特性に関する情報がある場合、それを格納します。
EXPLAIN_DIAGNOSTIC	EXPLAIN_STATEMENT 表で EXPLAIN されたステートメントの特定のインスタンスに対して生成された各診断メッセージの項目を含みます。
EXPLAIN_DIAGNOSTIC_DATA	EXPLAIN_DIAGNOSTIC 表で記録された診断メッセージのメッセージ・トークンを含みます。メッセージ・トークンは、メッセージを生成した SQL ステートメントの実行に固有の追加情報を提供します。

表 60. Explain 表のサマリー (続き)

表名	説明
EXPLAIN_INSTANCE	すべての Explain 情報用の主コントロール表です。Explain 表の各行は、この表内の固有の行にリンクされます。Explain 対象の SQL ステートメントまたは XQuery 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が、この表に保持されます。
EXPLAIN_OBJECT	SQL または XQuery 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・オブジェクトを示します。
EXPLAIN_OPERATOR	照会コンパイラーが SQL または XQuery ステートメントを満たすために必要とするすべての演算子が入っています。
EXPLAIN_PREDICATE	特定の演算子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 情報に関して発行された SQL ステートメントまたは XQuery ステートメントのテキストが格納されます。発行した SQL ステートメントまたは XQuery ステートメントと、オプティマイザーがアクセス・プランを選択するとき使用するバージョンが、この表に格納されます。 Explain スナップショットが要求されると、照会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、EXPLAIN_STATEMENT 表の SNAPSHOT 列に保管されます。
EXPLAIN_STREAM	個々の演算子とデータ・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。演算子は、EXPLAIN_OPERATOR 表に示されます。データ・オブジェクトは、EXPLAIN_OBJECT 表に示されています。
OBJECT_METRICS	特定の時刻におけるセクションの特定の実行で参照されるオブジェクトごとの実行時統計を含みます。複数のメンバーについてのオブジェクト統計を収集する場合、この表にはオブジェクトが参照されたメンバーごとに 1 行が含まれます。パーティション化されたオブジェクトについてのオブジェクト統計を収集する場合、この表にはデータ・パーティションごとに 1 行が含まれます。 この表に情報が格納されるのは、アクティビティ・イベント・モニターによってセクション実行時統計がキャプチャーされる場合のみです。

データ・オブジェクトの Explain 情報:

単一のアクセス・プランで、1 つの SQL または XQuery ステートメントを満たすために 1 つ以上のデータ・オブジェクトが使用される場合があります。

オブジェクト統計

Explain 機能は、各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ソートされているかどうか (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・プール内で占有するページ数
- オブジェクトが保管されている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、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 4 KB 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 (4 KB ページ単位)
- 索引内のデータ・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、およびツリー内のレベル数
- 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

データ演算子の Explain 情報:

SQL または XQuery ステートメントを満たし、結果をユーザーに戻すために、単一のアクセス・プランによってデータに対していくつかの操作が実行される場合があります。照会コンパイラーは、表スキャン、索引スキャン、ネスト・ループ結合、またはグループ化演算子など、必要な操作を判別します。

Explain 出力は、アクセス・プランで使用される各演算子に関する情報を示すだけでなく、アクセス・プランの累積効果も示します。

見積コスト情報

演算子に関する以下に示した累積コストの見積もりが記録されます。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た演算子に至るまでのコストが示されます。

- 合計コスト (timeron 単位)
- ページ入出力の数
- 処理命令の数
- 最初の行を取り出すためのコスト (timeron 単位)。必要な初期オーバーヘッドを含む
- コミュニケーション・コスト (フレーム単位)

timeron は、架空の相対単位です。timeron 値は、内部値、例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、オブティマイザーによって判別されます。

そのため、`timeron` 単位で見積コストが判別される時、SQL または XQuery ステートメントの `timeron` 値が毎回同じになるという保証はありません。

複数のネットワーク・アダプターが関係している場合は、最も高い値を持つアダプターの累積コミュニケーション・コストが返されます。これらのコミュニケーション・コストの値には、物理マシン間のネットワーク・トラフィックのコストしか含まれていません。これらの値には、パーティション・データベース環境における同一物理マシン上のノード・パーティション間でのフレームの受け渡し（つまり、ネットワークには流れないフレームの受け渡し）のコストは含まれていません。

演算子の特性

各演算子の特性を示す以下の情報が `Explain` 機能によって記録されます。

- アクセスされた表集合
- アクセスされた列セット
- データが配列されている列（オプティマイザーが、この配列を後続の演算子で使できると判別した場合）
- 適用された述部セット
- 返される行数の見積もり（カーディナリティー）

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。インスタンス内の各照会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、特定の `Explain` インスタンスを識別し、特定のステートメントに関する情報をこの `Explain` 機能の特定の呼び出しに関連付ける上で役立ちます。

- `Explain` 情報を要求したユーザー
- `Explain` 要求が開始された時刻
- `Explain` されたステートメントが入っているパッケージの名前
- `Explain` されたステートメントが入っているパッケージの SQL 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

照会コンパイラーが照会を最適化したデータベース・マネージャー環境に関する情報がキャプチャーされます。環境情報には、以下のものが含まれます。

- DB2 製品のバージョンおよびリリース番号
- 照会のコンパイルに使用された並列処理の度合い

`CURRENT DEGREE` 特殊レジスター、`DEGREE BIND` オプション、`SET RUNTIME DEGREE` コマンド、および `dft_degree` データベース構成パラメーターで、特定の照会のコンパイル時に使用される並列処理の度合いが決まります。

- 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出現する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の値が記録されます。
 - 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**)

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 インスタンスを一意的に識別する情報に加えて、以下の情報は個々の照会ステートメントを識別するのに役立ちます。

- ステートメントのタイプ。SELECT、DELETE、INSERT、UPDATE、位置指定 DELETE、位置指定 UPDATE、または SET INTEGRITY
- SYSCAT.STATEMENTS カタログ・ビューに記録された、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。 EXPLAIN MODE または EXPLAIN SNAPSHOT がアクティブになっていて、コマンド行プロセッサ (CLP) またはコール・レベル・インターフェース (CLI) セッション中に動的 Explain ステートメントがサブミットされた場合、QUERYTAG 値はそれぞれ「CLP」または「CLI」に設定されます。この場合、デフォルトで QUERYNO 値は、各ステートメントごとに 1 以上ずつ増分される数になります。その他の動的 Explain ステートメントのうち、CLP からでも CLI からでもないもの、または EXPLAIN ステートメントを使用しないもの場合はすべて、QUERYTAG 値は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 されたステートメントごとに、選択されたアクセス・プランを実行するために要する相対コストの見積もりがオプティマイザーにより記録されます。このコストは、*timeron* という考案された相対計測単位で示されます。経過時間の見積もりは、次の理由で提供されません。

- 照会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・プランの効率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 されたステートメントごとに、ステートメント・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、照会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。この変換は他の照会ステートメントに似ているように見えますが、必ずしも正しい照会言語構文に従っているわけでも、内部表記の実際の内容全体を反映しているわけでもありません。この変換は、単に、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コンテキストをユーザーが理解できるようにするために提供されています。照会をさらに最適化するためにコンパイラーでどのように書き直されたかを理解するには、ユーザー作成のステートメント・テキストを照会ステートメントの内部表記と比較します。書き直されたステートメントには、トリガーや制約などのステートメントに影響を及ぼす他の要素も示されます。この「最適化された」テキストで使用されるキーワードには、以下のようなものがあります。

\$Cn 派生列の名前。n は整数値を表します。

\$CONSTRAINTS

このタグは、コンパイル中に元のステートメントに追加された制約を示します。**\$WITH_CONTEXTS** 接頭部と組み合わせて表示されます。

\$DERIVED.Tn

派生表の名前。n は整数値を示します。

\$INTERNAL_FUNC

このタグは、**Explain** が実行された照会に対してコンパイラーが使用する機能があるものの、それは汎用には使用できないことを示します。

\$INTERNAL_PRED

このタグは、**Explain** が実行された照会のコンパイル中に追加された述部があるものの、それは汎用には使用できないことを示します。内部述部は、トリガーまたは制約のために元のステートメントに追加された付加的なコンテキストを満たすために、コンパイラーが使用します。

\$INTERNAL_XPATH

このタグは、単一のアノテーション付き XPath パターンを入力パラメーターとして取り、かつそのパターンに一致する 1 つ以上の列のある表を返すような内部表関数が存在することを示します。

\$RID このタグは、特定の行の行 ID (RID) 列を示します。

\$TRIGGERS

このタグは、コンパイル中に元のステートメントに追加されたトリガーを示します。**\$WITH_CONTEXTS** 接頭部と組み合わせて表示されます。

\$WITH_CONTEXTS(...)

元の照会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先頭に表示されます。この接頭部の後に、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。

Explain 情報のキャプチャーのガイドライン

SQL または XQuery ステートメントのコンパイル時に Explain データを要求するとキャプチャーできます。

追加バインド SQL または XQuery ステートメントが実行時にコンパイルされる場合、データはバインド時ではなく実行時に 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ユーザーのものではありません。

Explain 情報がキャプチャーされるのは、SQL または XQuery 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、動的照会ステートメントは、環境の変化に伴って再コンパイルが必要な場合、あるいは Explain 機能がアクティブな場合、再コンパイルされます。同じ PREPARE ステートメントを同じ照会ステートメントに対して発行すると、この照会を準備または実行するたびに、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。

BIND オプション **REOPT ONCE** または **REOPT 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、グローバル変数、または特殊レジスターを含む SQL または XQuery ステートメントはコンパイルされます。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、コンパイル時にこうした値が分からない場合にはデフォルトの見積値を使用してアクセス・パスが作成されます。

REOPT ONCE オプションを使用すると、指定された SQL または XQuery ステートメントと、パッケージ・キャッシュ内の同じステートメントとの突き合わせが試行されます。既に再最適化されてキャッシュに入れられた照会ステートメントの値が、指定された照会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場合、Explain 表には、新たに再最適化されたアクセス・プランと、再最適化に使用された値が入ります。

マルチ・パーティション・データベース・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場所と同じデータベース・パーティション上で行う必要があります。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 および XQuery ステートメント

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** ステートメントを含めます。

- 動的 SQL および XQuery 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た場合

- YES: SQL および XQuery コンパイラーは、Explain データをキャプチャーし、照会ステートメントを実行します。
- EXPLAIN: SQL および XQuery コンパイラーは Explain データをキャプチャーしますが、照会ステートメントは実行しません。
- RECOMMEND INDEXES: SQL および XQuery 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、照会ステートメントは実行されません。
- EVALUATE INDEXES: SQL および XQuery 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。この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、照会コンパイラーは次に、それら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。
- REOPT: 照会コンパイラーは、実行時のステートメント再最適化中にホスト変数、パラメーター・マーカー、グローバル変数、または特殊レジスターの実際の値が使用できる場合、静的または動的 SQL または XQuery ステートメントのために Explain データをキャプチャーします。
- **BIND** または **PREP** コマンドで **EXPLAIN ALL** オプションが指定されている場合には、CURRENT EXPLAIN MODE 特殊レジスターが NO に設定されていても、照会コンパイラーは実行時に動的 SQL および XQuery ステートメントの Explain データをキャプチャーします。

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、EXPLAIN_STATEMENT 表の SNAPSHOT 列に保管されます。

SQL または XQuery ステートメントがコンパイルされ、Explain データが要求されると、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 および XQuery ステートメント

EXPLSNAP ALL か **EXPLSNAP YES** 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** 節を使用する静的 EXPLAIN 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 および XQuery 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** 節を使用する EXPLAIN ステートメントを発行します。FOR SNAPSHOT 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 節では、すべての Explain 情報がキャプチャーされます。
- **CURRENT EXPLAIN SNAPSHOT** 特殊レジスターの場合、以下のように設定されます。

- YES: SQL および XQuery コンパイラーは、`EXPLAIN` スナップショット・データをキャプチャーし、照会ステートメントを実行します。
- EXPLAIN: SQL および XQuery コンパイラーは `EXPLAIN` スナップショット・データをキャプチャーしますが、照会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` 特殊レジスターが `NO` に設定されていても、照会コンパイラーは実行時に `EXPLAIN` スナップショット・データをキャプチャーします。

セクション Explain 情報のキャプチャーのガイドライン

セクション Explain 機能は、ランタイム・セクションの内容のみを使用して、ステートメントに関する Explain 情報を (直接またはツール経由で) キャプチャーします。セクション Explain は `db2expln` コマンドによって提供される機能と似ていますが、セクション Explain には、Explain 機能によって提供される詳細なレベルのアプローチが備えられています。

ランタイム・セクションの内容を使用してステートメントを Explain することで、実際に実行される内容 (実行後にセクションがキャプチャーされた場合は実際に実行された内容) に関する情報および診断を取得できます。これは、異なるアクセス・プランを生成する可能性のある `EXPLAIN` ステートメントとは対照的です (例えば、動的 SQL の場合、統計は最後にステートメントが実行されたときから更新されている可能性があり、その場合には、Explain されるステートメントを `EXPLAIN` ステートメントがコンパイルする際に別のアクセス・プランが選択されることとなります)。

セクション Explain インターフェースは `EXPLAIN` ステートメントによって生成されるものと似た情報を Explain 表に取り込みます。しかし、違いがいくつかあります。データが Explain 表に書き込まれた後、使用する既存の Explain ツールのいずれか (例えば `db2exfmt` コマンド) によってそのデータを処理することができます。

セクション Explain インターフェース

セクション Explain を実行できるインターフェース・プロシージャは、以下のリストに挙げるとおり、4 つあります。プロシージャの違いは、提供される入力 (つまり、セクションを配置するための手段) だけです。

`EXPLAIN_FROM_ACTIVITY`

入力として、アプリケーション ID、アクティビティ ID、UOW ID、およびアクティビティ・イベント・モニター名を取ります。プロシージャは、アクティビティ・イベント・モニターでこのアクティビティに対応するセクションを検索します (SQL アクティビティはセクションの特定の実行です)。このインターフェースを使用するセクション Explain には、セクション実行時統計が含まれています。セクションの特定の実行が行われるからです。

`EXPLAIN_FROM_CATALOG`

入力として、パッケージ名、パッケージ・スキーマ、ユニーク ID、およびセクション数を取ります。プロシージャは、カタログ表で特定のセクションを検索します。

EXPLAIN_FROM_DATA

入力として、実行可能 ID、セクション、およびステートメント・テキストを取ります。

EXPLAIN_FROM_SECTION

入力として、実行可能 ID および場所を取ります。場所は、以下のいずれかを使って指定します。

- メモリー内のパッケージ・キャッシュ
- パッケージ・キャッシュ・イベント・モニター名

プロシージャは、特定の場所でセクションを検索します。

実行可能 ID は一意的に、また一貫してセクションを識別します。実行可能 ID は、実行された各セクションに対してデータ・サーバーで生成される不透明なバイナリー・トークンです。実行可能 ID は、セクションのモニター・データを照会するため、またセクション Explain を実行するための入力として使用されます。

どちらの場合も、プロシージャは、識別されるランタイム・セクションに含まれている情報を使って Explain を実行し、*explain_schema* 入力パラメーターによって識別される Explain 表に Explain 情報を書き込みます。プロシージャを呼び出した後コミットを実行するのは、呼び出し元の責任です。

セクションから得られる Explain 情報を使った照会パフォーマンスの調査:

Explain 機能を使用して、ステートメント自体のセクションからアクセス・プランを生成することにより、特定のステートメントが実際に実行されるときの (または実際に実行されたときの) そのアクセス・プランを調べることができます。それに対し、EXPLAIN ステートメントを使用する場合は、ステートメントを再コンパイルすることによってアクセス・プランが作成されます。これら 2 つの各アクセス・プランの作成方法によって生成されるアクセス・プランは、それぞれ異なる場合があります。例えば、セクションのステートメントが 2 時間前にコンパイルされた場合、そのステートメントが使用するアクセス・プランは、ステートメントに対して EXPLAIN ステートメントを実行することによって生成されるアクセス・プランと異なる可能性があります。

アクティビティ・イベント・モニターの情報が得られる場合、EXPLAIN_FROM_ACTIVITY プロシージャを使って実行した後、セクションのアクセス・プランを生成できます。(セクション実行時統計を収集している場合、アクセス・プランの Explain 機能によって生成される見積もりとこの情報を一緒に表示することもできます。詳しくは、338 ページの『セクション実行時統計のキャプチャーおよびそのセクションへのアクセス』を参照してください。)

ステートメントのアクティビティ・イベント・モニター情報が得られない場合、EXPLAIN_FROM_SECTION プロシージャを使用して、パッケージ・キャッシュに保管されているセクションに基づいてステートメントが実行されるときのそのアクセス・プランを生成できます。このトピックでは、EXPLAIN_FROM_SECTION を使用してパッケージ・キャッシュのセクション情報に基づいてステートメントのアクセス・プラン情報を表示する方法を示します。


```

Value of output parameters
-----
Parameter Name : EXPLAIN_SCHEMA
Parameter Value : DB2DOCS

Parameter Name : EXPLAIN_REQUESTER
Parameter Value : DB2DOCS

Parameter Name : EXPLAIN_TIME
Parameter Value : 2010-11-08-13.57.52.984001

Parameter Name : SOURCE_NAME
Parameter Value : SQLC2H21

Parameter Name : SOURCE_SCHEMA
Parameter Value : NULLID

Parameter Name : SOURCE_VERSION
Parameter Value :

```

3. これで、SQL を使って Explain 表を調べるか、より見やすくするために **db2exfmt** コマンドを使って情報をフォーマットすることにより、Explain 情報を調べるができるようになりました。例えば、上記ステップから収集された Explain 情報に対して **db2exfmt -d gsdb -e db2docs -w 2010-11-08-13.57.52.984001 -n SQLC2H21 -s NULLID -t -#0** を実行すると、次のような出力が生成されます。

```

Connecting to the Database.
DB2 Universal Database Version 9.7, 5622-044 (c) Copyright IBM Corp. 1991, 2008
Licensed Material - Program Property of IBM
IBM DATABASE 2 Explain Table Format Tool

```

```

***** EXPLAIN INSTANCE *****

```

```

DB2_VERSION:      09.07.2
SOURCE_NAME:      SQLC2H21
SOURCE_SCHEMA:    NULLID
SOURCE_VERSION:
EXPLAIN_TIME:     2010-11-08-13.57.52.984001
EXPLAIN_REQUESTER: DB2DOCS

```

```

Database Context:
-----

```

```

Parallelism:      None
CPU Speed:        8.029852e-007
Comm Speed:       100
Buffer Pool size: 21418
Sort Heap size:   6590
Database Heap size: 1196
Lock List size:   21386
Maximum Lock List: 97
Average Applications: 1
Locks Available:  663821

```

```

Package Context:
-----

```

```

SQL Type:         Dynamic
Optimization Level: 5
Blocking:         Block All Cursors
Isolation Level:  Cursor Stability

```

```

----- STATEMENT 1 SECTION 201 -----

```

```

QUERYNO:          0
QUERYTAG:         CLP
Statement Type:   Select
Updatable:        No
Deletable:        No
Query Degree:     1

```

Original Statement:

```

-----
select cust_last_name, cust_cc_number, cust_interest_code
from gosalesct.cust_crdt_card C, gosalesct.cust_customer D,
    gosalesct.cust_interest E
where C.cust_code=d.cust_code AND c.cust_code=e.cust_code
group by d.cust_last_name, c.cust_cc_number, e.cust_interest_code
order by d.cust_last_name ASC, c.cust_cc_number DESC, e.cust_interest_code
ASC

```

Optimized Statement:

```

-----
SELECT Q5.CUST_LAST_NAME AS "CUST_LAST_NAME", Q5.CUST_CC_NUMBER AS
    "CUST_CC_NUMBER", Q5.CUST_INTEREST_CODE AS "CUST_INTEREST_CODE"
FROM
    (SELECT Q4.CUST_LAST_NAME, Q4.CUST_CC_NUMBER, Q4.CUST_INTEREST_CODE
    FROM
        (SELECT Q2.CUST_LAST_NAME, Q3.CUST_CC_NUMBER, Q1.CUST_INTEREST_CODE
        FROM GOSALESC.T.CUST_INTEREST AS Q1, GOSALESC.T.CUST_CUSTOMER AS Q2,
            GOSALESC.T.CUST_CRDT_CARD AS Q3
        WHERE (Q3.CUST_CODE = Q1.CUST_CODE) AND (Q1.CUST_CODE = Q2.CUST_CODE))
        AS Q4
    GROUP BY Q4.CUST_INTEREST_CODE, Q4.CUST_CC_NUMBER, Q4.CUST_LAST_NAME) AS
    Q5
ORDER BY Q5.CUST_LAST_NAME, Q5.CUST_CC_NUMBER DESC, Q5.CUST_INTEREST_CODE

```

Explain level: Explain from section

Access Plan:

```

-----
Total Cost:      1255.29
Query Degree:    1

      Rows
      RETURN
      ( 1)
      Cost
      I/O
      |
      31255
      GRPBY
      ( 2)
      1255.29
      NA
      |
      31255
      TBSCAN
      ( 3)
      1249.02
      NA
      |
      31255
      SORT
      ( 4)
      1242.74
      NA
      |
      31255
      ^HSJOIN
      ( 5)
      1134.96
      NA
      /-----+-----\
      31255          31255
      HSJOIN        TBSCAN
      ( 6)          ( 9)
      406.871      716.136
      NA          NA
      /-----+-----\
      31255          31255          31255
      TBSCAN        IXSCAN        TABLE: GOSALESC.T
      ( 7)          ( 8)          CUST_CUSTOMER
      235.505      159.488          Q2
      NA          NA
      |          |
      31255      -1
      TABLE: GOSALESC.T  INDEX: SYSIBM

```

CUST_CRDT_CARD SQL101108113609000
Q3 Q1

:

Objects Used in Access Plan:

Schema: SYSIBM
Name: SQL101108113609000
Type: Index
Last statistics update: 2010-11-08-13.29.58.531000
Number of rows: -1
Number of buffer pool pages: -1
Distinct row values: Yes
Tablespace name: GOSALES_TS
Tablespace overhead: 7.500000
Tablespace transfer rate: 0.060000
Prefetch page count: 32
Container extent page count: 32
Index clustering statistic: 1.000000
Index leaf pages: 37
Index tree levels: 2
Index full key cardinality: 31255
Base Table Schema: GOSALESCT
Base Table Name: CUST_INTEREST
Columns in index:
CUST_CODE(A)
CUST_INTEREST_CODE(A)

Schema: GOSALESCT
Name: CUST_CRDT_CARD
Type: Table
Last statistics update: 2010-11-08-11.59.58.531000
Number of rows: 31255
Number of buffer pool pages: 192
Distinct row values: No
Tablespace name: GOSALES_TS
Tablespace overhead: 7.500000
Tablespace transfer rate: 0.060000
Prefetch page count: 32
Container extent page count: 32
Table overflow record count: 0
Table Active Blocks: -1
Average Row Compression Ratio: 0
Percentage Rows Compressed: 0
Average Compressed Row Size: 0

Schema: GOSALESCT
Name: CUST_CUSTOMER
Type: Table
Last statistics update: 2010-11-08-11.59.59.437000
Number of rows: 31255
Number of buffer pool pages: 672
Distinct row values: No
Tablespace name: GOSALES_TS
Tablespace overhead: 7.500000
Tablespace transfer rate: 0.060000
Prefetch page count: 32
Container extent page count: 32
Table overflow record count: 0
Table Active Blocks: -1
Average Row Compression Ratio: 0
Percentage Rows Compressed: 0
Average Compressed Row Size: 0

Base Table For Index Not Already Shown:

Schema: GOSALESCT
Name: CUST_INTEREST
Time of creation: 2010-11-08-11.30.28.203002
Last statistics update: 2010-11-08-13.29.58.531000
Number of rows: 31255
Number of pages: 128
Number of pages with rows: 124
Table overflow record count: 0
Indexspace name: GOSALES_TS
Tablespace name: GOSALES_TS

```
Tablespace overhead:      7.500000
Tablespace transfer rate:  0.060000
Prefetch page count:     -1
Container extent page count: 32

Long tablespace name:     GOSALES_TS
```

(上記の出力では、表示上の都合でいくつかの行が削除されています。)

次のタスク

Explain 出力を分析し、照会を調整できる部分があるかどうか確認します。

セクション Explain の出力と EXPLAIN ステートメントの出力の相違:

セクション Explain を発行した後に得られる結果は、EXPLAIN ステートメントを実行した後に収集される結果と似ています。ここでは、存在する若干の相違について、影響を受ける Explain 表ごと、また **db2exfmt** ユーティリティによって生成される出力に与える影響 (存在する場合) ごとに説明します。

ストアド・プロシージャの出力パラメーター

EXPLAIN_REQUESTER、EXPLAIN_TIME、SOURCE_NAME、SOURCE_SCHEMA、および SOURCE_VERSION は、Explain 表のセクションに関する情報を調べるために使用されるキーを構成します。これらのパラメーターをいずれかの既存の Explain ツール (例えば **db2exfmt**) で使用して、セクションから取得される Explain 情報をフォーマットします。

EXPLAIN_INSTANCE 表

セクション Explain によって生成される行については、以下の列の設定が異なります。

- EXPLAIN_OPTION は値 S に設定されます
- SNAPSHOT_TAKEN は常に N に設定されます
- REMARKS は常に NULL です

EXPLAIN_STATEMENT 表

セクション Explain が Explain 出力を生成した場合、EXPLAIN_LEVEL 列は値 S に設定されます。注意すべき重要な点として、EXPLAIN_LEVEL 列は表の主キーの一部を構成し、他のほとんどの EXPLAIN 表の外部キーの一部を構成します。このため、この EXPLAIN_LEVEL 値はそれらの他の表にも存在します。

通常 EXPLAIN_LEVEL = P によって行と関連付けられる残りの列の値は、EXPLAIN_STATEMENT 表では EXPLAIN_LEVEL = S のときに存在します。ただし SNAPSHOT は例外です。SNAPSHOT は、EXPLAIN_LEVEL が S の際、常に NULL になります。

セクション Explain の生成時に元のステートメントを使用できなかった場合 (例えば、ステートメント・テキストが EXPLAIN_FROM_DATA プロシージャに提供されなかった場合)、EXPLAIN_LEVEL が 0 に設定されていると、STATEMENT_TEXT はストリング UNKNOWN に設定されます。

セクション Explain の **db2exfmt** 出力では、最適化ステートメントの後、次の行が追加で表示されます。

Explain level: Explain from section

EXPLAIN_OPERATOR 表

セクション Explain の発行後、コストを記録するすべての列のうち、TOTAL_COST 列と FIRST_ROW_COST 列だけに値が取り込まれます。コストを記録する他の列の値はすべて、-1 になります。

セクション Explain の **db2exfmt** 出力には、次の相違点があります。

- アクセス・プランのグラフで、入出力コストは NA として示されます
- 各演算子の詳細で示されるコストは、Cumulative Total Cost と Cumulative First Row Cost だけです

EXPLAIN_PREDICATE 表

相違はありません。

EXPLAIN_ARGUMENT 表

セクション Explain が発行される際、少数の引数タイプは EXPLAIN_ARGUMENT 表に書き込まれません。

EXPLAIN_STREAM 表

セクション Explain の発行後、以下の列には値が取り込まれません。

- SINGLE_NODE
- PARTITION_COLUMNS
- SEQUENCE_SIZES

セクション Explain の発行後、次の列の値は常に -1 になります。

- PREDICATE_ID

セクション Explain の発行後、以下の列は、基本表オブジェクトから生じたストリームのみに関する値を持つか、またはデフォルトでそれぞれ値なしと -1 になります。

- COLUMN_NAMES
- COLUMN_COUNT

セクション Explain の **db2exfmt** 出力では、このリストにある列のうち、値のない列または値が -1 の列の情報は、各演算子の Input Streams および Output Streams セクションから除外されます。

EXPLAIN_OBJECT 表

セクション Explain の発行後、STATS_SRC 列は常に空ストリングに設定され、CREATE_TIME 列は NULL に設定されます。

セクション Explain の発行後、以下の列の値は常に -1 になります。

- COLUMN_COUNT
- WIDTH
- FIRSTKEYCARD
- FIRST2KEYCARD
- FIRST3KEYCARD
- FIRST4KEYCARD
- SEQUENTIAL_PAGES
- DENSITY
- AVERAGE_SEQUENCE_GAP
- AVERAGE_SEQUENCE_FETCH_GAP
- AVERAGE_SEQUENCE_PAGES
- AVERAGE_SEQUENCE_FETCH_PAGES
- AVERAGE_RANDOM_PAGES
- AVERAGE_RANDOM_FETCH_PAGES
- NUMRIDS
- NUMRIDS_DELETED
- NUM_EMPTY_LEAFs
- ACTIVE_BLOCKS
- NUM_DATA_PART

セクション Explain の **db2exfmt** 出力では、ここでリストされている列の情報は、出力の下部近くにある表ごとおよび索引ごとの統計情報から除外されます。

セクション Explain は出力にコンパイラ参照オブジェクト (つまり、OBJECT_TYPE が + で始まる行) を組み込みません。これらのオブジェクトは、**db2exfmt** 出力には表示されません。

セクション実行時統計のキャプチャーおよびそのセクションへのアクセス:

セクション実行時統計は、アクセス・プランのセクションの実行の際に収集される実行時統計です。実行時統計を持つセクションをキャプチャーするには、アクティビティ・イベント・モニターを使用します。セクション実行時統計にアクセスするには、EXPLAIN_FROM_ACTIVITY ストアド・プロシージャを使用してセクション Explain を実行します。

セクション実行時統計を表示できるようにするには、セクション実行時統計がキャプチャーされたセクションに対してセクション Explain を実行する必要があります (つまり、セクションとセクション実行時統計の両方が Explain 機能への入力になります)。ここでは、セクション実行時統計の使用可能化、キャプチャー、およびアクセスに関する情報が提供されます。

セクション実行時統計の使用可能化

セクション実行時統計は、使用可能になっている場合のみ、実行時に更新されません。セクション実行時統計を使用可能にするには、データベース全体を対象とする

場合は **section_actuals** データベース構成パラメーターを使用し、特定のアプリケーションを対象とする場合は **WLM_SET_CONN_ENV** プロシージャを使用します。

セクション実行時統計は、使用可能になっている場合のみ、実行時に更新されます。**section_actuals** データベース構成パラメーターを使用して、セクション実行時統計を使用可能にします。セクション実行時統計を使用可能にするには、このパラメーターを **BASE** に設定します (デフォルト値は **NONE**)。例えば、

```
db2 update database configuration using section_actuals base
```

特定のアプリケーションのセクション実行時統計を使用可能にするには、**WLM_SET_CONN_ENV** プロシージャを使用し、**section_actuals** エレメントに **BASE** を指定します。例えば、

```
CALL WLM_SET_CONN_ENV(NULL,  
    '<collectactdata>WITH DETAILS, SECTION</collectactdata>  
    <collectsectionactuals>BASE</collectsectionactuals>  
    ')
```

注:

1. 作業単位の開始時に有効だった **section_actuals** データベース構成パラメーターの設定値が、その作業単位に含まれるすべてのステートメントに適用されません。**section_actuals** データベース構成パラメーターが動的に変更されても、アプリケーションは次の作業単位まで新しい値を認識しません。
2. アプリケーション用に **WLM_SET_CONN_ENV** プロシージャによって指定された **section_actuals** 設定は、直ちに有効になります。セクション実行時統計は、アプリケーションが次に発行するステートメントについて収集されます。
3. 自動統計プロファイル生成が使用可能に設定されている場合は、セクション実行時統計を使用可能にすることはできません (SQLCODE -5153)。

セクション実行時統計のキャプチャー

セクションおよびセクション実行時統計のキャプチャーは、アクティビティ・イベント・モニターを使って行います。アクティビティ情報の収集が使用可能になっている場合、アクティビティ・イベント・モニターは、アクティビティの実行完了時にアクティビティの詳細を書き出します。アクティビティ情報の収集の使用可能化は、ワークロード、サービス・クラス、しきい値、または作業アクションに対して **COLLECT ACTIVITY DATA** 節を使って行います。セクションおよび実行時統計の収集を指定するには (後者が使用可能な場合)、**COLLECT ACTIVITY DATA** 節の **SECTION** オプションを使用します。例えば、以下のステートメントは、**WL1** ワークロードと関連付けられている接続によって発行されるすべての **SQL** ステートメントが、ステートメントの完了時にアクティブなアクティビティ・イベント・モニターによって収集される情報 (セクションおよび実行時統計を含む) を入手することを示します。

```
ALTER WORKLOAD WL1 COLLECT ACTIVITY DATA WITH DETAILS,SECTION
```

パーティション・データベース環境では、実行されるステートメントに **COLLECT ACTIVITY DATA** 節が適用され、**COLLECT ACTIVITY DATA** 節が **SECTION** キーワードと **ON ALL DATABASE PARTITIONS** 節の両方を指定する場合、アクティビティが実行されたすべてのパーティションのセクション実行時統計がアクティビティ・イベント・モニターによってキャプチャーされます。 **ON ALL**

DATABASE PARTITIONS 節が指定されない場合、コーディネーター・パーティションの実行時統計だけがキャプチャーされます。なお、ワークロード、サービス・クラス、しきい値、または作業アクションに対する COLLECT ACTIVITY DATA 節のほかに、第 2 引数に値が「WITH DETAILS, SECTION」の collectactdata タグが含まれる WLM_SET_CONN_ENV プロシージャを使用してアクティビティ収集を使用可能にすることができます (個別アプリケーションが対象)。

制限 セクション実行時統計のキャプチャーには、次の制限があります。

- 現在実行中のアクティビティに関する情報をアクティビティ・イベント・モニターに送信するために
WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアド・プロシージャが使用される場合、セクション実行時統計はキャプチャーされません。WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアド・プロシージャによってアクティビティ・イベント・モニターのレコードが生成される場合、その its partial_record 列の値はすべて 1 になります。
- 反動的しきい値の違反が発生した場合、コーディネーター・パーティションのセクション実行時統計だけがキャプチャーされます。
- セクション Explain を使ってセクション実行時統計にアクセスするには、事前に Explain 表を DB2 パージョン 9.7 フィックスパック 1 以降にマイグレーションしておく必要があります。Explain 表がマイグレーションされていない場合、セクション Explain は機能しますが、セクション実行時統計の情報は Explain 表に取り込まれません。その場合、項目が EXPLAIN_DIAGNOSTIC 表に作成されます。
- アクティビティ・イベント・モニターによってセクション実行時統計のデータをキャプチャーするには、事前に既存の DB2 V9.7 アクティビティ・イベント・モニター表 (特にアクティビティ表) を再作成しておく必要があります。アクティビティ論理グループに SECTION_ACTUALS 列が含まれていない場合、アクティビティ・イベント・モニターによってキャプチャーされたセクションを使ってセクション Explain を実行できる可能性もありますが、Explain 表にはセクション実行時統計のデータが入りません。

セクション実行時統計へのアクセス

セクション実行時統計へのアクセスは、EXPLAIN_FROM_ACTIVITY プロシージャを使って行えます。セクション実行時統計がキャプチャーされたアクティビティに対してセクション Explain を実行すると、EXPLAIN_ACTUALS Explain 表にセクション実行時統計の情報が取り込まれます。

注: セクション実行時統計は、EXPLAIN_FROM_ACTIVITY プロシージャを使ってセクション Explain を実行する場合のみ取得できます。

EXPLAIN_ACTUALS 表は、既存の EXPLAIN_OPERATOR Explain 表の子表です。EXPLAIN_FROM_ACTIVITY が呼び出されると、セクション実行時統計が入手可能であれば、EXPLAIN_ACTUALS 表に実行時統計のデータが取り込まれます。セクション実行時統計が複数のデータベース・パーティションで収集される場合、EXPLAIN_ACTUALS 表には、データベース・パーティションにつき各演算子のための 1 行が用意されます。

照会パフォーマンスの低下を調査するための、実行時統計を持つセクション Explain の取得:

SQL 照会パフォーマンスの低下を解決するには、まずセクション実行時統計情報が含まれるセクション Explain を取得します。その後、セクション実行時統計の値を、オプティマイザーによって生成される見積もりアクセス・プランの値と比較することにより、アクセス・プランの妥当性を評価することができます。このタスクにより、照会パフォーマンスの低下を調査するためのセクション実行時統計の取得プロセスに進むことができます。

始める前に

調査の診断フェーズが完了済みであり、確かに SQL 照会パフォーマンスの低下が生じていること、またどのステートメントがパフォーマンスの低下に関係していると思われるかを確認しておく必要があります。

このタスクについて

このタスクにより、照会パフォーマンスの低下を調査するためのセクション実行時統計の取得プロセスに進むことができます。セクション実行時統計に含まれている情報は、オプティマイザーによって生成される見積もり値と比較するときに、照会パフォーマンスの低下を解決する上で役立つものとなります。

制約事項

『セクション実行時統計のキャプチャーおよびそのセクションへのアクセス』にある制限を参照してください。

手順

myApp.exe アプリケーションによって実行される照会のパフォーマンスの低下を調査するには、以下のステップを実行します。

1. セクション実行時統計を使用可能にします。

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

2. SYSINSTALLOBJECTS プロシージャを使って MYSCHEMA スキーマに EXPLAIN 表を作成します。

```
CALL SYSINSTALLOBJECTS( 'EXPLAIN', 'C', NULL, 'MYSCHEMA' )
```

注: すでに EXPLAIN 表を作成済みの場合、このステップはスキップできません。

3. 以下の 2 つのコマンドを発行して、myApp.exe アプリケーションによってサブミットされるアクティビティを収集するためのワークロード MYCOLLECTWL を作成し、それらのアクティビティのセクション・データの収集を使用可能にします。

```
CREATE WORKLOAD MYCOLLECTWL APPLNAME( 'MYAPP.EXE' )  
COLLECT ACTIVITY DATA WITH DETAILS,SECTION
```

これに続いて、以下を発行します。

```
GRANT USAGE ON WORKLOAD MYCOLLECTWL TO PUBLIC
```

注: 別個ワークロードを使用することを選択すると、アクティビティ・イベント・モニターによってキャプチャーされる情報の量が制限されます。

- 以下のステートメントを発行して、ACTEVMON というアクティビティ・イベント・モニターを作成します。

```
CREATE EVENT MONITOR ACTEVMON FOR ACTIVITIES WRITE TO TABLE
```

- 以下のステートメントを実行して、アクティビティ・イベント・モニター ACTEVMON を活動化します。

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

- myApp.exe アプリケーションを実行します。アプリケーションによって発行されるステートメントはすべて、アクティビティ・イベント・モニターによってキャプチャーされます。

- 以下のステートメントを発行して、アクティビティ・イベント・モニターの表を照会し、対象のステートメントの ID 情報を検索します。

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

以下は、SELECT ステートメントを発行した結果として生成された出力例です。

APPL_ID	UOW_ID	ACTIVITY_ID	STMT_TEXT
*N2.DB2INST1.0B5A12222841	1	1	SELECT * FROM ...

- 以下の CALL ステートメントで示されているように、アクティビティ ID 情報を EXPLAIN_FROM_ACTIVITY プロシージャへの入力として使用し、実行時統計を持つセクション Explain を取得します。

```
CALL EXPLAIN_FROM_ACTIVITY( '*N2.DB2INST1.0B5A12222841', 1, 1, 'ACTEVMON',
'MYSCHEMA', ?, ?, ?, ?, ? )
```

以下は、EXPLAIN_FROM_ACTIVITY 呼び出しの出力例です。

Value of output parameters

```
-----
Parameter Name : EXPLAIN_SCHEMA
Parameter Value : MYSCHEMA
```

```
Parameter Name : EXPLAIN_REQUESTER
Parameter Value : SWALKTY
```

```
Parameter Name : EXPLAIN_TIME
Parameter Value : 2009-08-24-12.33.57.525703
```

```
Parameter Name : SOURCE_NAME
Parameter Value : SQLC2H20
```

```
Parameter Name : SOURCE_SCHEMA
Parameter Value : NULLID
```

```
Parameter Name : SOURCE_VERSION
Parameter Value :
```

```
Return Status = 0
```

9. 以下のように、**db2exfmt** コマンドを使用し、EXPLAIN_FROM_ACTIVITY プロシージャの出力として返された Explain インスタンス・キーを入力として指定することにより、Explain データをフォーマットします。

```
db2exfmt -d test -w 2009-08-24-12.33.57.525703 -n SQLC2H20 -s NULLID -# 0 -t
```

Explain インスタンスの出力は次のようになります。

```
***** EXPLAIN INSTANCE *****

DB2_VERSION:      09.07.1
SOURCE_NAME:      SQLC2H20
SOURCE_SCHEMA:    NULLID
SOURCE_VERSION:
EXPLAIN_TIME:     2009-08-24-12.33.57.525703
EXPLAIN_REQUESTER: SWALKTY

Database Context:
-----
Parallelism:      None
CPU Speed:        4.000000e-05
Comm Speed:       0
Buffer Pool size: 198224
Sort Heap size:   1278
Database Heap size: 2512
Lock List size:   6200
Maximum Lock List: 60
Average Applications: 1
Locks Available:  119040

Package Context:
-----
SQL Type:         Dynamic
Optimization Level: 5
Blocking:         Block All Cursors
Isolation Level:  Cursor Stability

----- STATEMENT 1 SECTION 201 -----
QUERYNO:          0
QUERYTAG:         CLP
Statement Type:   Select
Updatable:        No
Deletable:        No
Query Degree:     1

Original Statement:
-----
select *
from syscat.tables

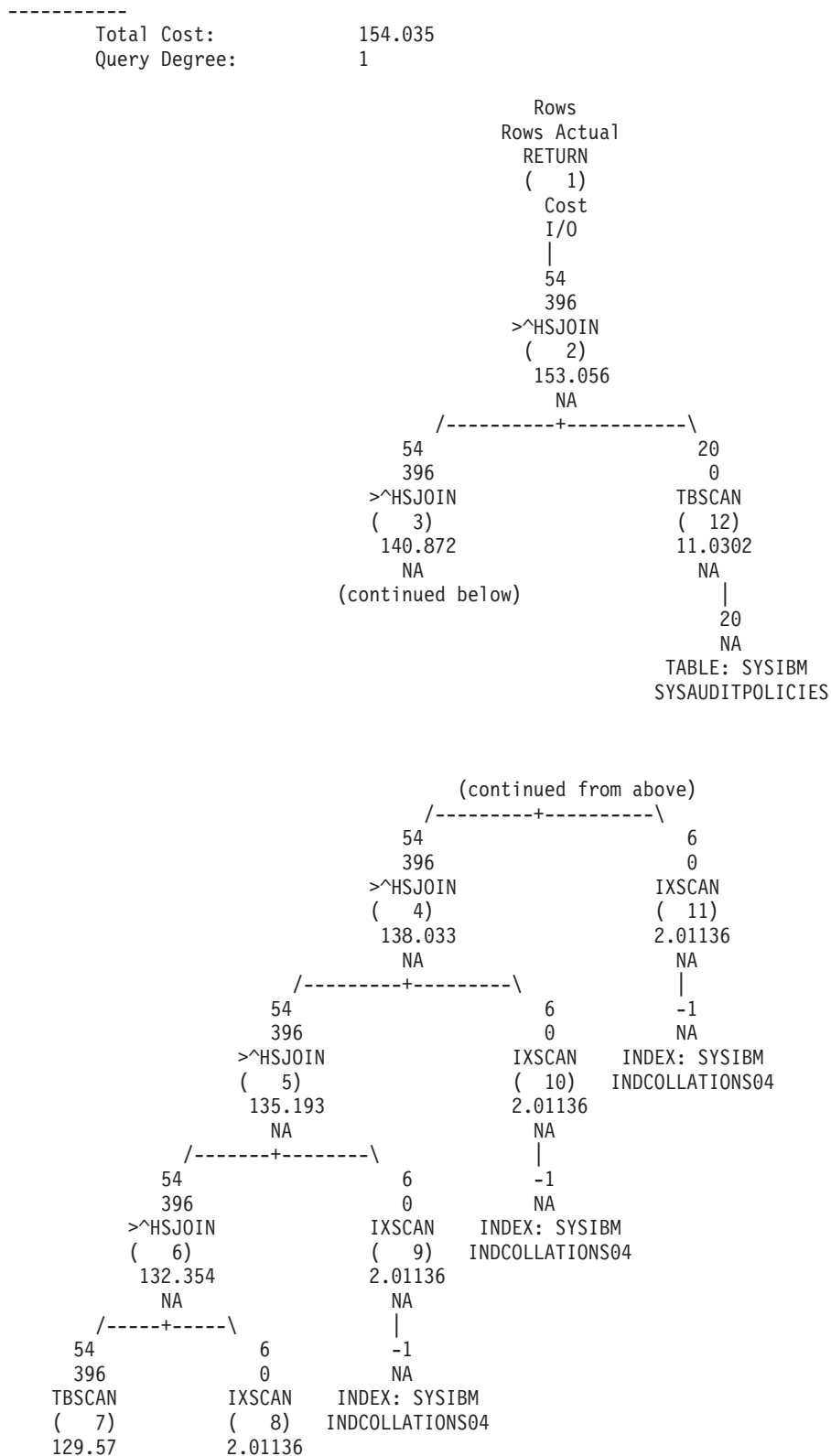
Optimized Statement:
-----
SELECT Q10.$C67 AS "TABSCHEMA", Q10.$C66 AS "TABNAME", Q10.$C65 AS "OWNER",
Q10.$C64 AS "OWNERTYPE", Q10.$C63 AS "TYPE", Q10.$C62 AS "STATUS",
Q10.$C61 AS "BASE_TABSCHEMA", Q10.$C60 AS "BASE_TABNAME", Q10.$C59 AS
"ROWTYPESCHEMA", Q10.$C58 AS "ROWTYPENAME", Q10.$C57 AS "CREATE_TIME",
Q10.$C56 AS "ALTER_TIME", Q10.$C55 AS "INVALIDATE_TIME", Q10.$C54 AS
"STATS_TIME", Q10.$C53 AS "COLCOUNT", Q10.$C52 AS "TABLEID", Q10.$C51
AS "TBSPACEID", Q10.$C50 AS "CARD", Q10.$C49 AS "NPAGES", Q10.$C48 AS
"FPAGES", Q10.$C47 AS "OVERFLOW", Q10.$C46 AS "TBSPACE", Q10.$C45 AS
"INDEX_TBSPACE", Q10.$C44 AS "LONG_TBSPACE", Q10.$C43 AS "PARENTS",
Q10.$C42 AS "CHILDREN", Q10.$C41 AS "SELFREFS", Q10.$C40 AS
"KEYCOLUMNS", Q10.$C39 AS "KEYINDEXID", Q10.$C38 AS "KEYUNIQUE",
Q10.$C37 AS "CHECKCOUNT", Q10.$C36 AS "DATACAPTURE", Q10.$C35 AS
"CONST_CHECKED", Q10.$C34 AS "PMAP_ID", Q10.$C33 AS "PARTITION_MODE",
```

'0' AS "LOG_ATTRIBUTE", Q10.\$C32 AS "PCTFREE", Q10.\$C31 AS
 "APPEND_MODE", Q10.\$C30 AS "REFRESH", Q10.\$C29 AS "REFRESH_TIME",

...

Explain level: Explain from section

Access Plan:



```

      NA          NA
      |          |
      54         -1
      NA          NA
TABLE: SYSIBM   INDEX: SYSIBM
SYSTABLES      INDCOLLATIONS04

```

...

10. Explain 出力のセクション実行時統計情報を調べます。セクション実行時統計の値を、オブティマイザーによって生成されるアクセス・プランの見積もり値と比較します。セクション実行時統計とアクセス・プランの見積もり値の間に相違がある場合は、その相違の原因を突き止め、適切なアクションを取ります。例えば (説明のための例ですが)、照会される表の 1 つに関して、表統計の日付が古いものだったことを突き止めたとします。これにより、オブティマイザーは、照会パフォーマンスの低下の原因となっている可能性のある正しくないアクセス・プランを選択します。この場合に取りべき一連のアクションは、表に対して **RUNSTATS** コマンドを実行して、表統計を更新することです。
11. アプリケーションを再実行して、照会の低下が持続しているかどうか確認します。

Explain 出力のセクション実行時統計情報の分析:

セクション実行時統計が使用可能な場合、これは Explain 出力のさまざまな部分に表示されます。ここでは、セクション実行時統計情報、演算子の詳細、およびオブジェクト統計が Explain 出力内のどこにあるかを説明します。

db2exfmt コマンドのグラフ出力のセクション実行時統計

Explain 実行時統計が使用可能な場合、それがグラフ内の見積もり行数の下に表示されます。グラフ出力には演算子のみに関する実行時統計が含まれており、オブジェクトの実行時統計はありません。オブジェクトについてはグラフで、NA (適用外) が表示されます。

db2exfmt コマンドのグラフ出力の例を以下に示します。

```

      Rows
Rows Actual
      RETURN
      ( 1)
      Cost
      I/O
      |
      3.21948 << オブティマイザーで使用された見積もり行数
      301 << 実行時に収集された実行時統計の行数
      DTQ
      ( 2)
      75.3961
      NA
      |
      3.21948
      130
      HSJOIN
      ( 3)
      72.5927
      NA
      /--+---\
      674      260
      220      130

```

```

TBSCAN      TBSCAN
( 4)        ( 5)
40.7052     26.447
NA          NA
|          |
337        130
NA          NA << グラフ出力にはオブジェクトの実行時統計が含まれない
TABLE: FF   TABLE: FF
T1          T2

```

パーティション・データベース環境では、グラフに表示されるカーディナリティーは、実行時統計が収集されるデータベース・パーティションにおける平均カーディナリティーです。平均値が表示されるのは、それがオブティマイザーによって見積もられる値であるからです。見積もりの平均値と比較するため、実際の平均値を示すことには意味があります。また、データベース・パーティションごとのセクション実行時統計の明細が、演算子詳細出力に示されます。これらの詳細を調べることにより、合計 (全パーティションにわたるもの)、最小、最大などの他の情報を確認することができます。

db2exfmt コマンド出力の演算子詳細

Estimated number of rows が含まれている行に続くストリーム・セクションに、演算子の実際のカーディナリティーが表示されます (Explain 出力内の Actual number of rows)。パーティション・データベース環境では、演算子が複数のデータベース・メンバーで実行されている場合、表示される実行時統計のカーディナリティーは、その環境における平均のカーディナリティーです。データベース・パーティションごとの値は、Explain Actuals という別のセクションに表示されます。このセクションはパーティション・データベース環境でのみ表示され、シリアル・モードでは表示されません。特定のデータベース・パーティションで実行時統計を使用できない場合、そのパーティション番号の横のデータベース・パーティション別の値リストには NA と表示されます。Output Streams セクションの Actual number of rows にも NA と表示されます。

db2exfmt コマンドの演算子詳細出力の例を以下に示します。

```

9) UNION : (Union)
Cumulative Total Cost:    10.6858
Cumulative First Row Cost: 9.6526

Arguments:
-----
UNIONALL: (UnionAll Parameterized Base Table)
DISJOINT

Input Streams:
-----
5) From Operator #10

Estimated number of rows: 30
Actual number of rows:    63
Partition Map ID:         3

7) From Operator #11

Estimated number of rows: 16
Actual number of rows:    99
Partition Map ID:         3

```

Output Streams:

8) To Operator #8

Estimated number of rows: 30
Actual number of rows: 162
Partition Map ID: 3

Explain Actuals: << このセクションは、パーティション・データベース環境でのみ表示されます

DB Partition number Cardinality

 1 193
 2 131

db2exfmt コマンド出力のオブジェクト統計

Explain 出力には、アクセス・プランで使用されるオブジェクトごとの統計が含まれています。パーティション表およびパーティション索引の場合、統計はデータ・パーティションごとに示されます。パーティション・データベース環境または DB2 pureScale 環境では、統計はメンバーごとに示されます。特定のメンバーに関する統計が入手できない場合、そのメンバーに関する値のリストで、メンバー番号の横に NA と表示されます。

以下に、**db2exfmt** コマンドの出力に表示されるオブジェクト統計の例を示します。

Runtime statistics for objects Used in Access Plan:

Schema: GOSALES
Name: ORDER_DETAILS
Type: Table

Member 0

Metrics

lock_wait_time:85899
lock_wait_time_global:25769
lock_waits_local:21474
lock_waits_global:85899
lock_escals_local:17179
lock_escals_global:2
direct_writes:12884
direct_read_reqs:1
pool_data_gbp_invalid_pages:446
pool_data_lbp_pages_found:445
pool_xda_l_reads:446
pool_xda_p_reads:15

Explain 情報の使用ガイドライン

Explain 情報を使用すると、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したり、パフォーマンス・チューニングの成果を評価したりすることができます。

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。

1. 変更前の照会の Explain 情報をキャプチャーし、結果の Explain 表を保存します。または、**db2exfmt** Explain ツールの出力を保存します。
2. 現行のカタログ統計を保存または印刷します。 **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。
3. データ定義言語 (DDL) ステートメント (CREATE TABLE、CREATE VIEW、CREATE INDEX、または 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 または XQuery ステートメントの場合は、アプリケーションを最初に行うときに、この情報を収集することができます。静的 SQL および XQuery ステートメントの場合は、バインド時にこの情報を収集できます。パフォーマンスの変化を分析するには、収集した情報を、以前に収集したこの参照情報と比較します。

例えば、分析によって、アクセス・パスを判別するときに索引がもう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、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がきます。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る可能性があります。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、このリストから別のアクションを選択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更新、コンテナの追加、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、Explain 機能を使用して、変更によってどのような影響が選択したアクセス・プランにあるかを判別します。例えば、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 MQT 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 出力を使用すると、選択したアクセス・プランとその相対コストを判別できますが、特定の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。

Explain 情報の分析のガイドライン

EXPLAIN 情報は、主に照会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。 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。次の領域での索引の使用法が探せません。

- 結合述部
- ローカル述部
- GROUP BY 節
- ORDER BY 節
- WHERE XMLEXISTS 節
- 選択リスト

さらに Explain 機能を使用して、別の索引を使用するのが良いかまったく索引を使用しないのが良いかも評価できます。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てから、照会を再コンパイルします。時間が経過するにつれ、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 (Explain データを調べることによって) 気付くかもしれません。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率が現在低下している場合は、次のようにできます。

- その索引に合わせてデータをクラスタ化するためにその表を再編成する
- **RUNSTATS** コマンドを使用して索引と表の両方の統計を収集する
- 照会を再コンパイルする

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には、再コンパイルされた照会の Explain 出力を調べます。

- アクセス・タイプ

Explain 出力を分析して、実行しているアプリケーションのタイプにおいて通常は最適でないデータ・アクセスのタイプを探します。例えば、

- 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、範囲区切り述部を指定した索引スキャンの筆頭候補です。なぜなら、通常それらのアプリケーションは、キー列に対して等価述部で修飾された数行しか返さないためです。 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ていない理由を判別することができます。

- 表示専用照会

「ブラウズ」タイプの照会の検索基準は非常に不明確である場合があり、その場合には多数の修飾行が生成されることとなります。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。

例えば、マージ・スキャン結合とソート演算子の両方がアクセス・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 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。多くの行が含まれる結合 (意思決定支援照会での結合など) は、通常、ハッシュ結合やマージ結合を使用するとより速く実行します。数行しか含まれない結合 (OLTP 照会など) は、一般に、NESTED LOOP 結合を使用するとより速く実行します。しかし、どちらの場合にも、上記の一般的な結合の動作が変わる可能性のある酌量すべき状況があります。例えば、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。

REFRESH TABLE および SET INTEGRITY ステートメントによりアクセス・プランを使用し、パフォーマンス上の問題を自己診断する

REFRESH TABLE または SET INTEGRITY ステートメントに対して Explain ユーティリティを実行することにより、これらのステートメントに関するパフォーマンス上の問題を自己診断するために使用できるアクセス・プランを生成できます。これは、マテリアライズ照会表 (MQT) を適切に保守するのに役立ちます。

REFRESH TABLE または SET INTEGRITY ステートメントのアクセス・プランを取得するには、以下のいずれかの方法を使用します。

- EXPLAIN ステートメントで EXPLAIN PLAN FOR REFRESH TABLE または EXPLAIN PLAN FOR SET INTEGRITY オプションを使用する。
- CURRENT EXPLAIN MODE 特殊レジスターを EXPLAIN に設定してから、REFRESH TABLE または SET INTEGRITY ステートメントを発行し、その後 CURRENT EXPLAIN MODE 特殊レジスターを NO に設定する。

制約事項

- REFRESH TABLE および SET INTEGRITY ステートメントは再最適化に適格でないため、REOPT Explain モード (または Explain スナップショット) はこれらの 2 つのステートメントに適していません。
- EXPLAIN ステートメントの WITH REOPT ONCE 節 (指定された EXPLAIN 可能ステートメントを再最適化することを示す) は REFRESH TABLE および SET INTEGRITY に適していません。

シナリオ

このシナリオでは、EXPLAIN および REFRESH TABLE ステートメントからアクセス・プランを生成し、それを使用してパフォーマンス上の問題の原因を自己診断する方法を示します。

1. 表を作成し、そこにデータを入力します。例えば、

```

create table t (
  i1 int not null,
  i2 int not null,
  primary key (i1)
);

insert into t values (1,1), (2,1), (3,2), (4,2);

create table mqt as (
  select i2, count(*) as cnt from t group by i2
)
data initially deferred
refresh deferred;

```

2. 次のように、EXPLAIN および REFRESH TABLE ステートメントを発行します。

```
explain plan for refresh table mqt;
```

このステップの代わりに、次のようにして、SET CURRENT EXPLAIN MODE 特殊レジスターで EXPLAIN モードを設定することもできます。

```

set current explain mode explain;
refresh table mqt;
set current explain mode no;

```

3. **db2exfmt** コマンドを使用して、Explain 表の内容をフォーマットし、アクセス・プランを取得します。このツールは、インスタンスの `sqllib` ディレクトリーの `misc` サブディレクトリーにあります。

```
db2exfmt -d dbname -o refresh.exp -1
```

4. アクセス・プランを分析して、パフォーマンス上の問題の原因を判別します。上記の例で、T が大規模な表である場合、表スキャンのコストは非常に高くなります。索引を作成すると、照会のパフォーマンスが向上します。

Explain 情報の収集および分析用のツール

DB2 データベース・サーバーには、SQL または XQuery ステートメントでオペ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を提供する、包括的な Explain 機能があります。

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 および XQuery ステートメントに関する情報が含まれています。Explain 情報のキャプチャー、表示、および分析を柔軟に行うために必要ないくつかのツールを利用できます。

アクセス・プランを徹底的に分析するために使用できる照会オペティマイザーの詳細情報は、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。

- **db2exfmt** ツールを使用して、フォーマット設定された出力で Explain 情報を表示します。
- 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、出力の操作、さまざまな照会の間での比較、または同じ照会を時間を変えて実行した場合の比較を容易に行えます。

静的 SQL または XQuery ステートメントの 1 つ以上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、**db2expln** ツールを使用します。このユーティリティーは、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。オペティ

マイザー情報については示しません。 **db2expln** ツールは、生成されたアクセス・プランを調べることにより、実行時に行われる操作に関する比較的コンパクトな説明文で概要を示します。

コマンド行で実行する Explain ツールは、sqllib ディレクトリーの misc サブディレクトリーにあります。

次の表では、DB2 Explain 機能で利用できるさまざまなツール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 61. Explain 機能ツール

希望する特性	Explain 表	db2expln	db2exfmt
テキスト出力		はい	はい
「簡易」静的 SQL および XQuery 分析		はい	
静的 SQL および XQuery のサポート	はい	はい	はい
動的 SQL および XQuery のサポート	はい	はい	はい
CLI アプリケーションのサポート	はい		はい
DRDA® アプリケーション・リクエスターで使用可能	はい		
詳細オプティマイザー情報	はい		はい
複数ステートメントの分析に適合	はい	はい	はい
アプリケーション内部から情報にアクセス可能	はい		

これらのツールの他、IBM Data Studio バージョン 3.1 以降を使用して、SQL ステートメントまたは XPATH ステートメントの現行アクセス・プランのダイアグラムを生成することもできます。詳細については、Visual Explain を使ったアクセス・プランのダイアグラミングを参照してください。

Explain 時点で有効なカタログ統計の表示

Explain 機能を使用すると、ステートメントの Explain 時に有効であった統計がキャプチャーされます。こうした統計はシステム・カタログに格納されているものとは異なる可能性があります。特に、リアルタイム統計収集が有効になっている場合には、そう言えます。 Explain 表にデータが追加されているものの Explain スナップショットが作成されていない場合には、EXPLAIN_OBJECT 表に記録される統計は一部に限定されます。

Explain されているステートメントに関連したすべてのカタログ統計をキャプチャーするには、Explain 表にデータが追加されると同時に Explain スナップショットを作成し、その後、SYSPROC.EXPLAIN_FORMAT_STATS スカラー関数を使用してスナップショットのカタログ統計をフォーマット設定します。

db2exfmt ツールを使用して Explain 情報をフォーマット設定する場合、Explain スナップショットが収集されているのであれば、自動的に SYSPROC.EXPLAIN_FORMAT_STATS 関数が使用されてカタログ統計が表示されません。

SQL および XQuery Explain ツール

db2expln コマンドは、SQL または XQuery の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。

このツールを使用して、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 および XQuery ステートメントでは、**db2expln** を使用してシステム・カタログに保管されたパッケージを調べます。動的 SQL および XQuery ステートメントでは、**db2expln** を使用して照会キャッシュ内のセクションを調べます。

Explain ツールは、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にあります。**db2expln** 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 コマンドは、`db2expln.bnd`、`db2exsrv.bnd`、および `db2exdyn.bnd` ファイルを用いて、データベースに初めてアクセスする際にデータベースにバインドされます。

db2expln 出力の説明:

db2expln コマンドからの Explain 出力には、各パッケージのパッケージ情報とセクション情報が含まれます。

- パッケージ情報には、バインド操作の日付とそれに関連するバインド・オプションが含まれます。
- セクション情報には、セクション番号と、Explain される SQL または XQuery ステートメントが含まれます。

SQL または XQuery ステートメントに対して選択されたアクセス・プランに関する Explain 出力が、セクション情報の下に表示されます。

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表示されます。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。またこうした棒線は、各操作の有効範囲も示します。ずっと右側にある、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表示される前に処理されます。

選択されたアクセス・プランは、元の SQL ステートメントが補強されたもの、有効 SQL ステートメント (ステートメント・コンソリテーターが使用可能に設定されている場合)、出力に表示される XQuery ステートメントのいずれかに基づいています。コンパイラーの照会書き直しコンポーネントは SQL または XQuery ステートメントを同等であるものより効率的な形式に変換するので、Explain 出力に表示されるアクセス・プランは想定したものと実質的に異なる場合があります。

Explain 表、および `SET CURRENT EXPLAIN MODE` ステートメントを含む Explain 機能は、最適化に使用された実際の SQL または XQuery 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 または XQuery のようなステートメントの形式で示します。

db2expln からの出力を、Explain 機能の出力と比較するときに、演算子 ID オプション (`-opids`) は便利です。**db2expln** が explain 機能から新しい演算子の処理を開

始するたびに、演算子 ID 番号が、`Explain` されるプランの左側に印刷されます。演算子 ID は、異なる表示のアクセス・プランの中で、ステップを比較するために使用することができます。`Explain` 機能の出力の中の演算子と、`db2expln` によって示される操作とが、常に 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。

表アクセス情報:

`db2expln` 出力内のステートメントは、アクセスしている表の名前とタイプを示します。

正規表の情報には、以下のいずれかの表アクセス・ステートメントが含まれます。

```
Access Table Name = schema.name ID = ts,n
Access Hierarchy Table Name = schema.name ID = ts,n
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・ビュー項目内での表に対応する TABLESPACEID と TABLEID です。

一時表の情報には、以下のいずれかの表アクセス・ステートメントが含まれます。

```
Access Temp Table ID = tn
Access Global Temp Table ID = ts,tn
```

ここで、ID は SYSCAT.TABLES カタログ・ビュー項目内での表 (`ts`) に対応する TABLESPACEID か、`db2expln` (`m`) によって割り当てられた対応する ID です。

表アクセス・ステートメントの後に、アクセスをさらに詳しく記述するために以下の追加のステートメントが提供されています。

- 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
- ロック・インテント
- 述部
- その他

列数ステートメント

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・ステートメント

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
```

このステートメントがない場合は、表は ORGANIZE BY DIMENSIONS 節を使用しないで作成されました。

並列スキャン・ステートメント

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、表を並列に読み取ることを示します。

```
Parallel Scan
```

このステートメントがない場合は、表の読み取りが 1 つのエージェント (またはサブエージェント) のみによって行われます。

スキャン方向ステートメント

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
```

このステートメントがない場合は、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・ステートメント

次のステートメントのうちの 1 つが、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かった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別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:
```

詳細は次のとおりです。

- *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
Index over XML data
```

その後には、索引の列ごとに出力の 1 つの行が続きます。この情報の有効な形式は、以下のとおりです。

```
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、以下のステートメントによって示されます。

```
#Key Columns = n
| Start Key: xxxxx
| Stop Key: xxxxx
```

ここで、xxxxx は以下のいずれかです。

- Start of Index
- End of Index
- Inclusive Value: または 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、キーの各部の項目を構成する以下の行のいずれかによって判別されます。

```
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・ストリングの最初の 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省略符号 (...) 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。これは、値として疑問符 (?) で示されます。

- 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、索引プリフェッチに対して順次検出プリフェッチが使用可能になっていることを示しています。このタイプのプリフェッチの MAXPAGES 値は、x で表されています。

```
Index Prefetch: Sequential (x)
```


- 次のステートメントは、索引プリフェッチに対して先読みプリフェッチが使用可能になっていることを示しています。

Index Prefetch: Readahead

- 次のステートメントは、索引プリフェッチに対して順次検出プリフェッチと先読みプリフェッチが使用可能になっていることを示しています。また、順次検出プリフェッチの MAXPAGES 値が x で表されることも示しています。

Index Prefetch: Sequential (x), Readahead

- 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・プリフェッチに対して順次検出プリフェッチが使用可能になっていることを示しています。このタイプのプリフェッチの MAXPAGES 値は、 x で表されています。

Data Prefetch: Sequential (x)

- 次のステートメントは、データ・プリフェッチに対して先読みプリフェッチが使用可能になっていることを示しています。

Data Prefetch: Readahead

- 次のステートメントは、データ・プリフェッチに対して順次検出プリフェッチと先読みプリフェッチが使用可能になっていることを示しています。また、順次検出プリフェッチの MAXPAGES 値が x で表されることも示しています。

Data Prefetch: Sequential (x), Readahead

- 索引マネージャ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、それらの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、次のステートメントで示します。

Fetch Direct Using Block or Row IOs

ロック・intent・ステートメント

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 Weak Exclusive
- None
- Share
- Update

述部ステートメント

アクセス・プランで使用される述部に関する情報を提供するステートメントには、3つのタイプがあります。

- 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 Predicate(s)
| #Predicates = n
```

- 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。この数には、集計またはソートなどのプッシュダウン操作は含まれません。

```
Sargable Predicate(s)
| #Predicates = n
```

- 次のステートメントは、一度データが戻された後に、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、照会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、表アクセスに使用されている分離レベルが、ステートメントの分離レベルとは異なる場合に表示されます。

```
Isolation Level: xxxx
```

これには、考えられる理由が幾つかあります。例えば、

- パッケージが反復可能読み取り (RR) 分離レベルにバインドされており、特定の参照整合性制約に影響を与える。こうした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 (CS) 分離レベルにダウングレードします。
- パッケージが非コミット読み取り (UR) 分離レベルにバインドされており、DELETE ステートメントが含まれている。削除操作のために表にアクセスすると、CS 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 **sortheap** 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・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

- 以下のステートメントは、表に揮発性のカーディナリティー属性セットがあることを示します。

Volatile Cardinality

一時表の情報:

一時表は、アクセス・プラン実行時の作業表として使用されます。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、または中間結果が使用可能メモリーに納まらないときに、一時表が使用されます。

一時表が必要な場合、以下のいずれかのステートメントが **db2expln** コマンドの出力に表示されます。

```

Insert Into Temp Table ID = tn      --> ordinary temporary table
Insert Into Shared Temp Table ID = tn  --> ordinary temporary table will be created
                                         by multiple subagents in parallel
Insert Into Sorted Temp Table ID = tn  --> sorted temporary table
Insert Into Sorted Shared Temp Table ID = tn  --> sorted temporary table will be created
                                         by multiple subagents in parallel

Insert Into Global Temp Table ID = ts,tn  --> declared global temporary table
Insert Into Shared Global Temp Table ID = ts,tn  --> declared global temporary table
                                                    will be created by multiple subagents
                                                    in parallel
Insert Into Sorted Global Temp Table ID = ts,tn  --> sorted declared global temporary table
Insert Into Sorted Shared Global Temp Table ID = ts,tn  --> sorted declared global temporary
                                                         table will be created by
                                                         multiple subagents in parallel

```

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。

これらの各ステートメントの後には、次の行が続きます。

```
#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
- マージ結合

- '= ANY' subquery
- '<> ALL' subquery
- INTERSECT または EXCEPT
- UNION (ALL キーワードの指定なし)

ソート済み一時表に関連付けられている多数のステートメントが、**db2expln** コマンド出力に表示される可能性があります。

- 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・キーの各列に関して、次の行のうちのいずれかが表示されます。

```
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、実行時に最適なソート・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、以下のステートメントが表示されません。

```
Sort Limited To Estimated Row Count
```

- 対称マルチプロセッサ (SMP) 環境で行われるソートの場合は、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
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持して、別の一時表には入れません。

- 次のステートメントは、ソート操作中に重複値を除去することを示しています。

```
Duplicate Elimination
```

- ソート操作中に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が表示されます。

```
Partial Aggregation
Intermediate Aggregation
Buffered Partial Aggregation
Buffered Intermediate Aggregation
```

一時表の完了

一時表が表アクセスの有効範囲内で作成される場合には、いつでも完了ステートメントが表示されます。このステートメントは、以下のいずれかになります。

```
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定義関数 (UDF) のことです。表関数は、関数の属性を詳述する以下のステートメントによって指示されます。特定の名称は、起動される表関数を一意的に識別します。

```
Access User Defined Table Function
| Name = schema.funcname
| Specific Name = specificname
| SQL Access Level = accesslevel
| Language = lang
| Parameter Style = parmstyle
| Fenced
| Called on NULL Input          Not Deterministic
| Not Federated                 Disallow Parallel
|                               Not Threadsafe
```

バイナリー結合情報:

db2expln コマンドからの出力には、EXPLAIN されたステートメント内の結合に関する情報が含まれる可能性があります。

バイナリー結合が実行されると必ず、以下のステートメントのいずれかが表示されます。

```
Hash Join
Merge Join
Nested Loop Join
```

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
```

マージ結合またはネスト・ループ結合の場合、結合の外部表は、(出力に表示される)直前のアクセス・ステートメント内で参照された表です。結合の内部表は、結合ステートメントの有効範囲内に含まれるアクセス・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、アクセス・ステートメントが逆方向になります。つまり、外部表は結合の有効範囲内に含まれ、内部表は結合の前に表示されます。

ハッシュ結合またはマージ結合の場合、次のような追加のステートメントが表示されることがあります。

- Early Out: Single Match Per Outer Row

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どうかだけを判別する必要があります。

- Residual Predicate(s)
| #Predicates = n

結合が完了したあとで、述部を適用することもできます。このステートメントには、適用される述部の数が表示されます。

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ることがあります。

- Process Hash Table For Join

ハッシュ表は内部表から作成されます。このステートメントは、内部表へのアクセス時にハッシュ表の作成が述部にプッシュダウンされた場合に表示されます。

- Process Probe Table For Hash 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。このステートメントは、外部表へのアクセス時にプローブ表が作成された場合に表示されます。

- Estimated Build Size: n

このステートメントは、ハッシュ表の作成に必要な見積もりバイト数を表示します。

- Estimated Probe Size: n

このステートメントは、プローブ表の作成に必要な見積もりバイト数を表示します。

ネストしたループ結合の場合、次の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ることがあります。

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。例えば、**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. 表 W から修飾行を取り出す。
2. W からの行を、表 X からの次の行と結合し、結果 P1 (部分結合の結果番号 1) を呼び出す。
3. P1 を、表 Y からの次の行と結合して、P2 を作成する。
4. P2 を、表 Z からの次の行と結合し、完全な結果行を 1 つ作成する。
5. Z にさらに行があれば、ステップ 4 に戻る。
6. Y にさらに行があれば、ステップ 3 に戻る。
7. X にさらに行があれば、ステップ 2 に戻る。
8. W にさらに行があれば、ステップ 1 に戻る。

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、**db2expln** 出力内の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固有な 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かの特性があり、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  
Not Piped
```

実行時に十分なメモリーがないと、パイプ接続されたデータ・ストリームがディスクに書き込まれることがあります。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
```

挿入、更新、および削除の情報:

INSERT、UPDATE、または DELETE ステートメントの Explain テキストは説明を要しません。

db2expln 出力内のこれらの SQL 操作で使用できるステートメント・テキストは、以下のとおりです。

```
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
```

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、表がアクセスされる前に、修飾行およびブロック ID をソートしておき、重複行を削除しておくか (索引 ORing の場合)、またはアクセスされているすべての索引に現れる ID を識別するための手法を使用すると (索引 ANDing の場合)、効率が良くなります。

Explain 出力に表示される ID 準備情報を使用する方法には、主に以下の 3 通りがあります。

- 次のステートメントはいずれも、索引 ORing を使用して修飾 ID のリストを準備しました。

```
Index ORing Preparation
Block Index ORing Preparation
```

索引 ORing は、複数の索引にアクセスし、その結果を結合して、いずれかの索引に現れる識別可能な 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 ORing を考慮します。

- 次のステートメントはいずれも、リスト・プリフェッチの際に入力データが準備されて使用されたことを示します。

```
List Prefetch Preparation
Block List Prefetch RID Preparation
```

- 索引 ANDing とは、複数の索引にアクセスし、その結果を結合して、アクセスされるすべての索引に現れる ID を組み込む技法を指します。索引 ANDing は、以下のいずれかのステートメントで開始されます。

```
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
```

索引 ANDing フィルター操作は、ID を処理し、ビット・フィルター手法を使用して、アクセスされるすべての索引に現れる ID を判別します。下記のステートメントは、索引 ANDing 用に ID が処理されたことを示します。

```
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
```

リスト・プリフェッチを任意のタイプの ID 準備に実行できる場合には、以下のステートメントによってそのことが示されます。

```
Prefetch: Enabled
```

集約の情報:

SQL ステートメント内の述部によって示される基準を満たす行で、集約が実行されます。

集約関数が実行されると、以下のいずれかのステートメントが **db2exp1n** 出力に表示されます。

```
Aggregation  
Predicate Aggregation  
Partial Aggregation  
Partial Predicate Aggregation  
Hashed Partial Aggregation  
Hashed Partial Predicate Aggregation  
Intermediate Aggregation  
Intermediate Predicate Aggregation  
Final Aggregation  
Final Predicate Aggregation
```

述部集約とは、データにアクセスした際に、集約操作が述部として処理されたことを表します。

集約ステートメントの後には、実行された集約関数のタイプを示す別のステートメントが続きます。

```
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。

述部集約が実行された場合、集約完了操作とそれに対応する出力が存在します。

```
Aggregation Completion  
Partial Aggregation Completion  
Hashed Partial Aggregation Completion  
Intermediate Aggregation Completion  
Final Aggregation Completion
```

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、特別なアクセス・プラン操作が必要になります。

- 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、**db2exp1n** コマンドからの出力にある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 つ以上のデータベース・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

- 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、ステートメントで DBPARTITIONNUM() スカラー関数が使用される場合に共通です。)

Directed by Node Number

- 以下のステートメントは、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データベース・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、ステートメントで HASHEDVALUE スカラー関数が使用される場合に共通です。)

```
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・データベース・パーティション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時に判別された、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 (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。

- 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、以下のいずれかのステートメントで記述されます。

各行は、コーディネーター・データベース・パーティションに送られます。

```
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、各行が送られます。

```
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、即時に送信できなかった行を扱う「完了」ステートメントがその後に表示されます。その場合、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
```

これらのステートメントの後には常に、検索される列の数の表示が付いていません。

```
#Columns = n
```

- 表キューが受信端で行をソートする場合は、以下のいずれかのステートメントが表示されます。

```
Output Sorted
Output Sorted and Unique
```

これらのステートメントの後には、ソート操作に使用されるキー数が続きます。

```
#Key Columns = n
```

ソート・キー中の列ごとに、次のいずれかのステートメントが表示されます。

```
Key n: (Ascending)  
Key n: (Descending)
```

- 表キューの受信端で述部が行に適用される場合は、次のステートメントが表示されます。

```
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、サブセクションの先頭まで、明示的にループバックし、次のステートメントが表示されます。

```
Jump Back to Start of Subsection
```

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。

db2exp1n コマンドからの次の出力は、データ・ソースが読み取られることを示しています。

```
Ship Distributed Subquery #n  
| #Columns = n
```

分散副照会から戻されるデータに述部が適用されると、適用される述部の数が以下のステートメントによって示されます。

```
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、以下のいずれかのステートメント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には、以下のステートメントが表示されます。

```
Ship Distributed Lock Table #n
```

データ・ソースに対するデータ定義言語 (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、以下のステートメント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・データベース・パーティションで実行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。

```
Distributed DDL Statement #n Completion
```

各分散サブステートメントの詳細は、個別に表示されます。

- 副照会のデータ・ソースは、以下のいずれかのステートメントで示されます。

```
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表示されます。

```
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表示されます。

```
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、値の数は以下のステートメントによって示されます。

```
#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、値の数は以下のステートメントによって示されます。

```
#Output Columns: n
```

その他の Explain 情報:

db2exp1n コマンドからの出力には、簡単に分類できない、有用な追加情報が含まれています。

- データ定義言語 (DDL) ステートメントのセクションは、次のステートメントの出力に示されます。

```
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) に関する SET ステートメントのセクションは、次のステートメントの出力に示されます。

```
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 節が含まれる場合、次のステートメン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、重複値を除去するために最初に行をソートする必要があります。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。

- **partial early distinct (PED)** 操作を実行して、(全部ではないかもしれないが) 多くの重複を除去した場合、次のステートメントが表示されます。これにより、後で照会を評価するときに処理しなければならないデータの量が少なくなります。

Hashed partial distinct filter

- 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ものの、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

- 以下のステートメントは、SQL ステートメントに **UNION** 演算子が含まれ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、**CASE** ステートメントなどの活動や、参照整合性制約やトリガーなどの内部機構を実行するときに表示されます。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
```

- 特定の更新操作と削除操作の前に、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
```

- ロールアウト最適化に適格なマルチディメンション・クラスタリング表に対する削除操作については、以下のいずれかのステートメントが表示されます。

```
CELL DELETE with deferred cleanup
CELL DELETE with immediate cleanup
```

- 次のステートメントは、行がアプリケーションに戻っている場合に表示されません。

```
Return Data to Application
| #Columns = n
```

操作が表アクセスにプッシュダウンされた場合、以下の完了フェーズ・ステートメントが出力に表示されます。

```
Return Data Completion
```

- ストアド・プロシージャが呼び出されている場合、以下のステートメントが表示されます。

```
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
```

- 以下のステートメントは、1 つ以上のラージ・オブジェクト (LOB) ロケーターが解放されている場合に表示されます。

```
Free LOB Locators
```

照会アクセス・プランの最適化

ステートメント・コンセントレーターによるコンパイル・オーバーヘッドの減少

ステートメント・コンセントレーターは、データベース・サーバーで動的 SQL ステートメントを変更し、全く同じではないものの類似している SQL ステートメントにおいて同じアクセス・プランを共有できるようにします。

オンライン・トランザクション処理 (OLTP) では、異なるリテラル値を持つ単純ステートメントが繰り返し生成される場合があります。そのようなワークロードでは、ステートメントを再コンパイルすると、かなりのオーバーヘッドが余分に生じる可能性があります。ステートメント・コンセントレーターは、リテラルの値にかかわらず、コンパイル済みステートメントを再使用できるようにしてこのオーバーヘッドを回避します。

デフォルトでは、ステートメント・コンセントレーターは使用不可です。それは、**stmt_conc** データベース構成パラメーターを **LITERALS** に設定すると、データベース内のすべての動的ステートメントで使用可能にできます。

ステートメント・コンセントレーターは、着信動的 SQL ステートメントを変更することによってパフォーマンスを改善します。ステートメント・コンセントレーターに適したワークロードでは、パッケージ・キャッシュに既に存在するステートメントを再利用することによって得られるコスト節減を考えれば、着信 SQL ステートメントの変更に関連するオーバーヘッドは小さなものです。

ステートメント集中の結果として動的ステートメントが変更された場合、元のステートメントと変更後のステートメントの両方が **Explain** 出力に表示されます。ステートメント・コンセントレーターが元のステートメント・テキストを変更した場合、イベント・モニター論理モニター・エレメントだけでなく、**MON_GET_ACTIVITY_DETAILS** 表関数からの出力にも元のステートメントが示されます。他のモニター・インターフェースでは、変更後のステートメント・テキストのみが示されます。

例えば、**stmt_conc** データベース構成パラメーターが **LITERALS** に設定されていて、次の 2 つのステートメントが実行されるとします。

```
select firstme, lastname from employee where empno='000020'  
select firstme, lastname from employee where empno='000070'
```

これらのステートメントはパッケージ・キャッシュで同じ項目を共有し、その項目では次のステートメントが使用されます。

```
select firstme, lastname from employee where empno=:L0
```

データ・サーバーは、元のステートメントで使用されていたリテラルに基づいて、**:L0** の値 ('000020' かまたは '000070') を提供します。

ステートメント集中はステートメント・テキストを変更するため、アクセス・プランの選択に影響を与えます。パッケージ・キャッシュ内の類似ステートメントが類似アクセス・プランを持つ場合に、ステートメント・コンセントレーターを使用するようにしてください。ステートメント内のリテラル値によってアクセス・プランが大きく異なってくる場合は、そのステートメントに対してステートメント・コンセントレーターを使用可能にすべきではありません。

ステートメント・コンセントレーターを使用すると、**VARCHAR** および **VARGRAPHIC** のストリング・リテラルの長さ属性が、ストリング・リテラルの長さより大きくなる場合があります。

アクセス・プランの再利用

バインドまたは再バインド操作を繰り返しても、パッケージ内の静的 SQL ステートメント用に選択されたアクセス・プランが、既存のアクセス・プランのまま変わらないように、あるいはそれと非常に類似したものとなるように要求することができます。

アクセス・プランの再利用により、明示的な承認なしにプランに大きな変更が加えられないようにすることができます。この場合、照会がアクセス・プランの改善見込みの恩恵を受けないことにもなり得ますが、アクセス・プランの再利用に備わっ

ている制御により、そのような改善をテストし、準備ができればそれをインプリメントすることが可能です。そのときまで、安定した、予測可能なパフォーマンスの既存のアクセス・プランを使用し続けることができます。

アクセス・プランの再利用は、ALTER PACKAGE ステートメントを介して、または BIND、REBIND、または PRECOMPILE コマンドの APREUSE オプションを使用することによって、使用可能にできます。アクセス・プランの再利用の対象であるパッケージは、SYSCAT.PACKAGES カタログ・ビューの APREUSE 列の値が Y です。

ALTER_ROUTINE_PACKAGE プロシージャは、SQL プロシージャのようなコンパイルされた SQL オブジェクトでアクセス・プランの再利用を可能にする便利な方法です。ただし、再バインドされる前にオブジェクトがドロップされるため、アクセス・プランは、コンパイルされたオブジェクトの妥当性再検査中には再利用できません。この場合、APREUSE はパッケージが次にバウンドまたは再バインドされるときに初めて有効になります。

アクセス・プランの再利用が最も有効なのは、スキーマおよびコンパイル環境への変更が最小に保たれるときです。大きな変更が行われる場合、以前のアクセス・プランを再作成できない場合があります。そのような大きな変更の例には、アクセス・プランで使用されていた索引のドロップ、または異なる最適化レベルでの SQL ステートメントの再コンパイルが含まれます。また、照会コンパイラーによるステートメントの分析に大きな変更があった場合にも、以前のアクセス・プランが再利用できなくなります。

アクセス・プランの再利用を最適化ガイドラインと組み合わせることができます。ステートメント・レベルのガイドラインは、それが適用される静的 SQL ステートメントに対するアクセス・プランの再利用よりも優先されます。指定されている一般的な最適化ガイドラインと競合しない場合、ステートメント・レベルのガイドラインのない静的ステートメントに対するアクセス・プランを再利用できます。空のガイドラインを持つステートメント・プロファイルを使用することにより、特定のステートメントではアクセス・プランの再利用を無効にする一方で、パッケージ内の他の静的ステートメントにはプランの再利用を有効なままにすることができます。

注: バージョン 9.7 より前のリリースによって作成されたパッケージからのアクセス・プランは、再利用できません。

アクセス・プランが再利用できない場合、コンパイルが続行されますが、警告 (SQL20516W) が、アクセス・プランを再利用する試行が成功しなかった理由を示す理由コードと共に戻されます。追加の情報が、EXPLAIN 機能を介して使用可能な診断メッセージで時々提供されます。

最適化クラス

SQL または XQuery ステートメントをコンパイルするときは、オプティマイザーによる、そのステートメント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。

最適化クラスごとに、照会のコンパイル中に考慮される最適化方針の数およびタイプが異なります。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。

SQL または XQuery ステートメント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、アクセス・プランの生成時に、オプティマイザーが最小限の最適化を使用するよう指示します。このクラスの特徴は以下のとおりです。
- オプティマイザーは、頻出値統計を考慮しません。
 - 基本照会書き直し規則のみを適用します。
 - 貪欲型結合列挙を使用します。
 - ネストされたループ結合および索引スキャン・アクセス方式だけを使用可能にします。
 - リスト・プリフェッチを、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ません。

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 または XQuery 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、次の特徴があります。
- オプティマイザーは、頻出値統計を考慮しません。
 - 照会書き直し規則のサブセットのみを適用します。
 - 貪欲型結合列挙を使用します。
 - リスト・プリフェッチを、生成されたアクセス方式で使用されないようにします。

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、複雑な照会のコンパイル・コストをクラス 3 以上に比べてはるかに低く抑えつつ、クラス 1 よりも大幅に高い水準の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、次の特徴があります。
- 使用可能な統計すべて (頻出値および変位値の統計を含む) を使用します。
 - すべての照会書き直し規則 (マテリアライズ照会表の経路指定を含む) を適用します。ただし、めったに該当することがない、計算を多用する規則を除きます。
 - 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結合列挙ではなく欲張り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、欲張り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、特定の照会が厳密に同じ仕方で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 このクラスは中程度の最適化を表し、DB2 for z/OS の照会最適化特性に最もよく合致します。この最適化クラスには、次の特性があります。
- 使用可能な場合は、頻出値統計が使用されます。
 - 「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。
 - 以下において、動的プログラミング結合列挙が使用されます。
 - 複合内部表の限定使用
 - 参照表に関係するスター・スキーマに対するデカルト積の限定使用
 - 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、さまざまな種類のアプリケーションに適しており、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。

- 5 このクラスは、オプティマイザーが高水準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。このクラスの特徴は以下のとおりです。
- 使用可能な統計すべて (頻出値および変位値の統計を含む) を使用します。
 - すべての照会書き直し規則 (マテリアライズ照会表の経路指定を含む) を適用します。ただし、めったに該当することがない、計算を多用する規則を除きます。
 - 以下において、動的プログラミング結合列挙が使用されます。
 - 複合内部表の限定使用
 - 参照表に関係するスター・スキーマに対するデカルト積の限定使用
 - リスト・プリフェッチ、索引の ANDing、およびマテリアライズ照会表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。

最適化クラス 5 (デフォルト) は、トランザクション処理と複合的な照会の両方を伴う混合環境に適しています。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。

複雑な動的 SQL または XQuery ステートメントのため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、オプティマイザーが検出した場合、最適化は縮小されます。縮小のエクステン트는、マシンのサイズと述部の数によって決まります。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のすべてを適用し続けます。しかし、照会オプティマイザーは欲張り型結合列挙を使用するため、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。

- 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。このクラスは最適化クラス 5 と同様です。ただしこの場合は、オプティマイザーが複雑な動的 SQL または XQuery ステートメントの照会最適化の量を縮小しようとしなない点で異なります。

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増加します。大規模な表を使用する非常に複雑な照会または非常に長時間実行する照会において、より包括的な最適化によってさらに優れたアクセス・プランを生成できるかどうかを調べる際に、このクラスを使用できます。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。

最適化クラスの選択:

最適化クラスを設定すると、最適化手法を明示的に指定できるという利点があります。

これは、特に以下の場合にそう言えます。

- 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
- ステートメント準備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。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 および XQuery 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、その照会のコンパイルおよび実行に費やされる時間を判別します。コンパイル時間が長すぎる場合は、最適化のクラスを低くしてください。実行時間が問題となる場合は、最適化のクラスを高くすること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選択する場合は、まず 1、2、3 のいずれかのクラスで試します。クラス 0、1、および 2 は、欲張り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。この種の照会には、以下の傾向があります。
 - 単一の表または少しの表だけにアクセスする
 - 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する
 - オンライン・トランザクション処理 (OLTP) に関与する
- 30 秒を超え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 以上では動的プログラミング結合列挙アルゴリズムが使用されます。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。以下の特性を持つ照会には、より高い最適化クラスの使用を考慮してください。

- 大きい表にアクセスする
- ビューの数が多い
- 述部の数が多い
- 副照会が多い
- 結合が多い
- UNION または INTERSECT などのセット演算子が多い
- 適格となる行が多い
- GROUP BY および HAVING 操作
- ネストした表式

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 および XQuery ステートメントには、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照会を生成します。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・プランを選択する必要があります。照会最適化クラス 2 以上を使用すると、このような照会でも改善することができます。

SAP アプリケーションの場合、常に最適化クラス 5 を使用してください。この最適化クラスは、**DB2_REDUCED_OPTIMIZATION** レジストリー変数の設定など、SAP 用に最適化された多くの DB2 フィーチャーを使用可能にします。

フェデレーテッド・データベースでは、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 および XQuery ステートメントを使用しているか、また、同一の動的照会が繰り返し実行されるかどうかを考慮してください。

このタスクについて

静的 SQL および XQuery ステートメントの場合、照会コンパイル時間とリソースは 1 回だけ費やされ、その結果得られるプランは何度も使用できます。一般に、静的 SQL および XQuery ステートメントは常にデフォルトの照会最適化クラス (5) を使用します。動的ステートメントは実行時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 または XQuery 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となります。このステートメントには、静的 SQL および XQuery 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。

さらなる最適化が照会にとってメリットとなるかどうかははっきりしない場合や、コンパイル時間およびリソース使用量が気になる場合は、ベンチマーク・テストを考慮してください。

手順

照会最適化クラスを指定するには、以下のようになります。

1. パフォーマンス要因を分析します。

- 動的照会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。

$$\frac{\text{コンパイル時間} + \text{すべての反復の実行時間の合計}}{\text{反復回数}}$$

反復回数は、コンパイルごとのステートメント実行回数の見積もりを表します。

注: 初期コンパイルの後、動的 SQL および XQuery 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いつでも、再コンパイルされます。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、キャッシュされたステートメントは後続の PREPARE ステートメントで再使用されません。

- 静的 SQL および XQuery 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 および XQuery ステートメントのコンパイル時間にも関心があるとしても、静的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推定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、通常は実行時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. 最適化クラスを指定します。

- 動的 SQL および XQuery ステートメントでは、**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。例えば、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 または XQuery 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ます。

CURRENT QUERY OPTIMIZATION 特殊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。動的および静的照会のデフォルト値は両方とも、**dft_queryopt** データベース構成パラメーターの値 (デフォルト値は 5) によって指定されます。また、バインド・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 および XQuery ステートメントでは、**PREP** コマンドおよび **BIND** コマンドで指定される最適化クラスが使用されます。 **SYSCAT.PACKAGES** カタログ・ビュー内の **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ステートメントに使用されます。この静的 SQL および XQuery 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、データ・サーバーは、**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クラスを使用します。

他のチューニング・オプションによる結果が不十分な場合に、最適化プロファイルを使用する

ベスト・プラクティス推奨事項に従っても、依然として最適なパフォーマンスが得られていないようなら、明示的な最適化ガイドラインを DB2 オプティマイザーに適用できます。

これらの最適化ガイドラインは、最適化プロファイルという XML 文書に記述されています。プロファイルでは、SQL ステートメントおよび関連する最適化ガイドラインが定義されています。

最適化プロファイルを広範囲に使用するなら、保守に多くの労力を要します。さらに重要な点として、最適化プロファイルは、既存の SQL ステートメントのパフォーマンス向上のためにのみ使用することができます。一貫してベスト・プラクティスに従うなら、将来加わるものも含めすべての照会について、安定した照会パフォーマンスを実現するのに役立ちます。

最適化プロファイルと最適化ガイドライン

最適化プロファイルは、1 つ以上の SQL ステートメントについての最適化ガイドラインを含めることのできる XML 文書です。各 SQL ステートメントとそれに関連付ける最適化ガイドラインとの間の対応は、SQL テキストおよび SQL ステートメントを明確に識別するために必要な他の情報を使用して確立されます。

DB2 オプティマイザーは、業界でも最高の性能を備えたコスト・ベースのオプティマイザーの 1 つです。とはいえ、まれにこのオプティマイザーも、必ずしも最適とは言えない実行プランを選択してしまうことがあります。DBA はデータベースに精通しているため、**db2advise**、**runstats**、**db2expln**、および最適化クラス設定などのユーティリティを使用して、データベースのパフォーマンスをより高めるためのオプティマイザーの調整に役立てることができます。しかし、すべてのチューニング・オプションを使い尽くしても期待した結果が得られない場合には、DB2 オプティマイザーに明示的な最適化ガイドラインを提供できます。

例えば、データベース統計の更新と、他のすべてのチューニング・ステップの実行を済ませたものの、オプティマイザーが、以下の副照会での **SUPPLIERS** 表へのアクセスのために **I_SUPPKEY** 索引をまだ選択していないとします。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM PARTS P, SUPPLIERS S, PARTSUPP PS
WHERE P.PARTKEY = PS.PS_PARTKEY
      AND S.S_SUPPKEY = PS.PS_SUPPKEY
      AND P.P_SIZE = 39
      AND P.P_TYPE = 'BRASS'
      AND S.S_NATION = 'MOROCCO'
      AND S.S_NATION IN ('MOROCCO', 'SPAIN')
      AND PS.PS_SUPPLYCOST = (SELECT MIN(P.S1.PS_SUPPLYCOST)
                              FROM PARTSUPP PS1, SUPPLIERS S1
                              WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                                    AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                                    AND S1.S_NATION = S.S_NATION))
```

この場合、明示的な最適化ガイドラインを使用して、オプティマイザーに影響を与えることができます。例えば、

```
<OPTGUIDELINES><IXSCAN TABLE="S" INDEX="I_SUPPKEY"/></OPTGUIDELINES>
```

最適化ガイドラインは、単純な XML 仕様を使用して指定されます。

OPTGUIDELINES エレメント内のそれぞれのエレメントは、DB2 オプティマイザーによって最適化ガイドラインとして解釈されます。この例には、1 つの最適化ガイドライン・エレメントがあります。**IXSCAN** エレメントは、オプティマイザーが索引アクセスを使用することを要求します。**IXSCAN** エレメントの **TABLE** 属性は、ターゲット表参照 (表参照の直接的な名前を使用) を示し、**INDEX** 属性は索引を指定します。

以下の例は、前の照会に基づいており、最適化プロファイルを使用してどのように最適化ガイドラインを DB2 オプティマイザーに渡すことができるのかを示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE VERSION="9.1.0.0">
<STMTPROFILE ID="Guidelines for TPCD Q9">
  <STMTKEY SCHEMA="TPCD">
    SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
    FROM PARTS P, SUPPLIERS S, PARTSUPP PS
    WHERE P.PARTKEY = PS.PS_PARTKEY
          AND S.S_SUPPKEY = PS.PS_SUPPKEY
          AND P.P_SIZE = 39
          AND P.P_TYPE = 'BRASS'
          AND S.S_NATION = 'MOROCCO'
          AND S.S_NATION IN ('MOROCCO', 'SPAIN')
          AND PS.PS_SUPPLYCOST = (SELECT MIN(P.S1.PS_SUPPLYCOST)
```



```

FROM PARTSUPP PS1, SUPPLIERS S1
WHERE P.P_PARTKEY = PS1.PS_PARTKEY
AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
AND S1.S_NATION = S.S_NATION))

</STMTKEY>
<OPTGUIDELINES><IXSCAN TABLE="S" INDEX="I_SUPPKEY" /></OPTGUIDELINES>
</STMTPROFILE>
</OPTPROFILE>

```

各 STMTPROFILE 要素は、1 つのアプリケーション・ステートメントに対して一連の最適化ガイドラインを提供します。ターゲットとなるステートメントは、STMTKEY サブエレメントによって識別されます。次いで、最適化プロファイルはスキーマ修飾名を付けられて、データベースに挿入されます。挿入された最適化プロファイルは、**BIND** コマンドや **PRECOMPILE** コマンドでこの修飾名が指定されたときに、ステートメントに対して施行されます。

最適化プロファイルを使用すると、アプリケーションやデータベースの構成を変更することなく、オプティマイザーに最適化ガイドラインを渡すことができます。行うことは、ただ単純な XML 文書を作成し、それをデータベースに挿入して、**BIND** コマンドや **PRECOMPILE** コマンドでその最適化プロファイルの名前を指定するだけです。オプティマイザーにより最適化ガイドラインと該当するステートメントが、自動的にマッチングされます。

最適化ガイドラインは、包括的なものにする必要はありませんが、希望する実行プランをターゲットにしたものでなければなりません。DB2 オプティマイザーは、ガイドラインがあっても、他の候補となるアクセス・プランを引き続き考慮し、既存のコスト・ベースのメソッドを使用して機能します。最適化ガイドラインは、特定の表参照をターゲットにしていれば、通常最適化の設定をオーバーライドできません。例えば、表 A と B の間のマージ結合を指定する最適化ガイドラインは、最適化クラス 0 では無効です。

オプティマイザーは、無効な最適化ガイドラインや適用できない最適化ガイドラインは無視します。無視された最適化ガイドラインがある場合、実行プランが作成され、理由コード 13 で SQL0437W の警告が戻されます。その後で、最適化ガイドラインの処理に関する詳細な診断情報は、EXPLAIN ステートメントを使用して入手できます。

最適化プロファイル:

最適化プロファイルの分析:

最適化プロファイルにはグローバル最適化ガイドラインを含めることができます。また、最適化プロファイルには、パッケージ内の個々の DML ステートメントに適用される特定の最適化ガイドラインを含めることもできます。グローバル最適化ガイドラインは、最適化プロファイルが有効になっている間に実行されるすべてのデータ操作言語 (DML) ステートメントに適用されます。

例えば、

- グローバル最適化ガイドラインを作成できます。これは、現行最適化プロファイルがアクティブである間に処理されるステートメントについて、オプティマイザーがマテリアライズ照会表 (MQT) Test.SumSales および Test.AvgSales を参照するように要求します。

- ステートメント・レベルの最適化ガイドラインを作成できます。これは、オプティマイザーがいつでも指定のステートメントを見つけたときに、I_SUPPKEY 索引を使用して SUPPLIERS 表にアクセスするように要求します。

最適化プロファイルの中の、次に示す 2 つの主なセクションに、前述の 2 タイプのガイドラインを指定することができます。

- グローバル最適化ガイドライン・セクションには OPTGUIDELINES エレメントを 1 つ含めることができます。
- ステートメント・プロファイル・セクションには任意の数の STMTPROFILE エレメントを含めることができます。

また、最適化プロファイルには OPTPROFILE エレメントも含まれています。これには、メタデータおよび処理ディレクティブが含まれます。

以下のコードは、有効な最適化プロファイルの一例です。この最適化プロファイルには、1 つのグローバル最適化ガイドライン・セクションと、1 つの STMTPROFILE エレメントが含まれるステートメント・プロファイル・セクションがあります。

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE>

  <!--
    Global optimization guidelines section.
    Optional but at most one.
  -->
  <OPTGUIDELINES>
    <MQT NAME="Test.AvgSales"/>
    <MQT NAME="Test.SumSales"/>
  </OPTGUIDELINES>

  <!--
    Statement profile section.
    Zero or more.
  -->
  <STMTPROFILE ID="Guidelines for TPCD Q9">
    <STMTKEY SCHEMA="TPCD">
      <![CDATA[SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE,
S.S_COMMENT FROM PARTS P, SUPPLIERS S, PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY
AND P.P_SIZE = 39 AND P.P_TYPE = 'BRASS'
AND S.S_NATION = 'MOROCCO' AND S.S_NATION IN ('MOROCCO', 'SPAIN')
AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
FROM PARTSUPP PS1, SUPPLIERS S1
WHERE P.P_PARTKEY = PS1.PS_PARTKEY AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
AND S1.S_NATION = S.S_NATION)]]>
    </STMTKEY>
    <OPTGUIDELINES>
      <IXSCAN TABID="Q1" INDEX="I_SUPPKEY"/>
    </OPTGUIDELINES>
  </STMTPROFILE>
</OPTPROFILE>
```

OPTPROFILE エレメント

最適化プロファイルは OPTPROFILE エレメントで始まります。前述の例では、このエレメントは、最適化プロファイルのバージョンが 9.1 であることを指定する VERSION 属性で構成されています。

グローバル最適化ガイドライン・セクション

グローバル最適化ガイドラインは、最適化プロファイルが有効にされたすべてのステートメントに適用されます。グローバル最適化ガイドライン・セクションは、グローバル OPTGUIDELINES エレメントで定義されます。前述の例では、このセクションには単一のグローバル最適化ガイドラインが含まれています。このガイドラインでは、最適化プロファイルが有効にされているどのステートメントを処理する際にも、MQT Test.AvgSales および Test.SumSales を考慮するようオプティマイザーに要求します。

ステートメント・プロファイル・セクション

ステートメント・プロファイルは、特定のステートメントに適用される最適化ガイドラインを定義します。最適化プロファイルにはゼロ個以上のステートメント・プロファイルを入れることができます。ステートメント・プロファイル・セクションは、STMTPROFILE エレメントで定義されます。前述の例では、このセクションには、最適化プロファイルが有効になっている特定のステートメントのガイドラインが含まれます。

各ステートメント・プロファイルには、ステートメント・キーおよびステートメント・レベルの最適化ガイドラインが含まれます。これらは、それぞれ STMTKEY および OPTGUIDELINES エレメントによって表されます。

- ステートメント・キーは、ステートメント・レベルの最適化ガイドラインが適用されるステートメントを識別します。この例では、STMTKEY エレメントにはオリジナル・ステートメント・テキストと、ステートメントを明白に識別するために必要なその他の情報が含まれています。オプティマイザーは、ステートメント・キーを使用して、ステートメント・プロファイルと該当するステートメントを突き合わせます。この関係により、アプリケーションを変更しなくても、ステートメントの最適化ガイドラインを提供することができます。
- ステートメント・プロファイルのステートメント・レベルの最適化ガイドライン・セクションは、OPTGUIDELINES エレメントによって表されます。このセクションは、1 つ以上のアクセス要求および結合要求で構成されます。これらの要求は、ステートメント内の表のアクセスおよび結合のための方法を指定します。ステートメント・プロファイル内のステートメント・キーのマッチングが成功すると、オプティマイザーはステートメントの最適化時に関連したステートメント・レベルの最適化ガイドラインを参照します。この例には、1 つのアクセス要求が含まれています。これは、ネストされた副選択で参照される SUPPLIERS 表が I_SUPPKEY という名前の索引を使用することを指定します。

グローバル最適化ガイドラインとステートメント・プロファイル・セクションの両方に共通するエレメント

OPTGUIDELINES エレメント以外で、前述のセクションの両方で使用可能なエレメントとして、次に示す REGISTRY エレメントと STMTMATCH エレメントがあります。

- REGISTRY エレメントは、特定のレジストリー変数をステートメント・レベルまたはグローバル・レベルで設定できます。REGISTRY エレメントは、OPTGUIDELINES エレメントの中にネストします。

REGISTRY エlementの中には OPTION Elementが含まれます。OPTION Elementには NAME 属性と VALUE 属性があり、これらは指定されたレジストリー変数の値を設定するために使用されます。

レジストリー変数にグローバル・レベルで値を指定すると、その値はプロファイルが適用される接続のすべてのステートメントに適用されます。レジストリー変数にステートメント・レベルで値を指定すると、その値は STMTKEY の中のそのステートメントに対してのみ適用されます。この値は、ステートメント・レベルのほうがグローバル・レベルよりも優先されます。

- STMTMATCH Elementは、コンパイル・ステートメントが最適化プロファイルのステートメントと一致したときに使用されるマッチングのタイプを設定します。

STMTMATCH Elementには EXACT 属性があり、これを TRUE か FALSE のいずれかに設定できます。STMTMATCH EXACT のデフォルト値は TRUE です。

STMTMATCH EXACT が TRUE に設定されていると、完全一致突き合わせが適用されます。STMTMATCH EXACT が FALSE に設定されていると、不完全一致突き合わせが適用されます。

最適化プロファイルの作成:

最適化プロファイルとは、1 つ以上のデータ操作言語 (DML) ステートメントについての最適化ガイドラインを記述した XML 文書です。

このタスクについて

最適化プロファイルにはさまざまな組み合わせでガイドラインを含めることができるため、以下の情報では、どの最適化プロファイルの作成にも共通するステップのみを詳述します。

手順

最適化プロファイルを作成するには、以下のようにします。

1. XML エディターを起動します。可能なら、スキーマ妥当性検査機能を持つものを使用してください。オプティマイザーは XML 妥当性検査を実行しません。最適化プロファイルは、現行の最適化プロファイル・スキーマに従って有効にする必要があります。
2. 意味のある名前を使用して XML 文書を作成します。適用先のステートメントの有効範囲を説明する名前を指定することができます。例えば、inventory_db.xml と指定します。
3. XML 宣言を文書に追加します。エンコード形式を指定しない場合、UTF-8 が想定されます。可能であれば、UTF-16 エンコード方式で文書を保管してください。このエンコード方式で処理すると、データ・サーバーはより効率的に機能します。

```
<?xml version="1.0" encoding="UTF-16"?>
```

4. 最適化プロファイル・セクションを文書に追加します。

```
<OPTPROFILE>  
</OPTPROFILE>
```

5. OPTPROFILE エlement内に、適宜、グローバルまたはステートメント・レベルの最適化ガイドラインを作成し、ファイルを保存します。

次のタスク

XML 文書を作成した後で、最適化プロファイルを SYSTOOLS.OPT_PROFILE 表に挿入し、この最適化プロファイルを使用するようにデータ・サーバーを構成します。

最適化プロファイル内の SQL コンパイラー・レジストリー変数:

最適化プロファイルには、さまざまなレジストリー変数を持たせることができます。これらは、アプリケーションで使用される特定の照会ステートメントまたは多くの照会ステートメントに適用できます。

最適化プロファイルにレジストリー変数を設定すると、さまざまなアプリケーションにさまざまな照会ステートメントを使用する際の柔軟性を向上させることができます。 **db2set** コマンドを使用してレジストリー変数を設定すると、レジストリー変数の値がインスタンス全体に適用されます。最適化プロファイルでは、レジストリー変数の値は最適化プロファイルで指定されたステートメントにしか適用されません。レジストリー変数を最適化プロファイルの中で設定することにより、他の照会ステートメントのレジストリー変数の設定値を気にせずに、特定のステートメントをアプリケーションに合わせて調整することができます。

最適化プロファイルには、一部のレジストリー変数しか設定できません。以下のレジストリー変数は、最適化プロファイルの中に設定できます。

- **DB2_ANTIJOIN**
- **DB2_EXTENDED_OPTIMIZATION** (値は、ON、OFF、および IXOR のみサポートされません)
- **DB2_INLIST_TO_NLJN**
- **DB2_MINIMIZE_LISTPREFETCH**
- **DB2_NEW_CORR_SQ_FF**
- **DB2_OPT_MAX_TEMP_SIZE**
- **DB2_REDUCED_OPTIMIZATION**
- **DB2_RESOLVE_CALL_CONFLICT**
- **DB2_SELECTIVITY**
- **DB2_SELUDI_COMM_BUFFER**
- **DB2_SORT_AFTER_TQ**

レジストリー変数は、グローバル・レベルでもステートメント・レベルでも設定できます。グローバル・レベルで設定されているレジストリー変数は、レジストリー変数の設定値を最適化プロファイル内のすべてのステートメントに対して使用します。ステートメント・レベルで設定されているレジストリー変数は、レジストリー変数の設定値を特定のステートメントにのみ適用します。同一のレジストリー変数がグローバル・レベルとステートメント・レベルの両方で設定されている場合、ステートメント・レベルのレジストリー変数値が優先になります。

レジストリー変数を設定するための構文

それぞれのレジストリー変数は、NAME 属性と VALUE 属性を使用して OPTION XML エlementで定義され、設定されます。これらはすべて REGISTRY Elementの中にネストします。以下に例を示します。

```
<REGISTRY>
  <OPTION NAME='DB2_SELECTIVITY' VALUE='YES'/>
  <OPTION NAME='DB2_REDUCED_OPTIMIZATION' VALUE='NO'/>
</REGISTRY>
```

このプロファイルを使用するアプリケーションの全ステートメントに OPTION Elementを適用するには、REGISTRY Elementと OPTION Elementをグローバル OPTGUIDELINES Elementに含めます。

特定の SQL ステートメントだけに OPTION Elementを適用するには、REGISTRY Elementと OPTION Elementを、該当するステートメント・レベルの STMTPROFILE Elementに含めます。STMTPROFILE Elementが異なれば、OPTION Elementの設定値も別の値にすることができます。

次の例は、アプリケーション・レベルとステートメント・レベルでのレジストリー変数の設定値を示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE>

  <!--Global section -->
  <OPTGUIDELINES>
    <!-- New: registry variables that apply to all SQL statements that
         use this profile -->
    <REGISTRY>
      <OPTION NAME='DB2_SELECTIVITY' VALUE='YES'/>
      <OPTION NAME='DB2_REDUCED_OPTIMIZATION' VALUE='NO'/>
    </REGISTRY>
  </OPTGUIDELINES>

  <!-- First statement level profile -->
  <STMTPROFILE ID='S1'>
    <STMTKEY>
      <![CDATA[select t1.c1, count(*) from t1,t2 where t1.c1 = t2.c1
                group by t1.c1]]>
    </STMTKEY>
    <OPTGUIDELINES>
      <!-- New: registry variables that JUST applies to the above
           SQL statement when using this profile -->
      <REGISTRY>
        <OPTION NAME='DB2_REDUCED_OPTIMIZATION' VALUE='NO_SORT_NLJOIN'/>
      </REGISTRY>
      <NLJOIN>
        <TBSCAN TABLE='T1'/>
        <TBSCAN TABLE='T2'/>
      </NLJOIN>
    </OPTGUIDELINES>
  </STMTPROFILE>

  <!-- Second statement level profile -->
  <STMTPROFILE ID='S2'>
    <STMTKEY><![CDATA[select * from T1 where c1 in( 10,20)]]</STMTKEY>
    <OPTGUIDELINES>
      <!-- New: registry variables that JUST applies to the above
           SQL statement when using this profile -->
      <REGISTRY>
        <OPTION NAME='DB2_REDUCED_OPTIMIZATION' VALUE='YES'/>
      </REGISTRY>
    </OPTGUIDELINES>
  </STMTPROFILE>
</OPTPROFILE>
```

```
</REGISTRY>
</OPTGUIDELINES>
</STMTPROFILE>

</OPTPROFILE>
```

優先順位

前述の例では、同じレジストリー変数を複数の箇所で設定しています。これらの設定の他に、**db2set** コマンドも、レジストリー変数の設定に使用できます。レジストリー変数は、以下の優先順位 (高いものから低いものへ) で適用されます。

1. ステートメント・レベルの最適化ガイドラインで定義された、ステートメント・レベルの最適化プロファイル設定値。
2. グローバル最適化ガイドラインで定義された、全体的な最適化プロファイル設定値。
3. **db2set** コマンドで設定されたレジストリー変数。

Explain 機能

Explain 機能は、データ操作言語 (DML) のコンパイルに影響を及ぼすすべての SQL コンパイラー・レジストリー変数に関する情報をキャプチャーし、この情報を Explain 表に書き込みます。

Explain 機能を使用して、特定のステートメントにどのレジストリー変数が使用されているかを判別できます。Explain 機能を活動化して、照会を実行します。ENVVAR セクションの 2 つのストリングは、設定元を示しています。

- [Global Optimization Guideline]
- [Statement-level Optimization Guideline]

以下に例を示します。

```
ENVVAR : (Environment Variable)
        DB2_EXTENDED_OPTIMIZATION = ON
```

```
ENVVAR : (Environment Variable)
        DB2_EXTENDED_OPTIMIZATION = ON [Global Optimization Guideline]
```

```
ENVVAR : (Environment Variable)
        DB2_EXTENDED_OPTIMIZATION = ON [Statement-level Optimization Guideline]
```

レジストリー変数が別々の場所で設定されている場合、優先順位が最も高いレジストリー変数が、Explain 出力に表示される唯一のレジストリー変数になります。

最適化プロファイルを使用するようにデータ・サーバーを構成する:

最適化プロファイルが作成され、その内容が現行最適化プロファイル・スキーマ (COPS) に対して妥当性検査された後、その内容を固有のスキーマ修飾名に関連付けて、SYSTOOLS.OPT_PROFILE 表に保管する必要があります。

手順

最適化プロファイルを使用するようにデータ・サーバーを構成するには、以下のようになります。

1. 最適化プロファイル表を作成します。

450 ページの『SYSTOOLS.OPT_PROFILE 表』を参照してください。表の各行には最適化プロファイルが 1 つ含まれることができます。SCHEMA 列および NAME 列によって最適化プロファイルが区別され、PROFILE 列には最適化プロファイルのテキストが収められます。

2. オプション: データベース・セキュリティー要件を満たすよう、表に対する任意の権限または特権を付与することができます。これは、オブティマイザーによる表の読み取り能力に影響を及ぼしません。
3. 使用する最適化プロファイルを表に挿入します。

最適化プロファイル内の STMTKEY フィールド:

STMTPROFILE では、ターゲットとなるステートメントは、STMTKEY サブエレメントによって識別されます。STMTKEY フィールドに定義されるステートメントは、アプリケーションによって実行されるステートメントと厳密に一致している必要があります。こうして、DB2 によってターゲットのステートメントを明確に識別することが可能となります。ただし、このステートメント内で「空白」は許容されます。

DB2 で、現在のコンパイル・キーに一致するステートメント・キーが検出されると、検索は停止されます。したがって、ステートメント・キーが現在のコンパイル・キーに一致する最適化プロファイル内に複数のステートメント・プロファイルがあった場合、そのようなステートメント・プロファイルのうちの最初のもの (文書の順序に基づいて) だけが使用されます。しかも、このような場合、エラーや警告は出されません。

ステートメント・キーは、コンパイル・キーのデフォルト・スキーマが「COLLEGE」で、関数パスが SYSIBM,SYSFUN,SYSPROC,DAVE であれば、ステートメント "select * from orders where foo(orderkey)>20" と一致します。

```
<STMTKEY SCHEMA='COLLEGE' FUNCPATH='SYSIBM,SYSFUN,SYSPROC,DAVE'>  
<![CDATA[select * from orders where foo(orderkey)>20]]>  
</stmtkey>
```

オブティマイザーが使用する最適化プロファイルの指定:

OPTPROFILE バインド・オプションを使用して最適化プロファイルがパッケージ・レベルで使用されることを指定するか、**CURRENT OPTIMIZATION PROFILE** 特殊レジスターを使用して最適化プロファイルがステートメント・レベルで使用されることを指定します。

この特殊レジスターには、最適化のために動的に準備されたステートメントによって使用される最適化プロファイルの修飾名が含まれます。CLI アプリケーションの場合、**CURRENTOPTIMIZATIONPROFILE** クライアント構成オプションを使用して、各接続にこの特殊レジスターを設定することができます。

また、**OPTPROFILE** バインド・オプションの設定は、**CURRENT OPTIMIZATION PROFILE** 特殊レジスターにデフォルトの最適化プロファイルを指定します。デフォルトの優先順位の順序は、次のとおりです。

- **OPTPROFILE** バインド・オプションは、その他の設定に関係なく、すべての静的ステートメントに適用されます。

- 動的ステートメントの場合、CURRENT OPTIMIZATION PROFILE 特殊レジスタの値は、優先順位の低い順から高い順に、以下によって決定されます。
 - **OPTPROFILE** バインド・オプション
 - **CURRENTOPTIMIZATIONPROFILE** クライアント構成オプション
 - アプリケーション内の最新の SET CURRENT OPTIMIZATION PROFILE ステートメント

アプリケーション内での最適化プロファイルの設定:

SET CURRENT OPTIMIZATION PROFILE ステートメントを使用することにより、アプリケーションにおいて、動的ステートメントの現行最適化プロファイルの設定を制御できます。

このタスクについて

ステートメントで指定する最適化プロファイル名はスキーマ修飾名でなければなりません。スキーマ名を指定しない場合、CURRENT SCHEMA 特殊レジスタの値が暗黙スキーマ修飾子として使用されます。

指定される最適化プロファイルは、別の SET CURRENT OPTIMIZATION PROFILE ステートメントが検出されるまで、後続のすべての動的ステートメントに適用されます。静的ステートメントは、この設定が評価される前に前処理されてパッケージされるため、影響を受けません。

手順

アプリケーション内で最適化プロファイルを設定するには、以下のようになります。

- アプリケーション内の任意の箇所で、SET CURRENT OPTIMIZATION PROFILE ステートメントを使用します。例えば、次のシーケンス内の最終ステートメントは、JON.SALES 最適化プロファイルにしたがって最適化されます。

```
EXEC SQL SET CURRENT OPTIMIZATION PROFILE = 'NEWTON.INVENTDB';

/* The following statements are both optimized with 'NEWTON.INVENTDB' */
EXEC SQL PREPARE stmt FROM SELECT ... ;
EXEC SQL EXECUTE stmt;

EXEC SQL EXECUTE IMMEDIATE SELECT ... ;

EXEC SQL SET CURRENT OPTIMIZATION PROFILE = 'JON.SALES';

/* This statement is optimized with 'JON.SALES' */
EXEC SQL EXECUTE IMMEDIATE SELECT ... ;
```

- アプリケーションの開始時に有効だったデフォルト最適化プロファイルをオプティマイザーが使用するようになる場合、NULL 値を指定します。例えば、

```
EXEC SQL SET CURRENT OPTIMIZATION PROFILE = NULL;
```

- オプティマイザーが最適化プロファイルを使用しないようになる場合、空ストリングを指定します。例えば、

```
EXEC SQL SET CURRENT OPTIMIZATION PROFILE = '';
```

- コール・レベル・インターフェース (CLI) アプリケーションを使用している場合は、**UPDATE CLI CONFIGURATION** コマンドを使用して、CURRENTOPTIMIZATIONPROFILE パラメーターを db2cli.ini ファイルに追加できます。例えば、

```
update cli cfg for section sanfran using currentoptimizationprofile jon.sales
```

この結果として、db2cli.ini ファイルに次の項目が生成されます。

```
[SANFRAN]
CURRENTOPTIMIZATIONPROFILE=JON.SALES
```

注: アプリケーション内の SET CURRENT OPTIMIZATION PROFILE ステートメントは、この設定をオーバーライドします。

最適化プロファイルのパッケージへのバインディング:

BIND コマンドまたは **PRECOMPILE** コマンドを使用してパッケージを準備する際、**OPTPROFILE** コマンド・パラメーターを使用すると、そのパッケージ用の最適化プロファイルを指定できます。

このタスクについて

この方法は最適化プロファイルを静的ステートメントに適用するための唯一の方法で、指定されたプロファイルはパッケージ内のすべての静的ステートメントに適用されます。また、この方法で指定される最適化プロファイルは、パッケージ内の動的ステートメントに使用されるデフォルト最適化プロファイルです。

手順

API (例えば sqlprep) またはコマンド行プロセッサ (CLP) を使用して、SQLJ または組み込み SQL で最適化プロファイルをバインドできます。

例として、以下のコードに、CLP からインベントリー・データベース最適化プロファイルをインベントリー・アプリケーションにバインドする方法を示します。

```
db2 prep inventapp.sqc bindfile optprofile newton.inventdb
db2 bind inventapp.bnd
db2 connect reset
db2 terminate
xlc -I$HOME/sqllib/include -c inventapp.c -o inventapp.o
xlc -o inventapp inventapp.o -ldb2 -L$HOME/sqllib/lib
```

最適化プロファイルにスキーマ名を指定しない場合、暗黙の修飾子として **QUALIFIER** コマンド・パラメーターが使用されます。

最適化プロファイルの変更:

最適化プロファイルを変更すると、変更を有効にするために、一定の手順を踏む必要があります。

このタスクについて

最適化プロファイルが参照されると、それはコンパイルされて、メモリー内にキャッシュされます。したがって、これらの参照も除去する必要があります。 **FLUSH OPTIMIZATION PROFILE CACHE** ステートメントは、最適化プロファイル・キャッシュから古いプロファイルを除去します。また、このステートメントは、古いプロファイルを使用して準備された動的プラン・キャッシュ内のステートメントをすべて無効にします (論理的な無効化)。

手順

最適化プロファイルを変更するには、以下のようになります。

1. ディスクにある最適化プロファイルの XML ファイルを編集し、必要な変更を行って、ファイルをディスクに保存します。
2. ファイルに加えられた変更が、DB2OptProfile.xsd ファイルの中にある現行の最適化プロファイル・スキーマ (COPS) に定義されている正しい形式の XML になっているかを妥当性検査します。この DB2OptProfile.xsd ファイルは sqllib ディレクトリーの misc サブディレクトリーにあります。
3. SYSTOOLS.OPT_PROFILE 表の既存の行を新しいプロファイルで更新します。
4. 最適化プロファイル・キャッシュをフラッシュするためのトリガーを作成しなかった場合は、FLUSH OPTIMIZATION PROFILE CACHE ステートメントを発行します。このステートメントは、最適化プロファイル・キャッシュに含まれている可能性のある最適化プロファイルのバージョンをすべて除去します。

最適化プロファイル・キャッシュをフラッシュすると、古い最適化プロファイルで準備された動的ステートメントも動的プラン・キャッシュで無効にされます。

タスクの結果

その後、最適化プロファイルを参照すると、オプティマイザーは新しいプロファイルを読み取り、それを最適化プロファイル・キャッシュに再ロードします。古い最適化プロファイルのもとで準備されたステートメントは論理的に無効化されます。それらのステートメントに対する呼び出しは新規の最適化プロファイルのもとで準備され、動的プラン・キャッシュに再キャッシュされます。

最適化プロファイルの削除:

必要ではない最適化プロファイルは、SYSTOOLS.OPT_PROFILE 表から削除することによって除去できます。最適化プロファイルが参照されると、それはコンパイルされて、メモリー内にキャッシュされます。そのため、オリジナル・プロファイルがすでに使用されている場合、最適化プロファイル・キャッシュからも削除された最適化プロファイルをフラッシュする必要があります。

手順

最適化プロファイルを削除するには、以下のようにします。

1. SYSTOOLS.OPT_PROFILE 表から最適化プロファイルを削除します。例えば、

```
delete from systools.opt_profile
  where schema = 'NEWTON' and name = 'INVENTDB'
```
2. 最適化プロファイル・キャッシュをフラッシュするためのトリガーを作成しなかった場合、FLUSH OPTIMIZATION PROFILE CACHE ステートメントを発行して、最適化プロファイル・キャッシュに含まれている可能性のある最適化プロファイルのバージョンをすべて削除します。450 ページの『最適化プロファイル・キャッシュをフラッシュするためのトリガー』を参照してください。

注: 最適化プロファイル・キャッシュをフラッシュすると、古い最適化プロファイルで準備された動的ステートメントも動的プラン・キャッシュで無効にされず。

タスクの結果

その後、その最適化プロファイルを参照すると、オプティマイザーは理由コード 13 の SQL0437W を返します。

最適化ガイドライン:

最適化ガイドラインのタイプ:

DB2 オプティマイザーは、2 つのフェーズでステートメントを処理します。1 つは照会書き直し最適化フェーズ、もう 1 つはプラン最適化フェーズです。

最適化ステートメントは、照会書き直し最適化フェーズによって決定されます。このフェーズでは、元のステートメントが、プラン最適化フェーズでより容易に最適化できる、意味的に同等のステートメントにトランスフォームされます。プラン最適化フェーズでは、数多くの選択肢を列挙して実行コストの見積もりが最小になるものを選択することによって、最適化ステートメントに最適なアクセス方式、結合方式、および結合順序が判別されます。

2 つの最適化フェーズで考慮される照会のトランスフォーメーション、アクセス方式、結合方式、結合順序、および他の最適化のための選択肢は、CURRENT QUERY OPTIMIZATION 特殊レジスター、REOPT バインド・オプション、および DB2_REDUCED_OPTIMIZATION レジストリー変数などの様々な DB2 パラメーターによって制御されます。こうした一群の最適化のための選択肢は、検索スペースと呼ばれます。

以下のタイプのステートメント最適化ガイドラインがサポートされています。

- 一般最適化ガイドライン を使用すると、一般の最適化パラメーターの設定に作用することができ、検索スペースに影響を与える可能性があるため、最初に適用されます。
- 照会書き直しガイドライン を使用すると、照会書き直し最適化フェーズで考慮されるトランスフォーメーションに作用することができ、プラン最適化フェーズで最適化されるステートメントに影響を与える可能性があるため、次に適用されます。
- プラン最適化ガイドライン を使用すると、プラン最適化フェーズで考慮されるアクセス方式、結合方式、および結合順序に作用することができ、最後に適用されます。

一般最適化ガイドライン:

一般最適化ガイドラインを使用すると、一般の最適化パラメーターを設定できます。

これらのガイドラインは、それぞれステートメント・レベルの有効範囲を持ちます。

照会書き直し最適化ガイドライン:

照会書き直しガイドラインを使用すると、照会書き直し最適化フェーズで考慮されるトランスフォーメーションに作用することができます。このフェーズでは、元のステートメントを意味的に同等の最適化ステートメントに変換します。

プラン最適化フェーズで、最適化ステートメントの最適な実行プランが決定されます。このため、照会書き直し最適化ガイドラインは、プラン最適化ガイドラインの適用可能性に影響を与える可能性があります。

各照会書き直し最適化ガイドラインは、オプティマイザーの照会トランスフォーメーション規則のいずれかに対応します。以下の照会トランスフォーメーション規則は、照会書き直し最適化ガイドラインから影響を受ける可能性があります。

- IN-LIST から結合へ
- 副照会から結合へ
- NOT-EXISTS 副照会からアンチ結合へ
- NOT-IN 副照会からアンチ結合へ

照会書き直し最適化ガイドラインは、常に適用可能なわけではありません。照会書き直し規則は、一度に 1 つ施行されます。つまり、後続の規則の前に施行される照会書き直し規則は、その後続の規則に関連付けられている照会書き直し最適化ガイドラインに作用します。環境の構成がいくつかの書き直し規則の動作に影響を与え、それが照会書き直し最適化ガイドラインの特定の規則への適用可能性に影響を与えます。

毎回同じ結果を得るためには、照会書き直し規則を施行する前に特定の条件を適用しておく必要があります。照会書き直しコンポーネントが照会への規則の適用を試みる際に、その規則に関連付けられている条件が満たされていない場合、その規則では照会書き直し最適化ガイドラインが無視されます。照会書き直し最適化ガイドラインが適用できない場合に、そのガイドラインが使用可能化のガイドラインである場合は、理由コード 13 で SQL0437W が戻されます。照会書き直し最適化ガイドラインが適用できない場合に、ガイドラインが使用不可能化のガイドラインである場合は、メッセージは戻されません。この場合、規則は使用不可にされていたものとして扱われるため、照会書き直し規則は適用されません。

照会書き直し最適化ガイドラインは、ステートメント・レベル・ガイドラインと述部レベル・ガイドラインの 2 つのカテゴリに分けることができます。ステートメント・レベルのカテゴリは、すべての照会書き直し最適化ガイドラインでサポートされています。述部レベルのカテゴリは、INLIST2JOIN でのみサポートされています。ステートメント・レベルの照会書き直し最適化ガイドラインは、照会全体に適用されます。述部レベルの照会書き直し最適化ガイドラインは、特定の述部のみ適用されます。ステートメント・レベルと述部レベルの両方の照会書き直し最適化ガイドラインが指定された場合は、述部レベルのガイドラインが、その特定の述部に関してステートメント・レベルのガイドラインをオーバーライドします。

各照会書き直し最適化ガイドラインは、最適化ガイドライン・スキーマ内で、対応する書き直し要求エレメントによって表されます。

以下の例は、INLIST2JOIN 書き直し要求エレメントによって表されているように、IN-LIST から結合への照会書き直し最適化ガイドラインを示しています。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY
      AND S.S_SUPPKEY = PS.PS_SUPPKEY
      AND P_SIZE IN (35, 36, 39, 40)
      AND S.S_NATION IN ('INDIA', 'SPAIN')
```

```

AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                        FROM "Tpcd".PARTSUPP PS1, "Tpcd".SUPPLIERS S1
                        WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                        AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                        AND S1.S_NATION = S.S_NATION)
ORDER BY S.S_NAME
<OPTGUIDELINES><INLIST2JOIN TABLE='P' /></OPTGUIDELINES>;

```

この特定の照会書き直し最適化ガイドラインは、述部 P_SIZE IN (35, 36, 39, 40) 内の定数のリストが表式に変換されるように指定します。この表式は、メインの副選択内の PARTS 表への、索引付けされたネスト・ループ結合のアクセスを駆動するのに適格になります。TABLE 属性は、この述部を適用する表参照を示して、ターゲット IN-LIST 述部を識別するために、使用されます。識別された表参照の IN-LIST 述部が複数ある場合、INLIST2JOIN 書き直し要求エレメントは未確定と見なされ、無視されます。

そのような場合、COLUMN 属性を追加して、ターゲット IN-LIST 述部をさらに修飾することができます。例えば、

```

SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P.P_PARTKEY = PS.PS_PARTKEY
AND S.S_SUPPKEY = PS.PS_SUPPKEY
AND P_SIZE IN (35, 36, 39, 40)
AND P_TYPE IN ('BRASS', 'COPPER')
AND S.S_NATION IN ('INDIA', 'SPAIN')
AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                        FROM "Tpcd".PARTSUPP PS1, "Tpcd".SUPPLIERS S1
                        WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                        AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                        AND S1.S_NATION = S.S_NATION)
ORDER BY S.S_NAME
<OPTGUIDELINES><INLIST2JOIN TABLE='P' COLUMN='P_SIZE' /></OPTGUIDELINES>;

```

INLIST2JOIN エレメントの TABLE 属性は、メインの副選択内の PARTS 表参照を識別します。COLUMN 属性を使用して、P_SIZE 列の IN-LIST 述部をターゲットとして識別します。一般的に、COLUMN 属性の値には、ターゲット IN-LIST 述部で参照される列の非修飾名を含めることができます。COLUMN 属性が TABLE 属性なしで提供される場合、照会書き直し最適化ガイドラインは無効と見なされ、無視されます。

OPTION 属性を使用すると、特定の照会書き直し最適化ガイドラインを有効または無効にできます。以下の例では、OPTION 属性が DISABLE に設定されているので、述部 P_SIZE IN (35, 36, 39, 40) 内の定数のリストは、表式に変換されません。OPTION 属性のデフォルト値は ENABLE です。ENABLE および DISABLE は、大文字で指定しなければなりません。

```

<OPTGUIDELINES>
  <INLIST2JOIN TABLE='P' COLUMN='P_SIZE' OPTION='DISABLE' />
</OPTGUIDELINES>

```

以下の例では、INLIST2JOIN 書き直し要求エレメントに TABLE 属性がありません。オプティマイザーはこれを、ステートメントのすべての IN-LIST 述部に対して IN-LIST から結合への照会トランスフォーメーションを使用不可にする要求と解釈します。

```

<OPTGUIDELINES><INLIST2JOIN OPTION='DISABLE' /></OPTGUIDELINES>

```

以下の例は、SUBQ2JOIN 書き直し要求エレメントによって表されているように、副照会から結合への照会書き直し最適化ガイドラインを示しています。副照会から結合へのトランスフォーメーションは、副照会を同等の表式に変換します。トランスフォーメーションは、EXISTS、IN、=SOME、=ANY、<>SOME、または <>ANY によって定量化される副照会の述部に適用されます。副照会から結合への照会書き直し最適化ガイドラインによって、副照会のマージが確実に行われるとは限りません。特定の副照会は、この照会書き直し最適化ガイドラインのターゲットにすることができません。トランスフォーメーションは、ステートメント・レベルで有効または無効にすることしかできません。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P.PARTKEY = PS.PS_PARTKEY
      AND S.S_SUPPKEY = PS.PS_SUPPKEY
      AND P.SIZE IN (35, 36, 39, 40)
      AND P.TYPE = 'BRASS'
      AND S.S_NATION IN ('INDIA', 'SPAIN')
      AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                              FROM "Tpcd".PARTSUPP PS1, "Tpcd".SUPPLIERS S1
                              WHERE P.PARTKEY = PS1.PS_PARTKEY
                                   AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                                   AND S1.S_NATION = S.S_NATION)

ORDER BY S.S_NAME
<OPTGUIDELINES><SUBQ2JOIN OPTION='DISABLE' /></OPTGUIDELINES>;
```

以下の例は、NOTEX2AJ 書き直し要求エレメントによって表されているように、NOT-EXISTS からアンチ結合への照会書き直し最適化ガイドラインを示しています。NOT-EXISTS からアンチ結合へのトランスフォーメーションは、副照会を、アンチ結合セマンティクスを使用して他の表と結合される表式に変換します (一致しない行のみが戻されます)。NOT-EXISTS からアンチ結合への照会書き直し最適化ガイドラインは、NOT EXISTS によって定量化された副照会の述部に適用されます。NOT-EXISTS からアンチ結合への照会書き直し最適化ガイドラインによって、副照会のマージが確実に行われるとは限りません。特定の副照会は、この照会書き直し最適化ガイドラインのターゲットにすることができません。トランスフォーメーションは、ステートメント・レベルで有効または無効にすることしかできません。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P.PARTKEY = PS.PS_PARTKEY
      AND S.S_SUPPKEY = PS.PS_SUPPKEY
      AND P.SIZE IN (35, 36, 39, 40)
      AND P.TYPE = 'BRASS'
      AND S.S_NATION IN ('INDIA', 'SPAIN')
      AND NOT EXISTS (SELECT 1
                     FROM "Tpcd".SUPPLIERS S1
                     WHERE S1.S_SUPPKEY = PS.PS_SUPPKEY)

ORDER BY S.S_NAME
<OPTGUIDELINES><NOTEX2AJ OPTION='ENABLE' /></OPTGUIDELINES>;
```

注: ステートメント・レベルで照会トランスフォーメーション規則を有効にしても、ステートメントの特定の部分に規則が確実に適用されるとは限りません。照会トランスフォーメーションが行われるかどうかは、通常の基準を使用して決定されます。例えば、照会ブロックに複数の NOT EXISTS 述部がある場合、オプティマイザーはそれらのいずれかをアンチ結合に変換することを考慮しません。ステートメント・レベルで照会トランスフォーメーションを明示的に有効にしても、この動作は変わりません。

以下の例は、NOTIN2AJ 書き直し要求エレメントによって表されているように、NOT-IN からアンチ結合への照会書き直し最適化ガイドラインを示しています。NOT-IN からアンチ結合へのトランスフォーメーションは、副照会を、アンチ結合セマンティクスを使用して他の表と結合される表式に変換します（一致しない行のみが戻されます）。NOT-IN からアンチ結合への照会書き直し最適化ガイドラインは、NOT IN によって定量化された副照会の述部に適用されます。NOT-IN からアンチ結合への照会書き直し最適化ガイドラインによって、副照会のマージが確実に行われるとは限りません。特定の副照会は、この照会書き直し最適化ガイドラインのターゲットにすることができません。トランスフォーメーションは、ステートメント・レベルで有効または無効にすることしかできません。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY
      AND S.S_SUPPKEY = PS.PS_SUPPKEY
      AND P_SIZE IN (35, 36, 39, 40)
      AND P_TYPE = 'BRASS'
      AND S.S_NATION IN ('INDIA', 'SPAIN')
      AND PS.PS_SUPPKEY NOT IN (SELECT S1.S_SUPPKEY
                                FROM "Tpcd".SUPPLIERS S1
                                WHERE S1.S_NATION = 'CANADA')
ORDER BY S.S_NAME
<OPTGUIDELINES><NOTIN2AJ OPTION='ENABLE'/></OPTGUIDELINES>
```

特定の照会書き直し最適化ガイドラインは、ステートメントに適用されている他の照会書き直しトランスフォーメーションのコンテキスト内で考慮されるとき、適用不能になる場合があります。つまり、変換を有効にするようにとのガイドラインの要求が適用できない場合は、警告が戻されます。例えば、別の照会トランスフォーメーションによって照会から除去される述部をターゲットにしている INLIST2JOIN 書き直しの有効化要求エレメントは、適用不能になります。さらに、ある照会書き直し最適化ガイドラインが正常に適用されることで、他の照会書き直しトランスフォーメーション規則の適用可能性が変わる場合があります。例えば、オプティマイザは IN-LIST から結合へのトランスフォーメーションを照会ブロックごとに 1 つだけ適用するため、IN-LIST を表式に変換する 1 つの要求のせいで、別の IN-LIST が表式に変換されなくなる場合があります。

プラン最適化ガイドライン:

プラン最適化ガイドラインは、ステートメントのアクセス方式、結合方式、結合順序、およびその他の実行プランの詳細が決定されるコスト・ベースの最適化のフェーズで適用されます。

プラン最適化ガイドラインでは、実行プランのすべての面を指定する必要はありません。実行プランのうち、指定されなかった面については、コスト・ベースの方法でオプティマイザによって決定されます。

プラン最適化ガイドラインには、2 つのカテゴリーがあります。

- **accessRequest** - アクセス要求は、ステートメントの表参照を満たすアクセス方式を指定します。
- **joinRequest** - 結合要求では、結合操作を実行する方式および順序を指定します。結合要求は、アクセス要求またはその他の結合要求で構成されます。

アクセス要求の最適化ガイドラインは、オブティマイザーのデータ・アクセス方式 (表スキャン、索引スキャン、リスト・プリフェッチなど) に対応します。結合要求の最適化ガイドラインは、オブティマイザーの結合方式 (ネスト・ループ結合、ハッシュ結合、マージ結合など) に対応します。各アクセス要求および結合要求は、ステートメント最適化ガイドライン・スキーマの対応するアクセス要求エレメントおよび結合要求エレメントによって表されます。

以下の例は、IXSCAN アクセス要求エレメントによって表されているように、索引スキャン・アクセス要求を示しています。この特定の要求では、ステートメントのメインの副選択内での SUPPLIERS 表へのアクセスに、オブティマイザーが I_SUPPKEY 索引を使用するよう指定します。オプションの INDEX 属性は、必要な索引を識別します。TABLE 属性は、アクセス要求が適用される表参照を識別します。TABLE 属性は、直接的な名前 (この例では、相関名 S) を使用して、ターゲット表参照を識別する必要があります。

SQL statement:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
 where p_partkey = ps.ps_partkey and
       s.s_suppkey = ps.ps_suppkey and
       p_size = 39 and
       p_type = 'BRASS' and
       s.s_nation in ('MOROCCO', 'SPAIN') and
       ps.ps_supplycost = (select min(ps1.ps_supplycost)
                           from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                           where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                                s1.s_suppkey = ps1.ps_suppkey and
                                s1.s_nation = s.s_nation)

 order by s.s_name
```

Optimization guideline:

```
<OPTGUIDELINES>
  <IXSCAN TABLE='S' INDEX='I_SUPPKEY' />
</OPTGUIDELINES>
```

以下の索引スキャン・アクセス要求エレメントは、オブティマイザーがステートメントのメインの副選択で PARTS 表に対して索引アクセスを使用するよう指定します。INDEX 属性がないため、オブティマイザーはコスト・ベースの方法で索引を選択します。関連する相関名がないため、TABLE 属性は修飾表名を使用して、ターゲット表参照を参照します。

```
<OPTGUIDELINES>
  <IXSCAN TABLE='"Tpcd".PARTS' />
</OPTGUIDELINES>
```

以下のリスト・プリフェッチ・アクセス要求は、LPREFETCH アクセス要求エレメントによって表されています。この特定の要求では、ステートメントのネストされた副選択内での SUPPLIERS 表へのアクセスに、オブティマイザーが I_SNATION 索引を使用するよう指定します。TABLE 属性は、相関名 S1 を使用します。これは、ネストされた副選択内で SUPPLIERS 表参照を識別する直接的な名前だからです。

```
<OPTGUIDELINES>
  <LPREFETCH TABLE='S1' INDEX='I_SNATION' />
</OPTGUIDELINES>
```

以下の索引スキャン・アクセス要求エレメントでは、メインの副選択での SUPPLIERS 表へのアクセスに、 옵ティマイザーが I_SNAME 索引を使用するよう指定します。FIRST 属性は、この表が、対応する FROM 節用に選択された結合順序内で、最初にアクセスされる表になるように指定します。FIRST 属性は任意のアクセス要求または結合要求に追加できますが、同じ FROM 節の表を参照する FIRST 属性を持つアクセス要求または結合要求は最大 1 つしかありません。

SQL statement:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
 where p_partkey = ps.ps_partkey
       s.s_suppkey = ps.ps_suppkey and
       p_size = 39 and
       p_type = 'BRASS' and
       s.s_nation in ('MOROCCO', 'SPAIN') and
       ps.ps_supplycost = (select min(ps1.ps_supplycost)
                           from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                           where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                               s1.s_suppkey = ps1.ps_suppkey and
                               s1.s_nation = s.s_nation)

 order by s.s_name
 optimize for 1 row
```

Optimization guidelines:

```
<OPTGUIDELINES>
  <IXSCAN TABLE='S' INDEX='I_SNAME' FIRST='TRUE' />
</OPTGUIDELINES>
```

以下の例は、複数のアクセス要求を単一のステートメント最適化ガイドラインに受け渡す方法について示しています。TBSCAN アクセス要求エレメントは、表スキャン・アクセス要求を表します。この特定の要求は、ネストされた副選択内の SUPPLIERS 表が完全表スキャンを使用してアクセスされるように指定します。LPREFETCH アクセス要求エレメントは、 옵ティマイザーが、メインの副選択での SUPPLIERS 表へのリスト・プリフェッチ索引アクセス中に、 I_SUPPKEY 索引を使用するように指定します。

```
<OPTGUIDELINES>
  <TBSCAN TABLE='S1' />
  <LPREFETCH TABLE='S' INDEX='I_SUPPKEY' />
</OPTGUIDELINES>
```

以下の例は、NLJOIN 結合要求エレメントによって表されているように、ネスト・ループ結合要求を示しています。一般的に、結合要求エレメントには、2 つの子エレメントが含まれています。最初の子エレメントは、結合操作への必要な外部入力を表し、2 番目の子エレメントは、結合操作への必要な内部入力を表します。子エレメントは、アクセス要求、他の結合要求、またはアクセス要求と結合要求の組み合わせとすることができます。この例で、最初の IXSCAN アクセス要求エレメントは、メインの副選択内の PARTS 表が結合操作の外部表になるように指定します。また、PARTS 表のアクセスが索引スキャンを使用して実行されることも指定します。2 番目の IXSCAN アクセス要求エレメントは、メインの副選択内の PARTSUPP 表が結合操作の内部表になるように指定します。また、表が索引スキャンを使用してアクセスされることも指定します。

```

<OPTGUIDELINES>
  <NLJOIN>
    <IXSCAN TABLE='Tpcd'.Parts' />
    <IXSCAN TABLE="PS" />
  </NLJOIN>
</OPTGUIDELINES>

```

以下の例は、HSJOIN 結合要求エレメントによって表されているように、ハッシュ結合要求を示しています。ACCESS アクセス要求エレメントは、ネストされた副選択内の SUPPLIERS 表が結合操作の外部表になるように指定します。このアクセス要求エレメントは、結合順序を指定することが第一の目的である場合に役立ちます。IXSCAN アクセス要求エレメントは、ネストされた副選択内の PARTSUPP 表が結合操作の内部表になること、およびオプティマイザーがその表へのアクセスに索引スキャンを選択することを指定します。

```

<OPTGUIDELINES>
  <HSJOIN>
    <ACCESS TABLE='S1' />
    <IXSCAN TABLE='PS1' />
  </HSJOIN>
</OPTGUIDELINES>

```

以下の例は、結合要求のネスティングによって、より大きな結合要求を構成する方法を示しています。例には、MSJOIN 結合要求エレメントによって表されているように、マージ結合要求が含まれます。結合操作の外部入力、NLJOIN 結合要求エレメントによって表されているように、メインの副選択の PARTS 表と PARTSUPP 表の結合の結果です。結合要求エレメントの内部入力は、IXSCAN アクセス要求エレメントによって表されているように、メインの副選択内の SUPPLIERS 表です。

```

<OPTGUIDELINES>
  <MSJOIN>
    <NLJOIN>
      <IXSCAN TABLE='Tpcd'.Parts' />
      <IXSCAN TABLE="PS" />
    </NLJOIN>
    <IXSCAN TABLE='S' />
  </MSJOIN>
</OPTGUIDELINES>

```

結合要求が有効になる場合、直接的または間接的にその内部にネストされたすべてのアクセス要求エレメントは、最適化されるステートメントの同じ FROM 節の表を参照する必要があります。

MQT マッチングの最適化ガイドライン

ユーザーはオプティマイザーの決定をオーバーライドし、MQTENFORCE エレメントによって特定のマテリアライズ照会表 (MQT) を強制的に選択させることができます。MQTENFORCE エレメントはグローバルおよびステートメントのどちらのプロファイル・レベルにおいても指定でき、以下のいずれかの属性を指定して使用できます。

NAME

選択する MQT の名前の一部、または完全修飾名を指定します。

TYPE タイプごとに MQT のグループを指定します。可能な値は以下のとおりです。

- NORMAL: すべての非複製 MQT

- REPLICATED: すべての複製 MQT
- ALL: すべての MQT

以下は、すべての複製 MQT と TPCD.PARTSMQT を強制的に選択するガイドライン例です。

```
<OPTGUIDELINES>
  <MQTENFORCE NAME='TPCD.PARTSMQT' />
  <MQTENFORCE TYPE='REPLICATED' />
</OPTGUIDELINES>
```

注: 一度に複数の属性を指定すると、最初の属性のみが使用されます。したがって、次の例の場合、

```
<MQTENFORCE NAME='TPCD.PARTSMQT' TYPE='REPLICATED' />
```

PARTSMQT MQT だけが強制的に選択されます。

ステートメント・レベルの最適化ガイドラインの作成:

ステートメント・プロファイルにおけるステートメント・レベルの最適化ガイドラインのセクションは、1 つ以上のアクセス要求または結合要求で構成されます。これらの要求により、ステートメントにおける表へのアクセス方法または結合方法が指定されます。

始める前に

他のすべてのチューニング・オプションを使い尽くします。例えば、

1. データ分散統計が **RUNSTATS** ユーティリティーによって最近更新されたことを確認します。
2. ワークロードに適切な最適化クラスを設定して、データ・サーバーが実行されていることを確認します。
3. オプティマイザーに、照会で参照される表にアクセスするための適切な索引があることを確認します。

手順

ステートメント・レベルの最適化ガイドラインを作成するには、以下のようになります。

1. ステートメント・レベルのガイドラインを挿入する最適化プロファイルを作成します。384 ページの『最適化プロファイルの作成』を参照してください。
2. ステートメントに対して Explain 機能を実行して、最適化ガイドラインが有用かどうかを判断します。それが有用と思われるならば、次に進みます。
3. 以下のような照会を実行して、元のステートメントを取得します。

```
select statement_text
  from explain_statement
 where explain_level = '0' and
        explain_requester = 'SIMMEN' and
        explain_time      = '2003-09-08-16.01.04.108161' and
        source_name       = 'SQLC2E03' and
        source_version    = '' and
        queryno           = 1
```

- 最適化プロファイルを編集し、ステートメント・キーにステートメント・テキストを挿入してステートメント・プロファイルを作成します。以下に例を示します。

```
<STMTPROFILE ID="Guidelines for TPCD Q9">
  <STMTKEY SCHEMA="TPCD"><![CDATA[SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE,
    S.S_COMMENT
    FROM PARTS P, SUPPLIERS S, PARTSUPP PS
    WHERE P.PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY
    AND P.P_SIZE = 39 AND P.P_TYPE = 'BRASS' AND S.S_NATION
    = 'MOROCCO' AND
    PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
    FROM PARTSUPP PS1, SUPPLIERS S1
    WHERE P.P_PARTKEY = PS1.PS_PARTKEY AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
    AND S1.S_NATION = S.S_NATION)]]>
  </STMTKEY>
</STMTPROFILE>
```

- ステートメント・キーの後に、ステートメント・レベルの最適化ガイドラインを挿入します。直接的な名前を使用して、アクセス要求および結合要求で参照されるオブジェクトを識別します。以下に、結合要求の例を示します。

```
<OPTGUIDELINES>
  <HSJOIN>
    <TBSCAN TABLE='PS1' />
    <IXSCAN TABLE='S1'
      INDEX='I1' />
  </HSJOIN>
</OPTGUIDELINES>
```

- ファイルを妥当性検査して保管します。

タスクの結果

期待した結果が得られない場合は、適宜、ガイドラインを変更するか、追加のガイドラインを作成し、最適化プロファイルを更新します。

最適化ガイドラインでの表参照の形成:

表参照 という語は、SQL ステートメント内またはビュー定義内のすべての表、ビュー、表式、または別名が参照する表を意味します。最適化ガイドラインは、元のステートメントにある直接的な名前を使用するか、あるいは最適化ステートメントの表参照に関連付けられた固有な相関名を使用して、表参照を識別することができます。

直接的な名前の連続である拡張名は、ビューに組み込まれている表参照を一意的に識別するのに役立ちます。別名は、最適化ガイドラインの表参照として使用できません。そのような場合には、表参照のガイドラインは無視されることとなります。ステートメント全体のコンテキストで固有ではない直接的な名前や拡張名を識別する最適化ガイドラインは、未確定と見なされ、適用されません。さらに、同一の表参照を識別する複数の最適化ガイドラインが存在する場合は、その表参照を識別するすべての最適化ガイドラインが競合していると見なされ、適用されません。照会のトランスフォーメーションが行われる可能性があるため、直接的な名前または拡張名が最適化の間もそのまま存在するという保証はありません。そのため、そのような表参照を識別するすべてのガイドラインが無視されます。

元のステートメントにある直接的な名前を使用した表参照の識別

表参照を、表の直接的な名前を使用して識別します。直接的な名前は、SQL ステートメント内で表が修飾されるときと同じ要領で指定されます。

SQL ID を指定するときの規則が、最適化ガイドラインの TABLE 属性の値にも適用されます。TABLE 属性の値は、ステートメントの直接的な名前とそれぞれ比較されます。この DB2 リリースでは、1 つの一致だけが許可されます。TABLE 属性値がスキーマ修飾されている場合、その値は、同等の直接的な修飾表名すべてと一致します。TABLE 属性値が修飾されていない場合、その値は、同等の相関名または直接的な表名すべてと一致します。そのため、TABLE 属性値は、そのステートメントで有効なデフォルトのスキーマによって暗黙的に修飾されたものと見なされます。これらの概念について、以下に例を示します。ステートメントはデフォルトのスキーマ Tpcd を使用して最適化されるものと想定します。

```
select s_name, s_address, s_phone, s_comment
  from parts, suppliers, partsupp ps
  where p_partkey = ps.ps_partkey and
        s_s_suppkey = ps.ps_suppkey and
        p_size = 39 and
        p_type = 'BRASS'
```

ステートメントの表参照を識別する TABLE 属性値には、""Tpcd".PARTS"、'PARTS'、'Parts' (ID に区切りが入っていないため、これは大文字に変換されます) があります。ステートメントの表参照の識別に失敗する TABLE 属性値には、""Tpcd2"、SUPPLIERS'、'PARTSUPP' (直接的な名前ではない)、および 'Tpcd.PARTS' (ID Tpcd には区切りが入っていなければならない、そうでないと大文字に変換されてしまう) があります。

直接的な名前は、元のステートメント、ビュー、SQL 関数、またはトリガーにあるどの表参照をターゲットにする際にも使用できます。

元のステートメントにある直接的な名前を使用したビューの表参照の識別

以下の例に示されているように、最適化ガイドラインでは、拡張構文を使用してビューに組み込まれた表参照を識別することができます。

```
create view "Rick".v1 as
  (select * from employee a where salary > 50000)

create view "Gustavo".v2 as
  (select * from "Rick".v1
   where deptno in ('52', '53', '54'))

select * from "Gustavo".v2 a
  where v2.hire_date > '01/01/2004'

<OPTGUIDELINES>
  <IXSCAN TABLE='A/"Rick".V1/A' />
</OPTGUIDELINES>
```

IXSCAN アクセス要求エレメントは、"Gustavo".V2 および "Rick".V1 の各ビューに組み込まれた EMPLOYEE 表参照に、索引スキャンを使用するように指定します。ビューの表参照を識別するための拡張構文は、スラッシュ文字によって分離された一連の直接的な名前です。TABLE 属性 A/"Rick".V1/A の値は、拡張構文の例です。シーケンスの最後の直接的な名前 (A) は、最適化ガイドラインのターゲットである表参照を識別します。シーケンスの最初の直接的な名前 (A) は、元のステート

メントで直接的に参照されるビューを識別します。中間の直接的な 1 つまたは複数の名前 ("Rick".V1) は、直接的なビュー参照からターゲット表参照へのパスの、ビュー参照に関係しています。最適化ガイドラインから直接的な名前への参照に関する規則 (前のセクションで説明されている) は、拡張構文の各ステップに適用されません。

ビューの EMPLOYEE 表参照の直接的な名前が、ステートメントによって直接的または間接的に参照されるすべての表に関して固有であったなら、拡張名前構文は必要ありませんでした。

拡張構文は、元のステートメント、SQL 関数、またはトリガーのどの表参照をターゲットにする際にも使用できます。

最適化ステートメントの相関名を使用した表参照の識別

最適化ガイドラインでは、最適化ステートメントの表参照に関連付けられた固有の相関名を使用して表参照を識別することもできます。最適化ステートメントは、元のステートメントと意味的に同等なバージョンで、最適化の照会書き直しのフェーズで決定されるものです。最適化ステートメントは、EXPLAIN 表から取得できます。最適化ステートメントの表参照の識別には、最適化ガイドラインの TABID 属性が使用されます。例えば、

Original statement:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from sm_tpcd.parts p, sm_tpcd.suppliers s, sm_tpcd.partsupp ps
 where p_partkey = ps.ps_partkey and
       s.s_suppkey = ps.ps_suppkey and
       p.p_size = 39 and
       p.p_type = 'BRASS' and
       s.s_nation in ('MOROCCO', 'SPAIN') and
       ps.ps_supplycost = (select min(ps1.ps_supplycost)
                           from sm_tpcd.partsupp ps1, sm_tpcd.suppliers s1
                           where p.p_partkey = ps1.ps_partkey and
                                s1.s_suppkey = ps1.ps_suppkey and
                                s1.s_nation = s.s_nation)

<OPTGUIDELINES>
  <HSJOIN>
    <TBSCAN TABLE='S1' />
    <IXSCAN TABID='Q2' />
  </HSJOIN>
</OPTGUIDELINES>
```

Optimized statement:

```
select q6.s_name as "S_NAME", q6.s_address as "S_ADDRESS",
       q6.s_phone as "S_PHONE", q6.s_comment as "S_COMMENT"
  from (select min(q4.$c0)
        from (select q2.ps_supplycost
              from sm_tpcd.suppliers as q1, sm_tpcd.partsupp as q2
              where q1.s_nation = 'MOROCCO' and
                   q1.s_suppkey = q2.ps_suppkey and
                   q7.p_partkey = q2.ps_partkey
              ) as q3
        ) as q4, sm_tpcd.partsupp as q5, sm_tpcd.suppliers as q6,
       sm_tpcd.parts as q7
 where p_size = 39 and
       q5.ps_supplycost = q4.$c0 and
```

```
q6.s_nation in ('MOROCCO', 'SPAIN') and
q7.p_type = 'BRASS' and
q6.s_suppkey = q5.ps_suppkey and
q7.p_partkey = q5.ps_partkey
```

この最適化ガイドラインは、ハッシュ結合要求を示しています。ここで、TBSCAN アクセス要求エレメントによって指定されているように、ネストされた副選択内の SUPPLIERS 表が外部表です。また、IXSCAN アクセス要求エレメントによって指定されているように、ネストされた副選択内の PARTSUPP 表が内部表です。TBSCAN アクセス要求エレメントは、TABLE 属性を使用して SUPPLIERS 表参照を識別しますが、その際、元のステートメントにある対応する直接的な名前を使用します。一方、IXSCAN アクセス要求エレメントは、TABID 属性を使用して PARTSUPP 表参照を識別しますが、その際、最適化ステートメントの表参照に関連付けられた固有の相関名を使用します。

1 つの最適化ガイドラインに TABLE 属性と TABID 属性の両方が指定されている場合は、両方の属性が同じ表参照を識別している必要があります。そうでない場合、その最適化ガイドラインは無視されます。

注: 最適化ステートメントの相関名については、現在のところ、DB2 製品の新しいリリースへのアップグレードを行っても変わらないという保証はありません。

未確定表参照

最適化ガイドラインが複数の直接的な名前または拡張名と一致する場合、その最適化ガイドラインは無効であると見なされ、適用されません。例えば、

```
create view v1 as
  (select * from employee
   where salary > (select avg(salary) from employee)

select * from v1
  where deptno in ('M62', 'M63')

<OPTGUIDE>
  <IXSCAN TABLE='V1/EMPLOYEE'/>
</OPTGUIDE>
```

直接的な名前 EMPLOYEE がビュー V1 の定義内で固有でないため、オプティマイザは、IXSCAN アクセス要求を未確定と見なします。

このあいまいさをなくすためには、固有の相関名を使用するようにビューを書き直すか、TABID 属性を使用することができます。最適化ステートメントの相関名はすべて固有であるため、TABID 属性で識別された表参照は未確定になることがあります。

競合する最適化ガイドライン

同じ表参照を複数の最適化ガイドラインで識別することはできません。例えば、

```
<OPTGUIDELINES>
  <IXSCAN TABLE='"Tpcd".PARTS' INDEX='I_PTYPE'/>
  <IXSCAN TABLE='"Tpcd".PARTS' INDEX='I_SIZE'/>
</OPTGUIDELINES>
```

IXSCAN エレメントはそれぞれ、メインの副選択内の "Tpcd".PARTS 表を参照します。

複数のガイドラインが同じ表を参照する場合は、最初のガイドラインだけが適用されます。それ以外のガイドラインはすべて無視され、エラーが戻ります。

各照会の述部レベルで使用可能にできるのは、INLIST2JOIN 照会再書き込み要求エレメント 1 つだけです。以下の例は、サポートされていない照会書き直し最適化ガイドラインを示しています。ここでは、2 つの IN-LIST 述部が述部レベルで有効にされています。両方のガイドラインが無視され、警告が戻されます。

```
<OPTGUIDELINES>
  <INLIST2JOIN TABLE='P' COLUMN='P_SIZE' />
  <INLIST2JOIN TABLE='P' COLUMN='P_TYPE' />
</OPTGUIDELINES>
```

最適化ガイドラインが使用されていることの確認:

オプティマイザーは、最適化プロファイルで指定された最適化ガイドラインに可能な限り従うようにしますが、無効または不適切なガイドラインについてはこれを拒否します。

始める前に

Explain 表が存在しないと、Explain 機能を使用できません。Explain 表を作成するためのデータ定義言語 (DDL) は、EXPLAIN.DDL に含まれています。これは、sqllib ディレクトリーの misc サブディレクトリーにあります。

手順

有効な最適化ガイドラインが使用されていることを確認するには、次のようにします。

1. ガイドラインの適用先であるステートメントに対して、EXPLAIN ステートメントを発行します。最適化ガイドラインが、最適化プロファイルを使用するステートメントで施行された場合には、その最適化プロファイルの名前が、EXPLAIN ARGUMENT 表に RETURN 演算子の引数として表示されます。最適化ガイドラインに含まれる SQL 組み込み最適化ガイドラインまたはステートメント・プロファイルが、現行のステートメントと一致する場合は、ステートメント・プロファイルの名前が RETURN 演算子の引数として表示されます。新しいこれら 2 つの引数値のタイプは、OPT_PROF と STMTPROF です。
2. EXPLAIN されたステートメントの結果を調べてください。以下の Explain 表に対する照会を変更して、EXPLAIN_REQUESTER、EXPLAIN_TIME、SOURCE_NAME、SOURCE_VERSION、および QUERYNO の特定の組み合わせで最適化プロファイル名やステートメント・プロファイル名を戻すようにできます。

```
SELECT VARCHAR(B.ARGUMENT_TYPE, 9) as TYPE,
       VARCHAR(B.ARGUMENT_VALUE, 24) as VALUE

FROM   EXPLAIN_STATEMENT A, EXPLAIN_ARGUMENT B

WHERE  A.EXPLAIN_REQUESTER = 'SIMMEN'
AND    A.EXPLAIN_TIME      = '2003-09-08-16.01.04.108161'
AND    A.SOURCE_NAME       = 'SQLC2E03'
AND    A.SOURCE_VERSION    = ''
AND    A.QUERYNO           = 1

AND    A.EXPLAIN_REQUESTER = B.EXPLAIN_REQUESTER
AND    A.EXPLAIN_TIME      = B.EXPLAIN_TIME
```

```

AND A.SOURCE_NAME      = B.SOURCE_NAME
AND A.SOURCE_SCHEMA    = B.SOURCE_SCHEMA
AND A.SOURCE_VERSION   = B.SOURCE_VERSION
AND A.EXPLAIN_LEVEL    = B.EXPLAIN_LEVEL
AND A.STMTNO           = B.STMTNO
AND A.SECTNO           = B.SECTNO

AND A.EXPLAIN_LEVEL    = 'P'

AND (B.ARGUMENT_TYPE = 'OPT_PROF' OR ARGUMENT_TYPE = 'STMTPROF')
AND B.OPERATOR_ID = 1

```

最適化ガイドラインがアクティブで、EXPLAIN されたステートメントが最適化ガイドラインの STMTKEY エレメントに含まれているステートメントと一致する場合、上記の例と同様の照会では、以下に示す出力と同様の出力が生成されます。STMTPROF 引数の値は、STMTPROFILE エレメントの ID 属性と同じです。

```

TYPE      VALUE
-----
OPT_PROF  NEWTON.PROFILE1
STMTPROF  Guidelines for TPCD Q9

```

最適化プロファイルおよびガイドライン用の XML スキーマ:

現行の最適化プロファイル・スキーマ:

所定の DB2 リリースに有効な最適化プロファイルの内容は、現行最適化プロファイル・スキーマ (COPS) として知られる XML スキーマによって記述されます。最適化プロファイルが適用されるのは DB2 Database for Linux, UNIX, and Windows サーバーだけです。

以下のリストは、DB2 製品の現行リリースにおける COPS の見本です。COPS は DB2OptProfile.xsd にもあり、これは、sqllib ディレクトリーの misc サブディレクトリーにあります。

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" version="1.0">
<!-- *****-->
<!-- Licensed Materials - Property of IBM -->
<!-- (C) Copyright International Business Machines Corporation 2009. All rights reserved. -->
<!-- U.S. Government Users Restricted Rights; Use, duplication or disclosure restricted by -->
<!-- GSA ADP Schedule Contract with IBM Corp. -->
<!-- *****-->
<!-- *****-->
<!-- Definition of the current optimization profile schema for V9.7.0.0 -->
<!-- -->
<!-- An optimization profile is composed of the following sections: -->
<!-- -->
<!-- + A global optimization guidelines section (at most one) which defines optimization -->
<!-- guidelines affecting any statement for which the optimization profile is in effect. -->
<!-- -->
<!-- + Zero or more statement profile sections, each of which defines optimization -->
<!-- guidelines for a particular statement for which the optimization profile -->
<!-- is in effect. -->
<!-- -->
<!-- The VERSION attribute indicates the version of this optimization profile -->
<!-- schema. -->
<!-- *****-->
<xs:element name="OPTPROFILE">
  <xs:complexType>
    <xs:sequence>
      <!-- Global optimization guidelines section. At most one can be specified. -->
      <xs:element name="OPTGUIDELINES" type="globalOptimizationGuidelinesType" minOccurs="0"/>
      <!-- Statement profile section. Zero or more can be specified -->
      <xs:element name="STMTPROFILE" type="statementProfileType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        <!-- Version attribute is currently optional -->
        <xs:attribute name="VERSION" use="optional"/>
    </xs:complexType>
</xs:element>

<!-- *****-->
<!-- Global optimization guidelines supported in this version: -->
<!-- + MQTOptimizationChoices elements influence the MQTs considered by the optimizer. -->
<!-- + computationalPartitionGroupOptimizationsChoices elements can affect repartitioning -->
<!-- optimizations involving nicknames. -->
<!-- + General requests affect the search space which defines the alternative query -->
<!-- transformations, access methods, join methods, join orders, and other optimizations, -->
<!-- considered by the compiler and optimizer. -->
<!-- + MQT enforcement requests specify semantically matchable MQTs whose usage in access -->
<!-- plans should be enforced regardless of cost estimates. -->
<!-- *****-->
<xs:complexType name="globalOptimizationGuidelinesType">
    <xs:sequence>
        <xs:group ref="MQTOptimizationChoices" />
        <xs:group ref="computationalPartitionGroupOptimizationChoices" />
        <xs:group ref="generalRequest"/>
        <xs:group ref="mqtEnforcementRequest" />
    </xs:sequence>
</xs:complexType>

<!-- *****-->
<!-- Elements for affecting materialized query table (MQT) optimization. -->
<!-- -->
<!-- -->
<!-- + MQTOPT - can be used to disable materialized query table (MQT) optimization. -->
<!-- If disabled, the optimizer will not consider MQTs to optimize the statement. -->
<!-- -->
<!-- -->
<!-- + MQT - multiple of these can be specified. Each specifies an MQT that should be -->
<!-- considered for optimizing the statement. Only specified MQTs will be considered. -->
<!-- -->
<!-- *****-->
<xs:group name="MQTOptimizationChoices">
    <xs:choice>
        <xs:element name="MQTOPT" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:attribute name="OPTION" type="optionType" use="optional"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="MQT" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
                <xs:attribute name="NAME" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:choice>
</xs:group>

<!-- *****-->
<!-- Elements for affecting computational partition group (CPG) optimization. -->
<!-- -->
<!-- -->
<!-- + PARTOPT - can be used disable the computational partition group (CPG) optimization -->
<!-- which is used to dynamically redistributes inputs to join, aggregation, -->
<!-- and union operations when those inputs are results of remote queries. -->
<!-- -->
<!-- -->
<!-- + PART - Define the partition groups to be used in CPG optimizations. -->
<!-- -->
<!-- *****-->
<xs:group name="computationalPartitionGroupOptimizationChoices">
    <xs:choice>
        <xs:element name="PARTOPT" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:attribute name="OPTION" type="optionType" use="optional"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="PART" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:attribute name="NAME" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:choice>
</xs:group>

<!-- *****-->
<!-- Definition of a statement profile. -->
<!-- Comprised of a statement key and optimization guidelines. -->
<!-- The statement key specifies semantic information used to identify the statement to -->
<!-- which optimization guidelines apply. The optional ID attribute provides the statement -->
<!-- profile with a name for use in EXPLAIN output. -->
<!-- *****-->

```

```

<xs:complexType name="statementProfileType">
  <xs:sequence>
    <!-- Statement key element -->
    <xs:element name="STMTKEY" type="statementKeyType"/>
    <xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
  </xs:sequence>
  <!-- ID attribute.Used in explain output to indicate the statement profile was used. -->
  <xs:attribute name="ID" type="xs:string" use="optional"/>
</xs:complexType>
<!--*****-->
<!-- Definition of the statement key. The statement key provides semantic information used -->
<!-- to identify the statement to which the optimization guidelines apply. -->
<!-- The statement key is comprised of: -->
<!-- + statement text (as written in the application) -->
<!-- + default schema (for resolving unqualified table names in the statement) -->
<!-- + function path (for resolving unqualified types and functions in the statement) -->
<!-- The statement text is provided as element data whereas the default schema and function -->
<!-- path are provided via the SCHEMA and FUNCPATH elements, respectively. -->
<!--*****-->
<xs:complexType name="statementKeyType" mixed="true">
  <xs:attribute name="SCHEMA" type="xs:string" use="optional"/>
  <xs:attribute name="FUNCPATH" type="xs:string" use="optional"/>
</xs:complexType>
<!--*****-->
<!-- -->
<!-- Optimization guideline elements can be chosen from general requests, rewrite -->
<!-- requests access requests, or join requests. -->
<!-- -->
<!-- General requests affect the search space which defines the alternative query -->
<!-- transformations, access methods, join methods, join orders, and other optimizations, -->
<!-- considered by the optimizer. -->
<!-- -->
<!-- Rewrite requests affect the query transformations used in determining the optimized -->
<!-- statement. -->
<!-- -->
<!-- Access requests affect the access methods considered by the cost-based optimizer, -->
<!-- and join requests affect the join methods and join order used in the execution plan. -->
<!-- -->
<!-- MQT enforcement requests specify semantically matchable MQTs whose usage in access -->
<!-- plans should be enforced regardless of cost estimates. -->
<!-- -->
<!--*****-->
<xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
<xs:complexType name="optGuidelinesType">
  <xs:sequence>
    <xs:group ref="generalRequest" minOccurs="0" maxOccurs="1"/>
    <xs:choice maxOccurs="unbounded">
      <xs:group ref="rewriteRequest" />
      <xs:group ref="accessRequest"/>
      <xs:group ref="joinRequest"/>
      <xs:group ref="mqtEnforcementRequest"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<!--*****-->
<!-- Choices of general request elements. -->
<!-- REOPT can be used to override the setting of the REOPT bind option. -->
<!-- DPFXMLMOVEMENT can be used to affect the optimizer's plan when moving XML documents -->
<!-- between database partitions. The value can be NONE, REFERENCE or COMBINATION. The -->
<!-- default value is NONE. -->
<!--*****-->
<xs:group name="generalRequest">
  <xs:sequence>
    <xs:element name="REOPT" type="reoptType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="DEGREE" type="degreeType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="QRYOPT" type="qryoptType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RTS" type="rtsType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="DPFXMLMOVEMENT" type="dpfXMLMovementType" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:group>
<!--*****-->
<!-- Choices of rewrite request elements. -->
<!--*****-->
<xs:group name="rewriteRequest">
  <xs:sequence>
    <xs:element name="INLIST2JOIN" type="inListToJoinType" minOccurs="0"/>
    <xs:element name="SUBQ2JOIN" type="subqueryToJoinType" minOccurs="0"/>
    <xs:element name="NOTEX2AJ" type="notExistsToAntiJoinType" minOccurs="0"/>
  </xs:sequence>
</xs:group>

```

```

        <xs:element name="NOTIN2AJ" type="notInToAntiJoinType" minOccurs="0"/>
    </xs:sequence>
</xs:group>
<!--***** -->
<!-- Choices for access request elements. -->
<!-- TBSCAN - table scan access request element -->
<!-- IXSCAN - index scan access request element -->
<!-- LPREFETCH - list prefetch access request element -->
<!-- IXAND - index ANDing access request element -->
<!-- IXOR - index ORing access request element -->
<!-- XISCAN - xml index access request element -->
<!-- XANDOR - XANDOR access request element -->
<!-- ACCESS - indicates the optimizer should choose the access method for the table -->
<!--***** -->
<xs:group name="accessRequest">
    <xs:choice>
        <xs:element name="TBSCAN" type="tableScanType"/>
        <xs:element name="IXSCAN" type="indexScanType"/>
        <xs:element name="LPREFETCH" type="listPrefetchType"/>
        <xs:element name="IXAND" type="indexAndingType"/>
        <xs:element name="IXOR" type="indexOringType"/>
        <xs:element name="XISCAN" type="indexScanType"/>
        <xs:element name="XANDOR" type="XANDORType"/>
        <xs:element name="ACCESS" type="anyAccessType"/>
    </xs:choice>
</xs:group>
<!--***** -->
<!-- Choices for join request elements. -->
<!-- NLJOIN - nested-loops join request element -->
<!-- MSJOIN - sort-merge join request element -->
<!-- HSJOIN - hash join request element -->
<!-- JOIN - indicates that the optimizer is to choose the join method. -->
<!--***** -->
<xs:group name="joinRequest">
    <xs:choice>
        <xs:element name="NLJOIN" type="nestedLoopJoinType"/>
        <xs:element name="HSJOIN" type="hashJoinType"/>
        <xs:element name="MSJOIN" type="mergeJoinType"/>
        <xs:element name="JOIN" type="anyJoinType"/>
    </xs:choice>
</xs:group>
<!--***** -->
<!-- MQT enforcement request element. -->
<!-- MQTENFORCE - This element can be used to specify semantically matchable MQTs whose -->
<!-- usage in access plans should be enforced regardless of Optimizer cost estimates. -->
<!-- MQTs can be specified either directly with the NAME attribute or generally using -->
<!-- the TYPE attribute. -->
<!-- Only the first valid attribute found is used and all subsequent ones are ignored. -->
<!-- Since this element can be specified multiple times, more than one MQT can be -->
<!-- enforced at a time. -->
<!-- Note however, that if there is a conflict when matching two enforced MQTs to the -->
<!-- same data source (base-table or derived) an MQT will be picked based on existing -->
<!-- tie-breaking rules, i.e., either heuristic or cost-based. -->
<!-- Finally, this request overrides any other MQT optimization options specified in -->
<!-- a profile, i.e., enforcement will take place even if MQTOPT is set to DISABLE or -->
<!-- if the specified MQT or MQTs do not exist in the eligibility list specified by -->
<!-- any MQT elements. -->
<!--***** -->
<xs:group name="mqtEnforcementRequest">
    <xs:sequence>
        <xs:element name="MQTENFORCE" type="mqtEnforcementType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<!--***** -->
<!-- REOPT general request element. Can override REOPT setting at the package, db, -->
<!-- dbm level. -->
<!--***** -->
<xs:complexType name="reoptType">
    <xs:attribute name="VALUE" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="ONCE"/>
                <xs:enumeration value="ALWAYS"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
<!--***** -->
<!-- RTS general request element to enable, disable or provide a time budget for -->

```

```

<!-- real-time statistics collection. -->
<!-- OPTION attribute allows enabling or disabling real-time statistics. -->
<!-- TIME attribute provides a time budget in milliseconds for real-time statistics collection.-->
<!--*****-->
<xs:complexType name="rtsType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
  <xs:attribute name="TIME" type="xs:nonNegativeInteger" use="optional"/>
</xs:complexType>
<!--*****-->
<!-- Definition of an "IN list to join" rewrite request -->
<!-- OPTION attribute allows enabling or disabling the alternative. -->
<!-- TABLE attribute allows request to target IN list predicates applied to a -->
<!-- specific table reference. COLUMN attribute allows request to target a specific IN list -->
<!-- predicate. -->
<!--*****-->
<xs:complexType name="inListToJoinType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
  <xs:attribute name="TABLE" type="xs:string" use="optional"/>
  <xs:attribute name="COLUMN" type="xs:string" use="optional"/>
</xs:complexType>
<!--*****-->
<!-- Definition of a "subquery to join" rewrite request -->
<!-- The OPTION attribute allows enabling or disabling the alternative. -->
<!--*****-->
<xs:complexType name="subqueryToJoinType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
<!--*****-->
<!-- Definition of a "not exists to anti-join" rewrite request -->
<!-- The OPTION attribute allows enabling or disabling the alternative. -->
<!--*****-->
<xs:complexType name="notExistsToAntiJoinType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
<!--*****-->
<!-- Definition of a "not IN to anti-join" rewrite request -->
<!-- The OPTION attribute allows enabling or disabling the alternative. -->
<!--*****-->
<xs:complexType name="notInToAntiJoinType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
<!--*****-->
<!-- Effectively the superclass from which all access request elements inherit. -->
<!-- This type currently defines TABLE and TABID attributes, which can be used to tie an -->
<!-- access request to a table reference in the query. -->
<!-- The TABLE attribute value is used to identify a table reference using identifiers -->
<!-- in the original SQL statement. The TABID attribute value is used to identify a table -->
<!-- referece using the unique correlation name provided via the -->
<!-- optimized statement. If both the TABLE and TABID attributes are specified, the TABID -->
<!-- field is ignored. The FIRST attribute indicates that the access should be the first -->
<!-- access in the join sequence for the FROM clause. -->
<!-- The SHARING attribute indicates that the access should be visible to other concurrent -->
<!-- similar accesses that may therefore share bufferpool pages. The WRAPPING attribute -->
<!-- indicates that the access should be allowed to perform wrapping, thereby allowing it to -->
<!-- start in the middle for better sharing with other concurrent accesses. The THROTTLE -->
<!-- attribute indicates that the access should be allowed to be throttled if this may -->
<!-- benefit other concurrent accesses. The SHARESPEED attribute is used to indicate whether -->
<!-- the access should be considered fast or slow for better grouping of concurrent accesses. -->
<!--*****-->
<xs:complexType name="accessType" abstract="true">
  <xs:attribute name="TABLE" type="xs:string" use="optional"/>
  <xs:attribute name="TABID" type="xs:string" use="optional"/>
  <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
  <xs:attribute name="SHARING" type="optionType" use="optional" default="ENABLE"/>
  <xs:attribute name="WRAPPING" type="optionType" use="optional" default="ENABLE"/>
  <xs:attribute name="THROTTLE" type="optionType" use="optional" default="ENABLE"/>
  <xs:attribute name="SHARESPEED" type="shareSpeed" use="optional"/>
</xs:complexType>
<!--*****-->
<!-- Definition of an table scan access request method. -->
<!--*****-->
<xs:complexType name="tableScanType">
  <xs:complexContent>
    <xs:extension base="accessType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Definition of an index scan access request element. The index name is optional. -->
<!--*****-->

```

```

<xs:complexType name="indexScanType">
  <xs:complexContent>
    <xs:extension base="accessType">
      <xs:attribute name="INDEX" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Definition of a list prefetch access request element. The index name is optional. -->
<!-- *****-->
<xs:complexType name="listPrefetchType">
  <xs:complexContent>
    <xs:extension base="accessType">
      <xs:attribute name="INDEX" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Definition of an extended access element which will be used by IXAND and ACCESS -->
<!-- requests. -->
<!-- A single index scan be specified via the INDEX attribute. Multiple indexes -->
<!-- can be specified via INDEX elements. The index element specification supersedes the -->
<!-- attribute specification. If a single index is specified, the optimizer will use the -->
<!-- index as the first index of the index ANDing access method and will choose addi- -->
<!-- tional indexes using cost. If multiple indexes are specified the optimizer will -->
<!-- use exactly those indexes in the specified order. If no indexes are specified -->
<!-- via either the INDEX attribute or INDEX elements, then the optimizer will choose -->
<!-- all indexes based upon cost. -->
<!-- Extension for XML support: -->
<!-- TYPE: Optional attribute. The allowed value is XMLINDEX. When the type is not -->
<!-- specified, the optimizer makes a cost based decision. -->
<!-- ALLINDEXES: Optional attribute. The allowed value is TRUE. The default -->
<!-- value is FALSE. -->
<!-- *****-->
<xs:complexType name="extendedAccessType">
  <xs:complexContent>
    <xs:extension base="accessType">
      <xs:sequence minOccurs="0">
        <xs:element name="INDEX" type="indexType" minOccurs="2" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      <xs:attribute name="TYPE" type="xs:string" use="optional" fixed="XMLINDEX"/>
      <xs:attribute name="ALLINDEXES" type="boolType" use="optional" fixed="TRUE"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Definition of an index ANDing access request element. -->
<!-- Extension for XML support: -->
<!-- All attributes and elements in extendedAccessType are included. -->
<!-- Note that ALLINDEXES is a valid option only if TYPE is XMLINDEX. -->
<!-- STARJOIN index ANDing: Specifying STARJOIN='TRUE' or one or more NLJOIN elements -->
<!-- identifies the index ANDing request as a star join index ANDing request. When that -->
<!-- is the case: -->
<!-- TYPE cannot be XMLINDEX (and therefore ALLINDEXES cannot be specified). -->
<!-- Neither the INDEX attribute nor INDEX elements can be specified. -->
<!-- The TABLE or TABID attribute identifies the fact table. -->
<!-- Zero or more semijoins can be specified using NLJOIN elements. -->
<!-- If no semijoins are specified, the optimizer will choose them. -->
<!-- If a single semijoin is specified, the optimizer will use it as the first semijoin -->
<!-- and will choose the rest itself. -->
<!-- If multiple semijoins are specified the optimizer will use exactly those semijoins -->
<!-- in the specified order. -->
<!-- *****-->
<xs:complexType name="indexAndingType">
  <xs:complexContent>
    <xs:extension base="extendedAccessType">
      <xs:sequence minOccurs="0">
        <xs:element name="NLJOIN" type="nestedLoopJoinType" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="STARJOIN" type="boolType" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Definition of an INDEX element method. Index set is optional. If specified, -->
<!-- at least 2 are required. -->
<!-- *****-->
<xs:complexType name="indexType">

```

```

    <xs:attribute name="IXNAME" type="xs:string" use="optional"/>
</xs:complexType>
<!-- *****-->
<!-- Definition of an XANDOR access request method. -->
<!-- *****-->
<xs:complexType name="XANDORType">
  <xs:complexContent>
    <xs:extension base="accessType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Use for derived table access or other cases where the access method is not of -->
<!-- consequence. -->
<!-- Extension for XML support: -->
<!-- All attributes and elements in extendedAccessType are included. -->
<!-- Note that INDEX attribute/elements and ALLINDEXES are valid options only if TYPE -->
<!-- is XMLINDEX. -->
<!-- *****-->
<xs:complexType name="anyAccessType">
  <xs:complexContent>
    <xs:extension base="extendedAccessType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Definition of an index ORing access -->
<!-- Cannot specify more details (e.g indexes). Optimizer will choose the details based -->
<!-- upon cost. -->
<!-- *****-->
<xs:complexType name="indexOringType">
  <xs:complexContent>
    <xs:extension base="accessType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Effectively the super class from which join request elements inherit. -->
<!-- This type currently defines join element inputs and also the FIRST attribute. -->
<!-- A join request must have exactly two nested sub-elements. The sub-elements can be -->
<!-- either an access request or another join request. The first sub-element represents -->
<!-- outer table of the join operation while the second element represents the inner -->
<!-- table. The FIRST attribute indicates that the join result should be the first join -->
<!-- relative to other tables in the same FROM clause. -->
<!-- *****-->
<xs:complexType name="joinType" abstract="true">
  <xs:choice minOccurs="2" maxOccurs="2">
    <xs:group ref="accessRequest"/>
    <xs:group ref="joinRequest"/>
  </xs:choice>
  <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
</xs:complexType>
<!-- *****-->
<!-- Definition of nested loop join access request. Subclass of joinType. -->
<!-- Does not add any elements or attributes. -->
<!-- *****-->
<xs:complexType name="nestedLoopJoinType">
  <xs:complexContent>
    <xs:extension base="joinType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Definition of merge join access request. Subclass of joinType. -->
<!-- Does not add any elements or attributes. -->
<!-- *****-->
<xs:complexType name="mergeJoinType">
  <xs:complexContent>
    <xs:extension base="joinType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Definition of hash join access request. Subclass of joinType. -->
<!-- Does not add any elements or attributes. -->
<!-- *****-->
<xs:complexType name="hashJoinType">
  <xs:complexContent>
    <xs:extension base="joinType"/>
  </xs:complexContent>
</xs:complexType>
<!-- *****-->
<!-- Any join is a subclass of binary join. Does not extend it in any way. -->
<!-- Does not add any elements or attributes. -->

```



```

<!--***** -->
<xs:complexType name="anyJoinType">
  <xs:complexContent>
    <xs:extension base="joinType"/>
  </xs:complexContent>
</xs:complexType>
<!--***** -->
<!-- The MQTENFORCE element can be specified with one of two attributes: -->
<!-- NAME: Specify the MQT name directly as a value to this attribute. -->
<!-- TYPE: Specify the type of the MQTs that should be enforced with this attribute. -->
<!-- Note that only the value of the first valid attribute found will be used. All -->
<!-- subsequent attributes will be ignored. -->
<!--***** -->
<xs:complexType name="mqtEnforcementType">
  <xs:attribute name="NAME" type="xs:string"/>
  <xs:attribute name="TYPE" type="mqtEnforcementTypeType"/>
</xs:complexType>
<!--***** -->
<!-- Allowable values for the TYPE attribute of an MQTENFORCE element: -->
<!-- NORMAL: Enforce usage of all semantically matchable MQTs, except replicated MQTs. -->
<!-- REPLICATED: Enforce usage of all semantically matchable replicated MQTs only. -->
<!-- ALL: Enforce usage of all semantically matchable MQTs. -->
<!--***** -->
<xs:simpleType name="mqtEnforcementTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="NORMAL"/>
    <xs:enumeration value="REPLICATED"/>
    <xs:enumeration value="ALL"/>
  </xs:restriction>
</xs:simpleType>
<!--***** -->
<!-- Allowable values for a boolean attribute. -->
<!--***** -->
<xs:simpleType name="boolType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="TRUE"/>
    <xs:enumeration value="FALSE"/>
  </xs:restriction>
</xs:simpleType>
<!--***** -->
<!-- Allowable values for an OPTION attribute. -->
<!--***** -->
<xs:simpleType name="optionType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ENABLE"/>
    <xs:enumeration value="DISABLE"/>
  </xs:restriction>
</xs:simpleType>
<!--***** -->
<!-- Allowable values for a SHARESPEED attribute. -->
<!--***** -->
<xs:simpleType name="shareSpeed">
  <xs:restriction base="xs:string">
    <xs:enumeration value="FAST"/>
    <xs:enumeration value="SLOW"/>
  </xs:restriction>
</xs:simpleType>
<!--***** -->
<!-- Definition of the qryopt type: the only values allowed are 0, 1, 2, 3, 5, 7 and 9 -->
<!--***** -->
<xs:complexType name="qryoptType">
  <xs:attribute name="VALUE" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="0"/>
        <xs:enumeration value="1"/>
        <xs:enumeration value="2"/>
        <xs:enumeration value="3"/>
        <xs:enumeration value="5"/>
        <xs:enumeration value="7"/>
        <xs:enumeration value="9"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
<!--***** -->
<!-- Definition of the degree type: any number between 1 and 32767 or the strings ANY or -1 -->
<!--***** -->
<xs:simpleType name="intStringType">

```

```

<xs:union>
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"></xs:minInclusive>
      <xs:maxInclusive value="32767"></xs:maxInclusive>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="ANY"/>
      <xs:enumeration value="-1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:union>
</xs:simpleType>

<xs:complexType name="degreeType">
  <xs:attribute name="VALUE" type="intStringType"/></xs:attribute>
</xs:complexType>
<!-- ***** -->
<!-- Definition of DPF XML movement types -->
<!-- ***** -->
<xs:complexType name="dpfXMLMovementType">
  <xs:attribute name="VALUE" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="REFERENCE"/>
        <xs:enumeration value="COMBINATION"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>

</xs:schema>

```

OPTPROFILE エlement用の XML スキーマ:

OPTPROFILE Elementは、最適化プロファイルのルートです。

Elementは、次のように定義します。

```

XML Schema
  <xs:element name="OPTPROFILE">
    <xs:complexType>
      <xs:sequence>
        <!-- Global optimization guidelines section. -->
        <!-- At most one can be specified. -->
        <xs:element name="OPTGUIDELINES"
          type="globalOptimizationGuidelinesType" minOccurs="0"/>
        <!-- Statement profile section. Zero or more can be specified -->
        <xs:element name="STMTPROFILE" type="statementProfileType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- Version attribute is currently optional -->
      <xs:attribute name="VERSION" use="optional"/>
    </xs:complexType>
  </xs:element>

```

説明

オプションの OPTGUIDELINES サブElementは、最適化プロファイルのグローバル最適化ガイドラインを定義します。各 STMTPROFILE サブElementは、ステートメント・プロファイルを定義します。VERSION 属性は、特定の最適化プロファイルの作成および妥当性検査時に照らし合わせる現在の最適化プロファイル・スキーマを識別します。

グローバル OPTGUIDELINES Elementの XML スキーマ:

OPTGUIDELINES エレメントは、最適化プロファイルのグローバル最適化ガイドラインを定義します。

これは、複合タイプ `globalOptimizationGuidelinesType` によって定義されます。

XML Schema

```
<xs:complexType name="globalOptimizationGuidelinesType">
  <xs:sequence>
    <xs:group ref="MQTOptimizationChoices"/>
    <xs:group ref="computationalPartitionGroupOptimizationChoices"/>
    <xs:group ref="generalRequest"/>
    <xs:group ref="mqtEnforcementRequest"/>
  </xs:sequence>
</xs:complexType>
```

説明

グローバル最適化ガイドラインの定義には、グループ

`MQTOptimizationChoices`、`computationalPartitionGroupOptimizationChoices`、または `generalRequest` のエレメントを使用できます。

- `MQTOptimizationChoices` グループ・エレメントを使用して、MQT 置換に作用することができます。
- `computationalPartitionGroupOptimizationChoices` グループ・エレメントを使用して、計算パーティション・グループの最適化に作用することができます。これには、リモート・データ・ソースから読み取られたデータの動的再配分が関係しています。これは、パーティション化されたフェデレーテッド・データベース構成にのみ当てはまります。
- `generalRequest` グループ・エレメントは最適化プロセスの特定のフェーズ固有ではなく、オプティマイザーの検索スペースを変更するために使用できます。グローバル、またはステートメント・レベルで指定できます。
- MQT 適用要求は、コストの見積もりに関係なくアクセス・プランでの使用が強制される、意味的に一致可能なマテリアライズ照会表 (MQT) を指定します。

MQT 最適化の選択項目:

`MQTOptimizationChoices` グループは、マテリアライズ照会表 (MQT) の最適化に作用するのに使用できる一連のエレメントを定義します。具体的には、これらのエレメントを使用して、MQT 置換の考慮を有効または無効することや、オプティマイザーで考慮される MQT の完全セットを指定することができます。

XML Schema

```
<xs:group name="MQTOptimizationChoices">
  <xs:choice>
    <xs:element name="MQTOPT" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:attribute name="OPTION" type="optionType" use="optional"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="MQT" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="NAME" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:choice>
</xs:group>
```

説明

MQTOPT エレメントは、MQT 最適化の考慮を有効または無効にするのに使用されます。OPTION 属性が取れるのは、値 ENABLE (デフォルト) または DISABLE です。

MQT エレメントの NAME 属性は、オプティマイザーで考慮の対象とされる MQT を識別します。NAME 属性内の MQT への参照の作成に関する規則は、直接的な表名への参照の作成の規則と同じです。1 つ以上の MQT エレメントを指定した場合、それらの MQT だけが、オプティマイザーで考慮の対象となります。1 つ以上の MQT を指定して MQT 置換を実行するかどうかの決定は、やはりコスト・ベースの決定になります。

例

以下の例は、MQT の最適化を無効にする方法を示しています。

```
<OPTGUIDELINES>
  <MQTOPT OPTION='DISABLE' />
</OPTGUIDELINES>
```

以下の例は、Tpcd.PARTSMQT 表および COLLEGE.STUDENTS 表に対して MQT 最適化を限定する方法を示しています。

```
<OPTGUIDELINES>
  <MQT NAME='Tpcd.PARTSMQT' />
  <MQT NAME='COLLEGE.STUDENTS' />
</OPTGUIDELINES>
```

計算パーティション・グループの最適化の選択項目:

computationalPartitionGroupOptimizationChoices グループは、計算パーティション・グループの最適化に作用するのに使用できる一連のエレメントを定義します。具体的には、これらのエレメントを使用して、計算グループの最適化を有効または無効にすることや、計算パーティション・グループの最適化に使用するパーティション・グループを指定することができます。

XML Schema

```
<xs:group name="computationalPartitionGroupOptimizationChoices">
  <xs:choice>
    <xs:element name="PARTOPT" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:attribute name="OPTION" type="optionType" use="optional"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="PART" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:attribute name="NAME" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:choice>
</xs:group>
```

説明

PARTOPT エレメントは、計算パーティション・グループの最適化の考慮を有効または無効にするのに使用されます。OPTION 属性が取れるのは、値 ENABLE (デフォルト) または DISABLE です。

PART エレメントを使用して、計算パーティション・グループの最適化に使用するパーティション・グループを指定することもできます。NAME 属性は、既存のパーティション・グループを識別するものでなければなりません。指定のパーティション・グループを使用した動的再配分の実行の決定は、引き続きコスト・ベースの決定になります。

例

以下の例は、計算パーティション・グループの最適化を無効にする方法を示しています。

```
<OPTGUIDELINES>
  <PARTOPT OPTION='DISABLE' />
</OPTGUIDELINES>
```

以下の例は、計算パーティション・グループの最適化に関して WORKPART パーティション・グループを使用することを指定する方法を示しています。

```
<OPTGUIDELINES>
  <MQT NAME='Tpcd.PARTSMQT' />
  <PART NAME='WORKPART' />
</OPTGUIDELINES>
```

グローバル要求としての一般最適化ガイドライン:

generalRequest グループは、最適化プロセスの特定のフェーズ固有のものではないガイドラインを定義し、オプティマイザの検索スペースを変更するために使用できます。

一般最適化ガイドラインは、グローバルおよびステートメント・レベルのどちらでも指定できます。一般最適化ガイドライン・エレメントの説明および構文は、グローバル最適化ガイドラインとステートメント・レベルの最適化ガイドラインのどちらでも同じです。詳しくは、『一般最適化ガイドライン用の XML スキーマ』を参照してください。

STMTPROFILE エレメントの XML スキーマ:

STMTPROFILE エレメントは、最適化プロファイル内のステートメント・プロファイルを定義します。

これは、複合タイプ statementProfileType によって定義されます。

XML Schema

```
<xs:complexType name="statementProfileType">
  <xs:sequence>
    <xs:element name="STMTMATCH" type="stmtMatchType" minOccurs="0"/>
    <xs:element name="STMTKEY" type="statementKeyType"/>
    <xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
  </xs:sequence>
  <xs:attribute name="ID" type="xs:string" use="optional"/>
</xs:complexType>
```

説明

ステートメント・プロファイルは、特定のステートメントの最適化ガイドラインを指定します。このプロファイルには、以下の部分が含まれています。

- ステートメントのマッチング

最適化プロファイル内のステートメントは、コンパイル済みのステートメント (複数可) と完全一致または不完全一致でマッチングされます。 `STMTMATCH` 属性の値は、どちらのマッチング方法が適用されるかを表します。

- ステートメント・キー

最適化プロファイルをアプリケーション内の複数のステートメントに対して有効にすることができます。最適化プロファイルはステートメント・キーを使用して、各ステートメント・プロファイルをアプリケーション内の対応するステートメントに自動的に突き合わせます。このマッチングにより、アプリケーションを編集しなくとも、ステートメントの最適化ガイドラインが提供されます。ステートメント・キーには、(アプリケーションで作成された) ステートメントのテキストと、適切なステートメントを明確に識別するのに必要なその他の情報が含まれています。 `STMTKEY` サブエレメントが、ステートメント・キーを表しています。

- ステートメント・レベルの最適化ガイドライン

ステートメント・プロファイルのこの部分は、ステートメント・キーによって識別されるステートメントに有効な最適化ガイドラインを指定します。詳しくは、『ステートメント・レベルの `OPTGUIDELINES` エレメントの XML スキーマ』を参照してください。

- ステートメント・プロファイル名

診断出力に表示されるユーザー指定の名前で、特定のステートメント・プロファイルを識別します。

***STMTMATCH* エレメント用の XML スキーマ:**

`STMTMATCH` エレメントを使用すると、最適化プロファイルまたはその中に含まれる特定のステートメントで使用されたマッチング方法が有効になります。

これは、複合タイプ `stmtMatchType` によって定義されます。

XML Schema

```
<xs:complexType name="stmtMatchType">
  <xs:attribute name="EXACT" type="boolean" use="optional" default="TRUE"/>
</xs:complexType>
```

説明

オプションの `EXACT` 属性により、マッチング方式が指定されます。この値が `TRUE` に設定されていると、完全一致突き合わせが適用されます。この値が `FALSE` に設定されていると、不完全一致突き合わせが適用されます。デフォルトの設定は完全一致突き合わせです。

例

次に示すのは、ステートメント・レベルでの `STMTMATCH` エレメントの定義例です。 `STMTKEY` エレメント内のステートメントに対して不完全一致突き合わせが有効になります。

```
<STMTPROFILE ID='S1'>
  <STMTMATCH EXACT='FALSE' />
  <STMTKEY>
```

```

        <![CDATA[select t1.c1, count(*) from t1,t2 where t1.c1 = t2.c1 and t1.c1 > 0]]>
    </STMTKEY>
    ...
</STMTPROFILE>

```

STMTKEY エlementの XML スキーマ:

STMTKEY Elementを使用すると、オプティマイザは、ステートメント・プロファイルをアプリケーション内の対応するステートメントと一致させられます。

これは、複合タイプ `statementKeyType` によって定義されます。

XML Schema

```

<xs:complexType name="statementKeyType" mixed="true">
  <xs:attribute name="SCHEMA" type="xs:string" use="optional"/>
  <xs:attribute name="FUNCPATH" type="xs:string" use="optional"/>
</xs:complexType>
</xs:schema>

```

説明

オプションの `SCHEMA` 属性を使用して、ステートメント・キーのデフォルトのスキーマ部分を指定することができます。

オプションの `FUNCPATH` 属性を使用して、ステートメント・キーの関数パス部分を指定することができます。複数のパスはコンマで区切る必要があります。指定する関数パスは、コンパイル・キーに指定されている関数パスと正確に一致していなければなりません。

例

以下の例は、特定のステートメントを、デフォルト・スキーマ 'COLLEGE' と関数パス 'SYSIBM,SYSFUN,SYSPROC,DAVE' に関連付けるステートメント・キー定義を示しています。

```

<STMTKEY SCHEMA='COLLEGE' FUNCPATH='SYSIBM,SYSFUN,SYSPROC,DAVE'>
  <![CDATA[select * from orders" where foo(orderkey) > 20]]>
</STMTKEY>

```

CDATA タグ (<![CDATA[で始まり、]]> で終わる) は、ステートメント・テキストに特殊な XML 文字「>」が含まれているために必要となります。

STMTMATCH Element用の XML スキーマ:

STMTMATCH Elementを使用すると、最適化プロファイルまたはその中に含まれる特定のステートメントで使用されたマッチング方法が有効になります。

これは、複合タイプ `stmtMatchType` によって定義されます。

XML Schema

```

<xs:complexType name="stmtMatchType">
  <xs:attribute name="EXACT" type="boolType" use="optional" default="TRUE"/>
</xs:complexType>

```

説明

オプションの `EXACT` 属性により、マッチング方式が指定されます。この値が `TRUE` に設定されていると、完全一致突き合わせが適用されます。この値が `FALSE`

に設定されていると、不完全一致突き合わせが適用されます。デフォルトの設定は完全一致突き合わせです。

例

次に示すのは、ステートメント・レベルでの `STMTMATCH` エレメントの定義例です。`STMTKEY` エレメント内のステートメントに対して不完全一致突き合わせが有効になります。

```
<STMTPROFILE ID='S1'>
  <STMTMATCH EXACT='FALSE' />
  <STMTKEY>
    <![CDATA[select t1.c1, count(*) from t1,t2 where t1.c1 = t2.c1 and t1.c1 > 0]]>
  </STMTKEY>
  ...
</STMTPROFILE>
```

ステートメント・キーおよびコンパイル・キーのマッチング:

ステートメント・キーは、ステートメント・レベルの最適化ガイドラインが適用されるアプリケーション・ステートメントを識別します。マッチングの方法は、最適化プロファイル内の `STMTMATCH` エレメントを使用して指定できます。

SQL ステートメントのコンパイル時には、ステートメントがコンパイラによってどのように意味的に解釈されるかについては、様々な要因が影響を及ぼします。

SQL ステートメントと、SQL コンパイラ・パラメーターの設定が一緒になって、コンパイル・キーを形成します。ステートメント・キーのそれぞれの部分は、コンパイル・キーの何らかの部分に対応します。

ステートメント・キーは、以下の部分で構成されます。

- ステートメント・テキスト。アプリケーションで作成されたとおりのステートメントのテキストです。
- デフォルト・スキーマ。非修飾の表名の暗黙的な修飾子として使用されるスキーマ名です。この部分はオプションですが、ステートメント内に非修飾の表名がある場合は指定する必要があります。
- 関数パス。非修飾関数およびデータ・タイプ参照を解決する際に使用する関数パスです。この部分はオプションですが、ステートメント内に非修飾のユーザー定義関数やユーザー定義タイプがある場合は指定する必要があります。

データ・サーバーでの SQL ステートメントのコンパイル時に、アクティブな最適化プロファイルが検出されると、その最適化プロファイルの各ステートメント・キーと現在のコンパイル・キーとのマッチングが試行されます。マッチングのタイプは、最適化プロファイルで完全一致突き合わせが指定されているか、不完全一致突き合わせが指定されているかによって決まります。最適化プロファイルで `STMTMATCH` エレメントを指定すると、どちらのタイプのマッチングを使用するかを指定できます。 `EXACT` 属性を `TRUE` または `FALSE` に設定することにより、完全一致突き合わせと不完全一致突き合わせのいずれかを有効にできます。`STMTMATCH` エレメントを指定しない場合は、自動的に完全一致突き合わせが有効になります。

完全一致突き合わせでは、ステートメント・キーの各指定部分が、コンパイル・キーの対応部分と一致する場合に、ステートメント・キーとコンパイル・キーは一致したことになります。ステートメント・キーの一部を指定しなかった場合、その省

略された部分はデフォルトで一致したものとみなされます。ステートメント・キーの各未指定部分は、ワイルドカードとして扱われて、任意のコンパイル・キーの対応部分にマッチングされます。

不完全一致突き合わせでは、ステートメント・キーのステートメント・テキストとコンパイル・キーのマッチングが行われるとき、リテラル、ホスト変数、およびパラメーター・マーカーは無視されます。

完全一致突き合わせと不完全一致突き合わせは、以下のように作動します。

- キーワードのマッチングでは、大/小文字を区別しない。例えば、`select` と `SELECT` は一致すると見なせます。
- 引用符で区切られていない ID のマッチングでは、大/小文字を区別しない。例えば、`T1` と `t1` は一致すると見なせます。
- 引用符で区切られている ID と引用符で区切られていない ID は一致すると見なせますが、例外が 1 つあります。例えば `T1` と `"T1"` は一致すると見なせます。また `t1` と `"T1"` も同様です。しかし、`t1` は `"t1"` と一致するとは見なされません。

`t1` と `"t1"` は一致すると見なされません。これは、`t1` と `"t1"` は別々の表を表すためです。引用符で区切られていない ID `t1` は大文字に変更され、表 `T1` を表しますが、引用符で区切られている ID `"t1"` は変更されず、表 `t1` を表します。この場合、この 2 つの ID が別々の表を表すため、大/小文字の区別が影響してきます。

データ・サーバーが、現行のコンパイル・キーと一致するステートメント・キーを検出すると、検索が停止します。ステートメント・キーが現在のコンパイル・キーに一致する複数のステートメント・プロファイルがある場合、そのようなステートメント・プロファイルのうちの最初のもの (文書の順序に基づいて) だけが使用されます。

不完全一致突き合わせ:

コンパイル中にアクティブな最適化プロファイルが存在する場合、コンパイル・ステートメントは最適化プロファイルの中のステートメントと、完全に、または不完全にマッチングされます。

不完全一致突き合わせは、コンパイルするステートメントと、最適化プロファイルの中のステートメントとの間で柔軟なマッチングを行うために使用します。不完全一致突き合わせは、コンパイルするステートメントを最適化プロファイルのステートメントとマッチングするときに、リテラル、ホスト変数、およびパラメーター・マーカーを無視します。したがって、述部にさまざまなリテラル値を持つ多くのさまざまなステートメントをコンパイルしても、ステートメントは一致します。例えば次のステートメントは不完全一致していますが、完全一致はしていません。

```
select c1 into :hv1 from t1 where c1 > 10
```

```
select c1 into :hv2 from t1 where c1 > 20
```

不完全一致突き合わせは、SQL ステートメントと XQuery ステートメントの両方に適用されます。しかし、SQL または XQuery のステートメントまたはステートメント・フラグメント (個々の列名も含む) を表す関数仮パラメーターとして渡されたストリング・リテラルは、不完全一致突き合わせされません。SQL ステートメント

で使用される XMLQUERY、XMLTABLE、および XMLEXISTS などの XML 関数は、完全一致突き合わせされます。ストリング・リテラルには、以下の内容を含めることができます。

- XQuery の中に SQL が組み込まれた、または SQL ステートメントの中に XQuery が組み込まれたステートメントの全体
- 列名などの識別子
- 検索パスを含む XML 式

XQuery の場合、不完全一致突き合わせはリテラルだけを無視します。以下のリテラルは、不完全一致突き合わせでは無視されます。ただし、ストリング・リテラルには若干の制限があります。

- 10 進リテラル
- 倍精度実数リテラル
- 整数リテラル
- db2-fn:sqlquery、db2-fn:xmlcolumn、db2-fn:xmlcolumn-contains 関数への入力パラメーター以外のストリング・リテラル

次の XQuery ステートメントは、不完全一致突き合わせが使用可能になっている場合に一致します。

```
xquery let $i:= db2-fn:sqlquery("select c1 from tab1")/a/b[c=1] return $i
```

```
xquery let $i:= db2-fn:sqlquery("select c1 from tab1")/a/b[c=2] return $i
```

不完全一致突き合わせの場合、特殊レジスターはサポートされません。次の例は、不完全一致突き合わせで一致しないいくつかのステートメント・タイプを示しています。

- c1 between 5 and :hv
5 between c1 and c2
- c1 in (select c1 from t1)
c1 in (1,2,3)
- c1 in (c1, 1, 2)
c1 in (c2, 1, 2)
- A = 5
A = 5 + :hv
- with RR
with RS
- c2 < CURRENT TIME
c2 < '11:12:40'
- c3 > CURRENT TIMESTAMP
c3 > '07/29/2010'

突き合わせを指定するための構文

最適化プロファイルの中で、完全一致突き合わせまたは不完全一致突き合わせをグローバル・レベルまたはステートメント・レベルで設定できます。XML 要素 STMTMATCH を使用すると、マッチングの方法を設定できます。

STMTMATCH エレメントには EXACT 属性があり、これを TRUE か FALSE のいずれかに設定できます。この値を TRUE に指定すると、完全一致突き合わせが有効になります。この値を FALSE に指定すると、不完全一致突き合わせが有効になり

ます。このエレメントを指定しない場合、または STMTMATCH エレメントのみを指定して EXACT 属性を指定しない場合は、デフォルトで完全一致突き合わせが有効になります。

最適化プロファイルに含まれるすべてのステートメントに突き合わせ方法を適用するには、トップの OPTPROFILE エレメントの直後に STMTMATCH エレメントをグローバル・レベルで指定します。

最適化プロファイルに含まれる特定のステートメントに突き合わせ方法を適用するには、STMTPROFILE エレメントの直後に STMTMATCH エレメントを指定します。

次の例は、グローバル・レベルとステートメント・レベルの両方の STMTMATCH エレメントがある最適化プロファイルを示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE>

  <!--Global section -->
    <STMTMATCH EXACT='FALSE' />

  <!-- Statement level profile -->

    <STMTPROFILE ID='S1'>
      <STMTMATCH EXACT='TRUE' />
      <STMTKEY>
        <![CDATA[select t1.c1, count(*) from t1,t2 where t1.c1 = t2.c1 and t1.c1 > 0]]>
      </STMTKEY>
      <OPTGUIDELINES>
        <NLJOIN>
          <TBSCAN TABLE='T1' />
          <TBSCAN TABLE='T2' />
        </NLJOIN>
      </OPTGUIDELINES>
    </STMTPROFILE>

    <STMTPROFILE ID='S2'>
      <STMTKEY><![CDATA[select * from T1 where c1 in( 10,20)]]>
      </STMTKEY>
      <OPTGUIDELINES>
        <REGISTRY>
          <OPTION NAME='DB2_REDUCED_OPTIMIZATION' VALUE='YES' />
        </REGISTRY>
      </OPTGUIDELINES>
    </STMTPROFILE>

  </OPTPROFILE>
```

優先順位

前述の例では、STMTMATCH エレメントがグローバル・レベルとステートメント・レベルの両方で設定されています。したがって、どのマッチング方式が実行されるかは、優先順位によって決まります。次に、優先順位を高いものから低いものの順で示します。

1. ステートメント・レベルのプロファイル設定
2. グローバル・レベルのプロファイル設定

前述の例では EXACT が FALSE に設定されているため、この順序は、不完全一致がグローバル・レベルで設定されていることを示します。したがって、コンパイル・ステートメントが最初のステートメントと一致しない限り、プロファイルに含まれるすべてのステートメントに不完全一致突き合わせが適用されます。次に、最初のステートメントの STMTMATCH エレメントの EXACT が TRUE に設定されているため、このステートメントでは完全一致突き合わせが有効になります。

このサンプル最適化プロファイル・ファイルの最後のステートメントには STMTMATCH エレメントがありません。ここではグローバル設定が優先となり、このステートメントには不完全一致突き合わせが適用されます。

最適化プロファイル内の SQL ステートメントでの不完全一致突き合わせの例:

最適化プロファイルでの不完全一致突き合わせは、STMTMATCH タグの EXACT 属性を FALSE に設定した場合に行われます。その場合、アクティブな最適化プロファイル内のステートメントに対してコンパイル・ステートメントのマッチングが行われます。コンパイラーは、さまざまなデフォルトのマッチング・ルール、および不完全一致突き合わせ固有のルールに基づいて、これらのステートメントをマッチングします。

次の例は、SQL ステートメントのマッチングで、不完全一致突き合わせが正常に行われたことを示しています。

例 1: 述部節での不完全一致突き合わせ

次に示すステートメント・フラグメントのそれぞれペアは、述部のリテラル値が異なっていますが、一致となります。

```
between '07/29/2010' and '08/29/2010'  
between '09/29/2010' and '10/29/2010'
```

```
'ab'      like :hv1  
'AYYANG' like :hv2
```

```
(A=1 AND B=1)      OR (A=2 AND B=2)  
(A=:hv1 AND B=:hv2) OR (A=3 AND B=3)
```

```
c1 > 0 selectivity 0.1  
c1 > 0 selectivity 0.9
```

```
c1 = ?  
c1 = :hv1
```

例 2: IN リスト述部での不完全一致突き合わせ

次に示すステートメント・フラグメントはすべて IN リスト述部の値が異なっていますが、一致となります。

```
c1 in (:hv1, :hv2, :hv3);
```

```
c1 in (:hv2, :hv3);
```

```
c1 in ( ?, ?, ?, ? );
```

```
c1 in (c1, c2 );
```

```
c1 in (:hv1, :hv2, c1, 1, 2, 3, c2, ?, ?);
```

例 3: 選択リストでの不完全一致突き合わせ

次のステートメント・フラグメントでは、選択リスト内のホスト変数が異なっていますが、一致となります。

```
select c1 into :hv1 from t1  
select c1 into :hv2 from t1
```

次のステートメント・フラグメントでは、選択リスト内のリテラルが異なっていますが、一致となります。

```
select 1, c1 from t1  
select 2, c1 from t1
```

次のステートメント・フラグメントでは、選択リスト内の副照会が異なっていますが、一致となります。

```
select c1, (select c1 from t1 where c2 = 0) from t1
select c1, (select c1 from t1 where c2 = 5) from t1
```

次のステートメント・フラグメントでは、選択リスト内の式が異なっていますが、一致となります。

```
select c1 + 1
select c1 + 2
```

例 4: 節が異なる場合の不完全一致突き合わせ

次のステートメント・フラグメントでは、`optimize` 節の行が異なっていますが、一致となります。

```
optimize for 1 row
optimize for 10 row
```

次のステートメント・フラグメントでは、`fetch` 節の行が異なっていますが、一致となります。

```
fetch first 10 rows only
fetch first 50 rows only
```

次のステートメント・フラグメントでは、`having` 節のリテラル値が異なっていますが、一致となります。

```
having c1 > 0
having c1 > 10
```

次に示すステートメント・フラグメントのそれぞれペアは、`order by` 節の列位置が異なっているか、`order by` 節の式のリテラル値が異なっていますが、一致となります。

```
order by 1,2,3
order by 4,5,6
```

```
order by c1+1, c2 + 2, 3
order by c1+2, c2 + 3, 4
```

次に示すステートメント・フラグメントのそれぞれのペアは、`set` 節のリテラル値またはホスト変数が異なっていますが、一致となります。

```
set c1 = 1
set c1 = 2
```

```
set queryno = 2
set queryno = 3
```

```
set querytag = 'query1'
set querytag = 'query2'
```

```
SET :HV00001 :HI00001 = <subquery>
SET :HV00009 :HI00009 = <subquery>
```

次に示すステートメント・フラグメントのそれぞれペアは、`group by` 節のリテラル値が異なっていますが、一致となります。

```
group by c1 + 1
group by c1 + 2
```

```
group by 1,2,3
group by 3,2,1
```

次に示すステートメント・フラグメントのそれぞれペアは、values 節のリテラル値が異なっていますが、一致となります。

```
values 1,2,3
values 3,4,5

values ( 10, 'xml', 'a' )
values ( 20, 'xml', 'ab' )
```

例 5: 非 XML 関数での不完全一致突き合わせ

次に示すステートメント・フラグメントのそれぞれペアは、関数呼び出しのリテラル値が異なっていますが、リテラルの数が同じであるため、一致となります。

```
decimal(c1, 5, 2)
decimal(c1, 9, 3)

Blob('%abc%')
Blob('cde%')

max(1, 100, 200)
max(INFINITY, NAN, SNAN)
```

例 6: 特殊な式での不完全一致突き合わせ

次に示すステートメント・フラグメントのそれぞれペアは、case when 式または mod 式のリテラル値が異なっていますが、一致となります。

```
order by mod(c1, 2)
order by mod(c1, 4)

case when b1 < 10 then 'A' else 'B' end
case when b1 < 20 then 'C' else 'D' end
```

ステートメント・レベルの OPTGUIDELINES エレメントの XML スキーマ:

ステートメント・プロファイルの OPTGUIDELINES エレメントは、関連したステートメント・キーによって識別されるステートメントに有効な最適化ガイドラインを定義します。これは、タイプ optGuidelinesType によって定義されます。

XML Schema

```
<xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
<xs:complexType name="optGuidelinesType">
  <xs:sequence>
    <xs:group ref="general request" minOccurs="0" maxOccurs="1"/>
    <xs:choice maxOccurs="unbounded">
      <xs:group ref="rewriteRequest"/>
      <xs:group ref="accessRequest"/>
      <xs:group ref="joinRequest"/>
      <xs:group ref="mqtEnforcementRequest"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

説明

optGuidelinesType グループは、OPTGUIDELINES エレメントの一連の有効なサブエレメントを定義します。それぞれのサブエレメントは、DB2 オプティマイザーによって最適化ガイドラインとして解釈されます。サブエレメントは、一般要求エレメント、書き直し要求エレメント、アクセス要求エレメント、または結合要求エレメントとして分類できます。

- 一般要求エレメント を使用して、オブティマイザーの検索スペースを変更できる一般最適化ガイドラインを指定します。
- 書き直し要求エレメント を使用して、照会書き直し最適化ガイドラインを指定します。このガイドラインは、最適化ステートメントが判別される際に適用される照会トランスフォーメーションに作用させることができます。
- アクセス要求エレメント および結合要求エレメント はプラン最適化ガイドラインです。このガイドラインは、最適化ステートメントの実行プランで使用されるアクセス方式、結合方式、および結合順序に作用するために使用できます。
- *MQT* 強制要求エレメント は、コストの見積もりに関係なくアクセス・プランでの使用が強制される、意味的に一致可能なマテリアライズ照会表 (*MQT*) を指定します。

注: ステートメント・プロファイルで指定される最適化ガイドラインは、最適化プロファイルのグローバル・セクションで指定される最適化ガイドラインよりも優先します。

一般最適化ガイドライン用の XML スキーマ:

`generalRequest` グループは、最適化プロセスの特定のフェーズに限定されないガイドラインを定義します。このガイドラインを使用して、オブティマイザーの検索スペースを変更することができます。

```
<!--***** --> \
<!-- Choices of general request elements. --> \
<!-- REOPT can be used to override the setting of the REOPT bind option. --> \
<!-- DPFXMLMOVEMENT can be used to affect the optimizer's plan when moving XML documents --> \
<!-- between database partitions. The allowed value can be REFERENCE or COMBINATION. The --> \
<!-- default value is NONE. --> \
<!--***** --> \
<xs:group name="generalRequest">
  <xs:sequence>
    <xs:element name="REOPT" type="reoptType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="DEGREE" type="degreeType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="QRYOPT" type="qryoptType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RTS" type="rtsType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="DPFXMLMOVEMENT" type="dpfXMLMovementType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="REGISTRY" type="registryType" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:group>
```

一般最適化ガイドラインは、グローバルおよびステートメント・レベルのどちらでも指定できます。一般最適化ガイドライン・エレメントの説明および構文は、グローバル最適化ガイドラインとステートメント・レベルの最適化ガイドラインのどちらでも同じです。

説明

一般要求エレメントは、最適化検索スペースに影響する一般最適化ガイドラインを定義します。最適化検索スペースに影響するということは、書き直しやコスト・ベースの最適化ガイドラインの適用度にも影響するということです。

DEGREE 要求:

DEGREE 一般要求エレメントを使用して、**DEGREE** バインド・パラメーターの設定、**dft_degree** データベース構成パラメーターの値、または前の **SET CURRENT DEGREE** ステートメントの結果をオーバーライドすることができます。

DEGREE 一般要求エレメントが考慮されるのは、インスタンスがパーティション内並列処理用に構成されている場合だけです。その他の場合には、警告が戻ります。これは、複合タイプ `degreeType` によって定義されます。

XML Schema

```
<xs:simpleType name="intStringType">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:minInclusive value="1"></xs:minInclusive>
        <xs:maxInclusive value="32767"></xs:maxInclusive>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="ANY"/>
        <xs:enumeration value="-1"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

<xs:complexType name="degreeType">
  <xs:attribute name="VALUE"
    type="intStringType"></xs:attribute>
</xs:complexType>
```

説明

DEGREE 一般要求エレメントの必須の `VALUE` 属性は、DEGREE オプションの設定を指定します。この属性は、1 から 32 767 までの整数値か、ストリング値 `-1` または `ANY` をとることができます。値 `-1` (または `ANY`) は、並列処理の度合いがデータ・サーバーによって決定されることを指定します。値 `1` は、パーティション内並列処理を照会で使用しないことを指定します。

DPFXMLMOVEMENT 要求:

DPFXMLMOVEMENT 一般要求エレメントは、パーティション・データベース環境において、パーティション・データベース間でタイプ XML の列を移動する、またはその列の参照のみを移動するプランの選択に関し、オブティマイザーの決定をオーバーライドするために使用できます。これは、複合タイプ `dpfXMLMovementType` によって定義されます。

```
<xs:complexType name="dpfXMLMovementType">
  <xs:attribute name="VALUE" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="REFERENCE"/>
        <xs:enumeration value="COMBINATION"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

説明

パーティション・データベース環境では、ステートメントの実行中にデータベース・パーティション間で、データを時々移動する必要があります。XML 列の場合、

オブティマイザーは、それらの列に含まれている実際の文書を移動するか、それとも元のデータベース・パーティション上のソース文書への参照だけを移動するかを選択できます。

DPFXMLMOVEMENT 一般要求エレメントには必須の **VALUE** 属性があり、それに指定可能な値は **REFERENCE** または **COMBINATION** です。XML 列を含む行を 1 つのデータベース・パーティションから別のパーティションに移動する必要がある場合には、以下のようにします。

- **REFERENCE** は、XML 文書への参照が表キュー (TQ) 演算子を介して移動されるように指定します。文書自体は、ソース・データベース・パーティションに残っています。
- **COMBINATION** は、一部の XML 文書が移動され、残りの XML 文書は参照のみが TQ 演算子を介して移動されることを指定します。

文書を移動するのか、それとも単にそれらの文書への参照を移動するのかについての決定は、照会が実行されるときに優先される条件に依存しています。

DPFXMLMOVEMENT 一般要求エレメントが指定されていない場合、オブティマイザーはパフォーマンスを最大にすることを意図したコスト・ベースの決定を行います。

QRYOPT 要求:

QRYOPT 一般要求エレメントを使用して、**QUERYOPT** バインド・パラメーターの設定、**dft_queryopt** データベース構成パラメーターの値、または前の **SET CURRENT QUERY OPTIMIZATION** ステートメントの結果をオーバーライドすることができます。これは、複合タイプ **qryoptType** によって定義されます。

XML Schema

```
<xs:complexType name="qryoptType">
  <xs:attribute name="VALUE" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="0"/>
        <xs:enumeration value="1"/>
        <xs:enumeration value="2"/>
        <xs:enumeration value="3"/>
        <xs:enumeration value="5"/>
        <xs:enumeration value="7"/>
        <xs:enumeration value="9"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

説明

QRYOPT 一般要求エレメントの必須の **VALUE** 属性は、**QUERYOPT** オプションの設定を指定します。この属性は、値 0、1、2、3、5、7、または 9 のいずれかを取ることができます。こうした値が何を表すかについての詳細は、『最適化クラス』を参照してください。

REOPT 要求:

REOPT 一般要求エレメントを使用して、**REOPT** バインド・パラメーターの設定をオーバーライドすることができます。このオプションは、パラメーター・マーカまたはホスト変数が含まれるステートメントの最適化に作用します。これは、複合タイプ `reoptType` によって定義されます。

XML Schema

```
<xs:complexType name="reoptType">
  <xs:attribute name="VALUE" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="ONCE"/>
        <xs:enumeration value="ALWAYS"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

説明

REOPT 一般要求エレメントの必須の `VALUE` 属性は、REOPT オプションの設定を指定します。この属性が取ることができる値は、`ONCE` または `ALWAYS` です。`ONCE` は、最初の一連のホスト変数またはパラメーター・マーカ値用にステートメントを最適化することを指定します。`ALWAYS` は、一連のホスト変数またはパラメーター・マーカ値ごとに、ステートメントを最適化することを指定します。

REGISTRY 要求:

REGISTRY エレメントは最適化プロファイルにレジストリー変数を設定するために使用します。REGISTRY エレメントには `OPTION` エレメントが組み込まれており、レジストリー変数はここで設定されます。

REGISTRY エレメントは複合タイプ `registryType` によって定義され、`OPTION` エレメントは複合タイプ `genericOptionType` によって定義されます。

XML Schema

```
<xs:complexType name="registryType">
  <xs:sequence>
    <xs:element name="OPTION" type="genericOptionType"
      minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="genericOptionType">
  <xs:attribute name="NAME" type="xs:string" use="required"/>
  <xs:attribute name="VALUE" type="xs:string" use="required"/>
</xs:complexType>
```

説明

REGISTRY エレメントは、レジストリー変数をグローバル・レベル、ステートメント・レベル、またはその両方で設定します。REGISTRY エレメントに組み込まれた `OPTION` エレメントには、`NAME` 属性と `VALUE` 属性があります。これらの属性によって、プロファイルまたはその中の特定のステートメントに適用されるレジストリー変数の名前と値が指定されます。

RTS 要求:

RTS 一般要求エレメントを使用して、リアルタイム統計収集を有効または無効にすることができます。また、リアルタイム統計収集が行われる時間を制限するためにも使用できます。

特定の照会またはワークロードでは、ステートメント・コンパイル時に余分なオーバーヘッドが生じないようにするために、リアルタイム統計収集を制限することが望ましい場合もあります。RTS 一般要求エレメントは、複合タイプ `rtsType` によって定義されます。

```
<!--*****--> \
<!-- RTS general request element to enable, disable or provide a time budget for --> \
<!-- real-time statistics collection. --> \
<!-- OPTION attribute allows enabling or disabling real-time statistics. --> \
<!-- TIME attribute provides a time budget in milliseconds for real-time statistics collection.--> \
<!--*****--> \
<xs:complexType name="rtsType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
  <xs:attribute name="TIME" type="xs:nonNegativeInteger" use="optional"/>
</xs:complexType>
```

説明

RTS 一般要求エレメントには、2 つのオプション属性があります。

- **OPTION** 属性は、リアルタイム統計収集を有効または無効にするために使用します。可能な値は、ENABLE (デフォルト) または DISABLE です。
- **TIME** 属性は、ステートメント・コンパイル時に (各ステートメントごとに) リアルタイム統計収集に費やす最大時間をミリ秒単位で指定します。

OPTION 属性で ENABLE を指定する場合、対応する構成パラメーターで自動統計収集およびリアルタイム統計を有効にする必要があります。これらを有効にしなかった場合、最適化ガイドラインが適用されず、理由コード 13 の SQL0437W が戻ります。

照会書き直し最適化ガイドラインの XML スキーマ:

`rewriteRequest` グループは、最適化プロセスの照会書き直しフェーズに作用するガイドラインを定義します。

XML Schema

```
<xs:group name="rewriteRequest">
  <xs:sequence>
    <xs:element name="INLIST2JOIN" type="inListToJoinType" minOccurs="0"/>
    <xs:element name="SUBQ2JOIN" type="subqueryToJoinType" minOccurs="0"/>
    <xs:element name="NOTEX2AJ" type="notExistsToAntiJoinType" minOccurs="0"/>
    <xs:element name="NOTIN2AJ" type="notInToAntiJoinType" minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

説明

INLIST2JOIN エレメントを使用してステートメント・レベルおよび述部レベルの両方の最適化ガイドラインを指定する場合、述部レベルのガイドラインがステートメント・レベルのガイドラインをオーバーライドします。

IN-LIST から結合への照会書き直し要求:

INLIST2JOIN 照会書き直し要求エレメントを使用すると、IN-LIST 述部から結合への書き直しトランスフォーメーションを有効または無効にできます。これは、ステートメント・レベルの最適化ガイドラインまたは述部レベルの最適化ガイドラインとして指定することができます。述部レベルの最適化ガイドラインの場合、照会ご

とに 1 つのガイドラインのみを有効にできます。INLIST2JOIN 要求エレメントは、複合タイプ `inListToJoinType` によって定義されます。

XML Schema

```
<xs:complexType name="inListToJoinType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
  <xs:attribute name="TABLE" type="xs:string" use="optional"/>
  <xs:attribute name="COLUMN" type="xs:string" use="optional"/>
</xs:complexType>
```

説明

INLIST2JOIN 照会書き直し要求エレメントには、3 つのオプション属性がありますが、サブエレメントはありません。OPTION 属性が取れるのは、値 ENABLE (デフォルト) または DISABLE です。TABLE 属性と COLUMN 属性は、IN-LIST 述部を指定するために使用します。これらの属性を指定しないと、または空ストリング ("") 値を指定すると、ガイドラインはステートメント・レベルのガイドラインとして処理されます。これらの属性の片方または両方を指定すると、述部レベルのガイドラインとして処理されます。TABLE 属性を指定しない場合、または空ストリング値を指定する場合、COLUMN 属性を指定すると、最適化ガイドラインは無視されて、理由コード 13 で SQL0437W が戻ります。

NOT-EXISTS からアンチ結合への照会書き直し要求:

NOTEX2AJ 照会書き直し要求エレメントを使用すると、NOT-EXISTS 述部からアンチ結合への書き直しトランスフォーメーションを有効または無効にできます。これは、ステートメント・レベルの最適化ガイドラインとしてのみ指定することができます。NOTEX2AJ 要求エレメントは、複合タイプ `notExistsToAntiJoinType` によって定義されます。

XML Schema

```
<xs:complexType name="notExistsToAntiJoinType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
```

説明

NOTEX2AJ 照会書き直し要求エレメントには、1 つのオプション属性がありますが、サブエレメントはありません。OPTION 属性が取れるのは、値 ENABLE (デフォルト) または DISABLE です。

NOT-IN からアンチ結合への照会書き直し要求:

NOTIN2AJ 照会書き直し要求エレメントを使用すると、NOT-IN 述部からアンチ結合への書き直しトランスフォーメーションを有効または無効にできます。これは、ステートメント・レベルの最適化ガイドラインとしてのみ指定することができます。NOTIN2AJ 要求エレメントは、複合タイプ `notInToAntiJoinType` によって定義されます。

XML Schema

```
<xs:complexType name="notInToAntiJoinType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
```

説明

NOTIN2AJ 照会書き直し要求エレメントには、1 つのオプション属性がありますが、サブエレメントはありません。OPTION 属性が取れるのは、値 ENABLE (デフォルト) または DISABLE です。

副照会から結合への照会書き直し要求:

SUBQ2JOIN 照会書き直し要求エレメントを使用すると、副照会から結合への書き直しトランスフォーメーションを有効または無効にできます。これは、ステートメント・レベルの最適化ガイドラインとしてのみ指定することができます。

SUBQ2JOIN 要求エレメントは、複合タイプ `subqueryToJoinType` によって定義されます。

XML Schema

```
<xs:complexType name="subqueryToJoinType">
  <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
</xs:complexType>
```

説明

SUBQ2JOIN 照会書き直し要求エレメントには、1 つのオプション属性がありますが、サブエレメントはありません。OPTION 属性が取れるのは、値 ENABLE (デフォルト) または DISABLE です。

プラン最適化ガイドライン用の XML スキーマ:

プラン最適化ガイドラインは、アクセス要求または結合要求で構成できます。

- **アクセス要求** は、表参照のアクセス方式を指定します。
- **結合要求** は、結合操作を実行する方式とシーケンスを指定します。結合要求は、その他のアクセス要求または結合要求で構成されます。

使用可能なアクセス要求の大半は、例えば、表スキャン、索引スキャン、およびリスト・プリフェッチなどの、オプティマイザーのデータ・アクセス方式に対応します。使用可能な結合要求の大半は、ネスト・ループ結合、ハッシュ結合、およびマージ結合などの、オプティマイザーの結合方式に対応します。各アクセス要求または結合要求エレメントは、プラン最適化に作用するために使用できます。

アクセス要求:

`accessRequest` グループは、有効な一連のアクセス要求エレメントを定義します。アクセス要求は、表参照のアクセス方式を指定します。

XML Schema

```
<xs:group name="accessRequest">
  <xs:choice>
    <xs:element name="TBSCAN" type="tableScanType"/>
    <xs:element name="IXSCAN" type="indexScanType"/>
    <xs:element name="LPREFETCH" type="listPrefetchType"/>
    <xs:element name="IXAND" type="indexAndingType"/>
    <xs:element name="IXOR" type="indexOringType"/>
    <xs:element name="XISCAN" type="indexScanType"/>
  </xs:choice>
</xs:group>
```

```

        <xs:element name="XANDOR" type="XANDORType"/>
        <xs:element name="ACCESS" type="anyAccessType"/>
    </xs:choice>
</xs:group>

```

説明

- TBSCAN、IXSCAN、LPREFETCH、IXAND、IXOR、XISCAN、および XANDOR

上記の要素は、DB2 データ・アクセス方式に対応しますが、ステートメント内で参照されるローカル表にのみ適用されます。これらの要素は、ニックネーム (リモート表) または派生表 (副選択の結果) を参照することはできません。

- ACCESS

オプティマイザーがアクセス方式を選択するようにするこの要素は、結合順序 (アクセス方式ではない) が主要な関心事である際に使用できます。ターゲット表参照が派生表であるときは、ACCESS 要素を使用する必要があります。XML 照会の場合、この要素は、属性 TYPE = XMLINDEX を指定して使用し、オプティマイザーが XML 索引アクセス・プランを選択するように指定することもできます。

アクセスのタイプ:

TBSCAN、IXSCAN、LPREFETCH、IXAND、IXOR、XISCAN、XANDOR、および ACCESS 要素に共通する局面は、抽象タイプ `accessType` によって定義されます。

XML Schema

```

<xs:complexType name="accessType" abstract="true">
  <xs:attribute name="TABLE" type="xs:string" use="optional"/>
  <xs:attribute name="TABID" type="xs:string" use="optional"/>
  <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
  <xs:attribute name="SHARING" type="optionType" use="optional"
    default="ENABLE"/>
  <xs:attribute name="WRAPPING" type="optionType" use="optional"
    default="ENABLE"/>
  <xs:attribute name="THROTTLE" type="optionType" use="optional"/>
  <xs:attribute name="SHARESPEED" type="shareSpeed" use="optional"/>
</xs:complexType>

<xs:complexType name="extendedAccessType">
  <xs:complexContent>
    <xs:extension base="accessType">
      <xs:sequence minOccurs="0">
        <xs:element name="INDEX" type="indexType" minOccurs="2"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      <xs:attribute name="TYPE" type="xs:string" use="optional"
        fixed="XMLINDEX"/>
      <xs:attribute name="ALLINDEXES" type="boolType" use="optional"
        fixed="TRUE"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

説明

すべてのアクセス要求エレメントは、複合タイプ `accessType` を拡張します。そのようなエレメントはいずれも、`TABLE` または `TABID` 属性を使用して、ターゲット表参照を指定する必要があります。アクセス要求エレメントから正しい表参照を作成する方法については、『最適化ガイドラインでの表参照の形成』を参照してください。

アクセス要求エレメントは、オプションの `FIRST` 属性を指定することもできます。`FIRST` 属性を指定する場合、その値は `TRUE` でなければなりません。アクセス要求エレメントに `FIRST` 属性を追加すると、実行プランには、対応する `FROM` 節の結合シーケンス内の最初の表として指定の表を組み込む必要があることを示します。`FIRST` 属性を指定できるアクセスまたは結合要求は、`FROM` 節あたり 1 つだけです。1 つの同じ `FROM` 節の表をターゲットとする複数のアクセスまたは結合要求が `FIRST` 属性を指定した場合、そのような要求のうちの最初のもの以外はすべて無視されて、警告 (SQL0437W と理由コード 13) が戻ります。

新しい最適化ガイドラインを使用すると、コンパイラーのスキャン・シェアリングの決定に影響を与えることができます。コンパイラーが共有スキャン、ラッピング・スキャン、またはスロットル調整を許可していた場合、適切なガイドラインを指定することによって、共有スキャン、ラッピング・スキャン、またはスロットル調整を回避できます。共有スキャンは、スキャン・シェアリングに加わっている他のスキャンから参照でき、その共有スキャンの情報に基づいて他のスキャンは特定の決定を下せます。ラッピング・スキャンは、表内の任意のポイントで開始して、既にバッファ・プール内にあるページを活用できます。共有の全体レベルを増すために、スロットル・スキャンは遅延しています。

有効な `optionType` 値 (`SHARING`、`WRAPPING`、および `THROTTLE` 属性) は `DISABLE` と `ENABLE` (デフォルト) です。`SHARING` および `WRAPPING` は、コンパイラーがそれらを使用不可にするよう選択するとき、使用できません。`ENABLE` の使用は、それらの場合には、効果がありません。`THROTTLE` は有効または無効にすることができます。有効な `SHARESPEED` 値 (コンパイラーのスキャン速度の見積もりをオーバーライドするための値) は `FAST` および `SLOW` です。デフォルトを使用すると、コンパイラーはその見積もりに基づいて値を判別できます。

`TYPE` 属性の唯一サポートされている値は `XMLINDEX` です。これは、`IXAND`、`IXOR`、`XANDOR`、または `XISCAN` などの XML 索引アクセス方式の 1 つを使用して表にアクセスする必要があることを、オプティマイザーに示します。この属性を指定しないと、指定された表のアクセス・プランを選択するときに、オプティマイザーはコスト・ベースの決定をします。

オプションの `INDEX` 属性を使用して、索引名を指定することができます。

オプションの `INDEX` エレメントを使用して、2 つ以上の索引の名前を索引エレメントとして指定できます。`INDEX` 属性と `INDEX` エレメントが両方とも指定されている場合、`INDEX` 属性は無視されます。

オプションの `ALLINDEXES` 属性 (唯一のサポートされている値は `TRUE`) は、`TYPE` 属性に `XMLINDEX` の値がある場合にのみ、指定できます。`ALLINDEXES`

属性が指定されている場合、オプティマイザーは、指定された表へのアクセスに、コストに関係なく、すべての適用可能なリレーショナル索引および XML データの索引を使用する必要があります。

任意のアクセス要求:

ACCESS アクセス要求エレメントを使用して、オプティマイザーが表へのアクセスに適した方式をコスト・ベースで選択するように要求することができますし、派生表を参照する際にはこのエレメントを使用しなければなりません。派生表は、別の副選択の結果です。以下のアクセス要求エレメントは、複合タイプ anyAccessType によって定義されます。

XML Schema

```
<xs:complexType name="anyAccessType">
  <xs:complexContent>
    <xs:extension base="extendedAccessType"/>
  </xs:complexContent>
</xs:complexType>
```

説明

複合タイプ anyAccessType は、抽象タイプ extendedAccessType の単なる拡張です。新規のエレメントや属性は追加されていません。

TYPE 属性 (唯一サポートされている値は XMLINDEX) は、IXAND、IXOR、XANDOR、または XISCAN などの XML 索引アクセス方式の 1 つを使用して表にアクセスする必要があることを、オプティマイザーに示します。この属性を指定しないと、指定された表のアクセス・プランを選択するときに、オプティマイザーはコスト・ベースの決定をします。

TYPE 属性の値が XMLINDEX の場合にのみ、オプションの INDEX 属性を使用して索引名を指定できます。この属性が指定されている場合、オプティマイザーは以下のいずれかのプランを選択することができます。

- 指定された XML データの索引を使用する XISCAN プラン
- XANDOR プラン (指定する XML データの索引が XANDOR の下の索引の 1 つになる)。オプティマイザーは、XANDOR プラン内の適用可能なすべての XML データの索引を使用します。
- IXAND プラン (指定された索引が IXAND の先行索引となる)。オプティマイザーは、コスト・ベース方式で IXAND プランにさらに索引を追加します。
- コスト・ベースの IXOR プラン

TYPE 属性に XMLINDEX の値がある場合にのみ、オプションの INDEX エレメントを使用して、索引の 2 つ以上の名前を索引エレメントとして指定できます。このエレメントが指定されている場合、オプティマイザーは以下のプランのいずれかを選択することができます。

- XANDOR プラン (指定された XML データの索引が XANDOR の下に現れる)。オプティマイザーは、XANDOR プラン内の適用可能なすべての XML データの索引を使用します。
- IXAND プラン (指定された索引は、指定された順序で、IXAND の索引となる)
- コスト・ベースの IXOR プラン

INDEX 属性と INDEX エレメントが両方とも指定されている場合、INDEX 属性は無視されます。

オプションの ALLINDEXES 属性 (唯一のサポートされている値は TRUE) は、TYPE 属性に XMLINDEX の値がある場合にのみ、指定できます。この属性が指定されている場合、オプティマイザーは、指定された表へのアクセスに、コストに関係なく、すべての適用可能なリレーショナル索引および XML データの索引を使用する必要があります。オプティマイザーは以下のプランのいずれかを選択します。

- すべての適用可能な XML データの索引が XANDOR 演算子の下に現れる XANDOR プラン
- すべての適用可能なリレーショナル索引および XML データの索引が IXAND 演算子の下に現れる IXAND プラン
- IXOR プラン
- XISCAN プラン (単一の索引のみが表に定義されており、その索引が XML タイプである場合に限る)

例

以下のガイドラインでは、任意のアクセス要求の例を示します。

```
<OPTGUIDELINES>
  <HSJOIN>
    <ACCESS TABLE='S1' />
    <IXSCAN TABLE='PS1' />
  </HSJOIN>
</OPTGUIDELINES>
```

以下の例は、SECURITY 表への XML 索引アクセスを使用することを指定する、ACCESS ガイドラインを示しています。オプティマイザーは、XISCAN、IXAND、XANDOR、または IXOR プランなどの任意の XML 索引プランを選出できます。

```
SELECT * FROM security
  WHERE XMLEXISTS('$SDOC/Security/SecurityInformation/
    StockInformation[Industry= "OfficeSupplies"]')
```

```
<OPTGUIDELINES>
  <ACCESS TABLE='SECURITY' TYPE='XMLINDEX' />
</OPTGUIDELINES>
```

以下の例は、SECURITY 表へのすべての可能な索引アクセスが使用されることを指定する、ACCESS ガイドラインを示しています。選択の方法は、オプティマイザーに任せられています。2 つの XML 索引 SEC_INDUSTRY および SEC_SYMBOL が 2 つの XML 述部と一致しているとします。オプティマイザーは、コスト・ベース方式を使用して、XANDOR または IXAND アクセス方式を選択します。

```
SELECT * FROM security
  WHERE XMLEXISTS('$SDOC/Security/SecurityInformation/
    StockInformation[Industry= "Software"]') AND
    XMLEXISTS('$SDOC/Security/Symbol[.="IBM"]')
```

```
<OPTGUIDELINES>
  <ACCESS TABLE='SECURITY' TYPE='XMLINDEX' ALLINDEXES='TRUE' />
</OPTGUIDELINES>
```

以下の例は、SECURITY 表が少なくとも SEC_INDUSTRY XML 索引を使用してアクセスされることを指定する、ACCESS ガイドラインを示しています。最適マイザーは、コスト・ベース方式で、以下のアクセス・プランのいずれかを選択します。

- SEC_INDUSTRY XML 索引を使用する XISCAN プラン
- SEC_INDUSTRY 索引を IXAND の最初のレグとする IXAND プラン。最適マイザーは、コスト・ベース分析に従って、IXAND プランでさらに多くのリレーショナル索引または XML 索引を自由に使用します。例えば、リレーショナル索引が TRANS_DATE 列で使用可能な場合、最適マイザーがこの索引を有ると見なすなら、これが IXAND の追加のレグとして現れます。
- SEC_INDUSTRY 索引および他の適用可能な XML 索引を使用する XANDOR プラン

```
SELECT * FROM security
  WHERE trans_date = CURRENT DATE AND
        XMLEXISTS('$SDOC/Security/SecurityInformation/
  StockInformation[Industry= "Software"']) AND
        XMLEXISTS('$SDOC/Security/Symbol[.="IBM"']')

<OPTGUIDELINES>
  <ACCESS TABLE='SECURITY' TYPE='XMLINDEX' INDEX='SEC_INDUSTRY' />
</OPTGUIDELINES>
```

索引 ANDing アクセス要求:

IXAND アクセス要求エレメントを使用して、最適マイザーがローカル表にアクセスする際に索引 ANDing データ・アクセス方式を使用するように指定できます。これは、複合タイプ indexAndingType によって定義されます。

XML Schema

```
<xs:complexType name="indexAndingType">
  <xs:complexContent>
    <xs:extension base="extendedAccessType">
      <xs:sequence minOccurs="0">
        <xs:element name="NLJOIN" type="nestedLoopJoinType" minOccurs="1"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="STARJOIN" type="boolType" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

説明

複合タイプ indexAndingType は、extendedAccessType の拡張です。STARJOIN 属性および NLJOIN エレメントが指定されない場合、indexAndingType は extendedAccessType の単なる拡張になります。extendedAccessType タイプは、オプションの INDEX 属性、オプションの INDEX サブエレメント、オプションの TYPE 属性、およびオプションの ALLINDEXES 属性を追加して、抽象タイプ accessType を拡張します。INDEX 属性を使用して、索引の ANDing 操作で使用される最初の索引を指定することができます。INDEX 属性を使用すると、最適マイザーはコスト・ベース方式で追加の索引およびアクセス・シーケンスを選択します。INDEX サブエレメントを使用して、正確な一連の索引およびアクセス・シーケンスを指定することができます。INDEX サブエレメントの出現順は、個々の索引スキュンを実行すべき順序を示しています。INDEX サブエレメントを指定すると、それによって INDEX 属性の指定が置き換えられます。

- 索引を指定しないと、オプティマイザーは索引およびアクセス・シーケンスを両方ともコスト・ベース方式で選択します。
- 属性またはサブエレメントを使用して索引を指定する場合、その索引は TABLE または TABID 属性で識別される表で定義されている索引を識別する必要があります。
- 定義された索引が表にない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。

TYPE 属性 (唯一サポートされている値は XMLINDEX) は、XML データの 1 つ以上の索引を使用して表にアクセスする必要があることを、オプティマイザーに示します。

TYPE 属性の値が XMLINDEX の場合にのみ、オプションの INDEX 属性を使用して XML 索引名を指定できます。リレーショナル索引は、TYPE 属性の指定に関係なく、オプションの INDEX 属性で指定できます。指定された索引は、IXAND プランの先行索引として、オプティマイザーによって使用されます。オプティマイザーは、コスト・ベース方式で IXAND プランにさらに索引を追加します。

TYPE 属性の値が XMLINDEX の場合にのみ、オプションの INDEX エレメントを使用して、XML データの索引の 2 つ以上の名前を索引エレメントとして指定できます。リレーショナル索引は、TYPE 属性の指定に関係なく、オプションの INDEX エレメントで指定できます。指定された索引は、指定された順序で IXAND プランの索引として、オプティマイザーによって使用されます。

TYPE 属性が存在していない場合、INDEX 属性および INDEX エレメントは、リレーショナル索引で有効なままです。

INDEX 属性と INDEX エレメントが両方とも指定されている場合、INDEX 属性は無視されます。

オプションの ALLINDEXES 属性 (唯一のサポートされている値は TRUE) は、TYPE 属性に XMLINDEX の値がある場合にのみ、指定できます。この属性が指定されている場合、オプティマイザーは、指定された表へのアクセスに、コストに関係なく、IXAND プラン内のすべての適用可能なリレーショナル索引および XML データの索引を使用する必要があります。

TYPE 属性が指定されていても、INDEX 属性、INDEX エレメント、または ALLINDEXES 属性がいずれも指定されていない場合、オプティマイザーは、少なくとも 1 つの XML データの索引を持つ IXAND プランを選択します。プラン内の他の索引は、リレーショナル索引か、XML データの索引である可能性があります。索引の順序および選択は、コスト・ベースの方法でオプティマイザーによって決定されます。

索引 ANDing アクセス要求では、レコード索引の前にブロック索引が置かれている必要があります。この要件が満たされないと、理由コード 13 の SQL0437W が戻ります。索引 ANDing アクセス方式では、索引付けが可能な述部が各索引に少なくとも 1 つ必要です。必須の述部が存在しないために索引 ANDing が適格ではない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。ス

テートメントに関して有効な検索スペース内に索引 ANDing データ・アクセス方式がない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。

IXAND アクセス要求エレメントを使用して、スター型結合の索引 ANDing プランを要求することができます。IXAND エレメントのオプションの STARJOIN 属性は、IXAND がスター型結合の索引 ANDing プラン用であることを指定します。NLJOIN は IXAND のサブエレメントになることができ、適切に構成されたスター型結合の半結合でなければなりません。STARJOIN="FALSE" は、通常の基本アクセスの索引 ANDing プランの要求を指定します。STARJOIN="TRUE" は、スター型結合の索引 ANDing プランの要求を指定します。デフォルト値はコンテキストによって決まります。IXAND に 1 つ以上の半結合の子エレメントがある場合、デフォルトは TRUE になり、それ以外の場合、デフォルトは FALSE になります。STARJOIN="TRUE" が指定される場合、次のようになります。

- INDEX、TYPE、および ALLINDEXES 属性を指定できません
- INDEX エレメントを指定できません

NLJOIN エレメントが指定される場合、次のようになります。

- INDEX、TYPE、および ALLINDEXES 属性を指定できません
- INDEX エレメントを指定できません
- STARJOIN 属性でサポートされる唯一の値は TRUE です

以下の例では、索引 ANDing アクセス要求を示しています。

SQL statement:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
 where p_partkey = ps.ps_partkey and
       s.s_suppkey = ps.ps_suppkey and
       p_size = 39 and
       p_type = 'BRASS' and
       s.s_nation in ('MOROCCO', 'SPAIN') and
       ps.ps_supplycost = (select min(ps1.ps_supplycost)
                           from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                           where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                               s1.s_suppkey = ps1.ps_suppkey and
                               s1.s_nation = s.s_nation)

 order by s.s_name
 optimize for 1 row
```

Optimization guideline:

```
<OPTGUIDELINES>
  <IXAND TABLE="Tpcd".PARTS' FIRST='TRUE'>
    <INDEX IXNAME='ISIZE' />
    <INDEX IXNAME='ITYPE' />
  </IXAND>
</OPTGUIDELINES>
```

索引 ANDing 要求は、メインの副選択内の PARTS 表へのアクセスの必要を満たすために、索引 ANDing データ・アクセス方式を使用するように指定します。最初の索引スキャンは ISIZE 索引を使用し、2 番目の索引スキャンは ITYPE 索引を使用します。索引は、INDEX エレメントの IXNAME 属性によって指定されます。

FIRST 属性の設定は、PARTS 表が、SUPPLIERS、PARTSUPP、および同じ FROM 節内の派生表を含む結合順序内の最初の表になるように指定します。

以下の例は、スター型結合の索引 ANDing のガイドラインを示しています。このガイドラインでは最初の半結合を指定し、残りはオプティマイザーが選択しています。また、指定された半結合で、外部表の特定のアクセス方式および内部表の索引も、オプティマイザーが選択しています。

```
<IXAND TABLE="F">
  <NLJOIN>
    <ACCESS TABLE="D1"/>
    <IXSCAN TABLE="F"/>
  </NLJOIN>
</IXAND>
```

以下のガイドラインは、詳細を含むすべての半結合を指定しており、オプティマイザーに IXAND のプランおよび IXAND の下のプランに関して選択の余地を残していません。

```
<IXAND TABLE="F" STARJOIN="TRUE">
  <NLJOIN>
    <TBSCAN TABLE="D1"/>
    <IXSCAN TABLE="F" INDEX="FX1"/>
  </NLJOIN>
  <NLJOIN>
    <TBSCAN TABLE="D4"/>
    <IXSCAN TABLE="F" INDEX="FX4"/>
  </NLJOIN>
  <NLJOIN>
    <TBSCAN TABLE="D3"/>
    <IXSCAN TABLE="F" INDEX="FX3"/>
  </NLJOIN>
</IXAND>
```

索引 ORing アクセス要求:

IXOR アクセス要求エレメントを使用して、オプティマイザーがローカル表にアクセスする際に索引 ORing データ・アクセス方式を使用するように指定できます。これは、複合タイプ `indexOringType` によって定義されます。

XML Schema

```
<xs:complexType name="indexOringType">
  <xs:complexContent>
    <xs:extension base="accessType"/>
  </xs:complexContent>
</xs:complexType>
```

説明

複合タイプ `indexOringType` は、抽象タイプ `accessType` の単なる拡張です。新規の要素や属性は追加されていません。ステートメントに関して有効な検索スペース内に索引 ORing アクセス方式がない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。オプティマイザーは、索引 ORing 操作で使用する述部および索引をコスト・ベース方式で選択します。索引 ORing アクセス方式では、少なくとも 1 つの索引付けが可能な IN 述部、または索引付けおよび論理 OR 演算による結合が可能な語を伴う述部が必要です。必須の述部または索引が存在しないために索引 ORing が適格ではない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。

以下の例では、索引 ORing アクセス要求を示しています。

SQL statement:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
 where p_partkey = ps.ps_partkey and
       s.s_suppkey = ps.ps_suppkey and
       p_size = 39 and
       p_type = 'BRASS' and
       s.s_nation in ('MOROCCO', 'SPAIN') and
       ps.ps_supplycost = (select min(ps1.ps_supplycost)
                           from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                           where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                                 s1.s_suppkey = ps1.ps_suppkey and
                                 s1.s_nation = s.s_nation)

 order by s.s_name
 optimize for 1 row
```

Optimization guideline:

```
<OPTGUIDELINES>
  <IXOR TABLE='S' />
</OPTGUIDELINES>
```

この索引 ORing アクセス要求は、索引 ORing データ・アクセス方式を使用して、メインの副選択内で参照される SUPPLIERS 表にアクセスするように指定します。オブティマイザーは、索引 ORing 操作に適切な述部および索引をコスト・ベース方式で選択します。

索引スキャン・アクセス要求:

IXSCAN アクセス要求エレメントを使用して、オブティマイザーがローカル表にアクセスする際に索引スキャンを使用するように指定できます。これは、複合タイプ indexScanType によって定義されます。

XML Schema

```
<xs:complexType name="indexScanType">
  <xs:complexContent>
    <xs:extension base="accessType">
      <xs:attribute name="INDEX" type="xs:string" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

説明

複合タイプ indexScanType は、オプションの INDEX 属性を追加して、抽象 accessType を拡張します。INDEX 属性は、表へのアクセスで使用される索引の名前を指定します。

- ステートメントに関して有効な検索スペース内に索引スキャン・アクセス方式がない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。
- INDEX 属性を指定する場合、TABLE または TABID 属性で識別される表で定義されている索引を識別する必要があります。索引が存在しない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。
- INDEX 属性を指定しないと、オブティマイザーはコスト・ベース方式で索引を選択します。ターゲット表で索引が定義されていない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。

以下のガイドラインでは、索引スキャン・アクセス要求の例を示します。

```
<OPTGUIDELINES>
  <IXSCAN TABLE='S' INDEX='I_SUPPKEY' />
</OPTGUIDELINES>
```

リスト・プリフェッチ・アクセス要求:

LPREFETCH アクセス要求エレメントを使用して、オプティマイザーがローカル表にアクセスする際にリスト・プリフェッチ索引スキャンを使用するように指定できます。これは、複合タイプ `listPrefetchType` によって定義されます。

XML Schema

```
<xs:complexType name="listPrefetchType">
  <xs:complexContent>
    <xs:extension base="accessType">
      <xs:attribute name="INDEX" type="xs:string" use="optional" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

説明

複合タイプ `listPrefetchType` は、オプションの `INDEX` 属性を追加して、抽象タイプ `accessType` を拡張します。`INDEX` 属性は、表へのアクセスで使用される索引の名前を指定します。

- ステートメントに関して有効な検索スペース内にリスト・プリフェッチ・アクセス方式がない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。
- リスト・プリフェッチ・アクセス方式では、索引付けが可能な述部が少なくとも 1 つ必要です。必須の述部が存在しないためにリスト・プリフェッチ・アクセス方式が適格ではない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。
- `INDEX` 属性を指定する場合、`TABLE` または `TABID` 属性で指定される表で定義されている索引を識別する必要があります。索引が存在しない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。
- `INDEX` 属性を指定しないと、オプティマイザーはコスト・ベース方式で索引を選択します。ターゲット表で索引が定義されていない場合、アクセス要求は無視されて、理由コード 13 の SQL0437W が戻ります。

以下のガイドラインでは、リスト・プリフェッチ・アクセス要求の例を示します。

```
<OPTGUIDELINES>
  <LPREFETCH TABLE='S1' INDEX='I_SNATION' />
</OPTGUIDELINES>
```

表スキャン・アクセス要求:

TBSCAN アクセス要求エレメントを使用して、オプティマイザーがローカル表にアクセスする際に順次表スキャンを使用するように指定できます。これは、複合タイプ `tableScanType` によって定義されます。

XML Schema

```
<xs:complexType name="tableScanType">
```

```

    <xs:complexContent>
      <xs:extension base="accessType"/>
    </xs:complexContent>
  </xs:complexType>

```

説明

複合タイプ `tableScanType` は、抽象タイプ `accessType` の単なる拡張です。新規の要素や属性は追加されていません。ステートメントに関して有効な検索スペース内に表スキャン・アクセス方式がない場合、アクセス要求は無視されて、理由コード 13 の `SQL0437W` が戻ります。

以下のガイドラインでは、表スキャン・アクセス要求の例を示します。

```

<OPTGUIDELINES>
  <TBSCAN TABLE='S1' />
</OPTGUIDELINES>

```

XML 索引 ANDing および ORing アクセス要求:

XANDOR アクセス要求要素を使用して、オプティマイザーがローカル表にアクセスする際に、XANDOR 処理された複数の XML データの索引スキャンを使用するように指定できます。これは、複合タイプ `XANDORType` によって定義されます。

XML Schema

```

<xs:complexType name="XANDORType">
  <xs:complexContent>
    <xs:extension base="accessType"/>
  </xs:complexContent>
</xs:complexType>

```

説明

複合タイプ `XANDORType` は、抽象タイプ `accessType` の単なる拡張です。新規の要素や属性は追加されていません。

例

以下の照会があるとします。

```

SELECT * FROM security
WHERE trans_date = CURRENT DATE AND
  XML EXISTS('$SDOC/Security/SecurityInformation/
    StockInformation[Industry = "Software"]') AND
  XML EXISTS('$SDOC/Security/Symbol[.="IBM"'])

```

以下の XANDOR ガイドラインは、SECURITY 表が、すべての適用可能な XML 索引に対する XANDOR 演算を使用してアクセスされることを指定します。リレーショナル索引は XANDOR 演算子とともに使用できないため、SECURITY 表のリレーショナル索引は考慮されません。

```

<OPTGUIDELINES>
  <XANDOR TABLE='SECURITY' />
</OPTGUIDELINES>

```

XML 索引スキャン・アクセス要求:

XISCAN アクセス要求エレメントを使用して、オプティマイザーがローカル表にアクセスする際に XML データの索引のスキャンを使用するように指定できます。これは、複合タイプ `indexScanType` によって定義されます。

XML Schema

```
<xs:complexType name="indexScanType">
  <xs:complexContent>
    <xs:extension base="accessType"/>
    <xs:attribute name="INDEX" type="xs:string" use="optional"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

説明

複合タイプ `indexScanType` は、オプションの `INDEX` 属性を追加して、抽象 `accessType` を拡張します。`INDEX` 属性は、表へのアクセスで使用される XML データの索引の名前を指定します。

- ステートメントに関して有効な検索スペース内に XML データの索引スキャン・アクセス方式がない場合、アクセス要求は無視されて、理由コード 13 の `SQL0437W` が戻ります。
- `INDEX` 属性を指定する場合、`TABLE` または `TABID` 属性で識別される表で定義されている XML データの索引を識別する必要があります。索引が存在しない場合、アクセス要求は無視されて、理由コード 13 の `SQL0437W` が戻ります。
- `INDEX` 属性を指定しないと、オプティマイザーはコスト・ベース方式で XML データの索引を選択します。ターゲット表で XML データの索引が定義されていない場合、アクセス要求は無視されて、理由コード 13 の `SQL0437W` が戻ります。

例

以下の照会があるとします。

```
SELECT * FROM security
WHERE XMLEXISTS('$SDOC/Security/SecurityInformation/
StockInformation[Industry = "OfficeSupplies"]')
```

以下の XISCAN ガイドラインは、`SECURITY` 表が `SEC_INDUSTRY` という名前の XML 索引を使用してアクセスされることを指定します。

```
<OPTGUIDELINES>
  <XISCAN TABLE='SECURITY' INDEX='SEC_INDUSTRY' />
</OPTGUIDELINES>
```

結合要求:

`joinRequest` グループは、有効な一連の結合要求エレメントを定義します。結合要求は、2 つの表の結合用方式を指定します。

XML Schema

```
<xs:group name="joinRequest">
  <xs:choice>
    <xs:element name="NLJOIN" type="nestedLoopJoinType"/>
    <xs:element name="HSJOIN" type="hashJoinType"/>
  </xs:choice>
</xs:group>
```

```

        <xs:element name="MSJOIN" type="mergeJoinType"/>
        <xs:element name="JOIN" type="anyJoinType"/>
    </xs:choice>
</xs:group>

```

説明

- NLJOIN、MSJOIN、および HSJOIN

これらのエレメントは、それぞれネスト・ループ、マージ、およびハッシュ結合方式に対応します。

- JOIN

オプティマイザが結合方式を選択するようにするこのエレメントは、結合順序が主要な関心事でない場合に使用できます。

どの結合要求エレメントにも、結合操作の入力表を表す 2 つのサブエレメントが入っています。結合要求はまた、オプションの FIRST 属性を指定することもできます。

以下のガイドラインでは、結合要求の例を示します。

```

<OPTGUIDELINES>
  <HSJOIN>
    <ACCESS TABLE='S1' />
    <IXSCAN TABLE='PS1' />
  </HSJOIN>
</OPTGUIDELINES>

```

ネスト順序が、最終的に結合順序を決定します。以下の例は、より小さな結合要求によって、より大きな結合要求を構成する方法を示しています。

```

<OPTGUIDELINES>
  <MSJOIN>
    <NLJOIN>
      <IXSCAN TABLE='"Tpcd".Parts' />
      <IXSCAN TABLE="PS" />
    </NLJOIN>
    <IXSCAN TABLE='S' />
  </MSJOIN>
</OPTGUIDELINES>

```

結合タイプ:

すべての結合要求エレメントに共通する局面は、抽象タイプ joinType によって定義されます。

XML Schema

```

<xs:complexType name="joinType" abstract="true">
  <xs:choice minOccurs="2" maxOccurs="2">
    <xs:group ref="accessRequest"/>
    <xs:group ref="joinRequest"/>
  </xs:choice>
  <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
</xs:complexType>

```

説明

複合タイプ joinType を拡張する結合要求エレメントには、正確に 2 つのサブエレメントがなければなりません。どちらのサブエレメントも、accessRequest グループ

から選択されるアクセス要求エレメント、または `joinRequest` グループから選択される別の結合要求エレメントにすることができます。結合要求に最初に現れるサブエレメントが、結合操作の外部表を指定し、2 番目のエレメントが内部表を指定します。

`FIRST` 属性を指定する場合、その値は `TRUE` でなければなりません。結合要求エレメントに `FIRST` 属性を追加すると、結合要求のターゲットである表を、対応する `FROM` 節の結合シーケンスの最外部の表とするような実行プランが必要であることを指示します。`FIRST` 属性を指定できるアクセスまたは結合要求は、`FROM` 節あたり 1 つだけです。1 つの同じ `FROM` 節の表をターゲットとする複数のアクセスまたは結合要求が `FIRST` 属性を指定する場合、要求のうちの最初のもの以外はすべて無視されて、理由コード 13 の `SQL0437W` が戻ります。

任意の結合要求:

`JOIN` 結合要求エレメントを使用して、特定の順序で 2 つの表を結合するために適切な方式をオプティマイザーが選択するように指定できます。

どちらの表も、アクセス要求サブエレメントでの指定どおりに、ローカル表または派生表のいずれでも良く、結合要求サブエレメントでの指定どおりに結合操作の結果になります。派生表は、別の副選択の結果です。以下の結合要求エレメントは、複合タイプ `anyJoinType` によって定義されます。

XML Schema

```
<xs:complexType name="anyJoinType">
  <xs:complexContent>
    <xs:extension base="joinType"/>
  </xs:complexContent>
</xs:complexType>
```

説明

複合タイプ `anyJoinType` は、抽象タイプ `joinType` の単なる拡張です。新規のエレメントや属性は追加されていません。

以下の例では、表集合の特定の結合順序を強制する、`JOIN` 結合要求エレメントの使用について示しています。

SQL statement:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
  where p_partkey = ps.ps_partkey and
         s.s_suppkey = ps.ps_suppkey and
         p_size = 39 and
         p_type = 'BRASS' and
         s.s_nation in ('MOROCCO', 'SPAIN') and
         ps.ps_supplycost = (select min(ps1.ps_supplycost)
                              from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                              where "Tpcd".parts.p_partkey = ps1.p_partkey and
                                    s1.s_suppkey = ps1.ps_suppkey and
                                    s1.s_nation = s.s_nation)

  order by s.s_name
```

Optimization guideline:

```
<OPTGUIDELINES>
  <JOIN>
```

```

    <JOIN>
      <ACCESS TABLE='Tpcd'.PARTS' />
      <ACCESS TABLE='S' />
    </JOIN>
    <ACCESS TABLE='PS'>
  </JOIN>
</OPTGUIDELINES>

```

JOIN 結合要求エレメントは、メインの副選択内の PARTS 表が SUPPLIERS 表と結合され、この結果が PARTSUPP 表に結合されるように指定します。オプティマイザーは、この特定の順序の結合に関して、コスト・ベース方式で結合方式を選択します。

ハッシュ結合要求:

HSJOIN 結合要求エレメントを使用して、オプティマイザーがハッシュ結合方式を使用して 2 つの表を結合するように指定できます。

どちらの表も、アクセス要求サブエレメントでの指定どおりに、ローカル表または派生表のいずれでも良く、結合要求サブエレメントでの指定どおりに結合操作の結果になります。派生表は、別の副選択の結果です。以下の結合要求エレメントは、複合タイプ hashJoinType によって定義されます。

XML Schema

```

<xs:complexType name="hashJoinType">
  <xs:complexContent>
    <xs:extension base="joinType" />
  </xs:complexContent>
</xs:complexType>

```

説明

複合タイプ hashJoinType は、抽象タイプ joinType の単なる拡張です。新規のエレメントや属性は追加されていません。ステートメントに関して有効な検索スペース内にハッシュ結合方式がない場合、結合要求は無視されて、理由コード 13 の SQL0437W が戻ります。

以下のガイドラインでは、ハッシュ結合要求の例を示します。

```

<OPTGUIDELINES>
  <HSJOIN>
    <ACCESS TABLE='S1' />
    <IXSCAN TABLE='PS1' />
  </HSJOIN>
</OPTGUIDELINES>

```

マージ結合要求:

MSJOIN 結合要求エレメントを使用して、オプティマイザーがマージ結合方式を使用して 2 つの表を結合するように指定できます。

どちらの表も、アクセス要求サブエレメントでの指定どおりに、ローカル表または派生表のいずれでも良く、結合要求サブエレメントでの指定どおりに結合操作の結果になります。派生表は、別の副選択の結果です。以下の結合要求エレメントは、複合タイプ mergeJoinType によって定義されます。

XML Schema

```
<xs:complexType name="mergeJoinType">
  <xs:complexContent>
    <xs:extension base="joinType"/>
  </xs:complexContent>
</xs:complexType>
```

説明

複合タイプ `mergeJoinType` は、抽象タイプ `joinType` の単なる拡張です。新規の要素や属性は追加されていません。ステートメントに関して有効な検索スペース内にマージ結合方式がない場合、結合要求は無視されて、理由コード 13 の `SQL0437W` が戻ります。

以下のガイドラインでは、マージ結合要求の例を示します。

```
<OPTGUIDELINES>
  <MSJOIN>
    <NLJOIN>
      <IXSCAN TABLE='Tpcd'.Parts' />
      <IXSCAN TABLE="PS" />
    </NLJOIN>
    <IXSCAN TABLE='S' />
  </MSJOIN>
</OPTGUIDELINES>
```

ネスト・ループ結合要求:

`NLJOIN` 結合要求要素を使用して、オプティマイザーがネスト・ループ結合方式を使用して 2 つの表を結合するように指定できます。

どちらの表も、アクセス要求サブ要素での指定どおりに、ローカル表または派生表のいずれでも良く、結合要求サブ要素での指定どおりに結合操作の結果になります。派生表は、別の副選択の結果です。以下の結合要求要素は、複合タイプ `nestedLoopJoinType` によって定義されます。

XML Schema

```
<xs:complexType name="nestedLoopJoinType">
  <xs:complexContent>
    <xs:extension base="joinType"/>
  </xs:complexContent>
</xs:complexType>
```

説明

複合タイプ `nestedLoopJoinType` は、抽象タイプ `joinType` の単なる拡張です。新規の要素や属性は追加されていません。ステートメントに関して有効な検索スペース内にネスト・ループ結合方式がない場合、結合要求は無視されて、理由コード 13 の `SQL0437W` が戻ります。

以下のガイドラインでは、ネスト・ループ結合要求の例を示します。

```
<OPTGUIDELINES>
  <NLJOIN>
    <IXSCAN TABLE='Tpcd'.Parts' />
    <IXSCAN TABLE="PS" />
  </NLJOIN>
</OPTGUIDELINES>
```

SYSTOOLS.OPT_PROFILE 表:

SYSTOOLS.OPT_PROFILE 表には、すべての最適化プロファイルが含まれています。

この表を作成するには、次の 2 つの方法があります。

- 次のようにして、SYSINSTALLOBJECTS プロシージャを呼び出します。

```
db2 "call sysinstallobjects('opt_profiles', 'c', '', '')"
```

- 次のようにして CREATE TABLE ステートメントを発行します。

```
create table systools.opt_profile (  
  schema varchar(128) not null,  
  name   varchar(128) not null,  
  profile blob (2m)   not null,  
  primary key (schema, name)  
)
```

SYSTOOLS.OPT_PROFILE 表の列は次のように定義されます。

SCHEMA

最適化プロファイルのスキーマ名を指定します。名前には最高 30 文字の英数字または下線文字を含めることができますが、それを以下に示すように VARCHAR(128) として定義できます。

NAME

最適化プロファイルのベース名を指定します。名前には最高 128 文字の英数字または下線文字を含めることができます。

PROFILE

最適化プロファイルを定義する XML 文書を指定します。

最適化プロファイル・キャッシュをフラッシュするためのトリガー:

最適化プロファイル・キャッシュは、SYSTOOLS.OPT_PROFILE 表内の項目が更新または削除されるときに自動的にフラッシュされます。

プロファイル・キャッシュの自動フラッシュを行えるようにするには、その前に以下の SQL プロシージャとトリガーを作成する必要があります。

```
CREATE PROCEDURE SYSTOOLS.OPT_FLUSH_CACHE( IN SCHEMA VARCHAR(128),  
                                           IN NAME VARCHAR(128) )  
  
LANGUAGE SQL  
MODIFIES SQL DATA  
BEGIN ATOMIC  
  -- FLUSH stmt (33) + quoted schema (130) + dot (1) + quoted name (130) = 294  
  DECLARE FSTMT VARCHAR(294) DEFAULT 'FLUSH OPTIMIZATION PROFILE CACHE '; --  
  
  IF NAME IS NOT NULL THEN  
    IF SCHEMA IS NOT NULL THEN  
      SET FSTMT = FSTMT || ''' ' || SCHEMA || '.'; --  
    END IF; --  
  
    SET FSTMT = FSTMT || ''' ' || NAME || '''; --  
  
    EXECUTE IMMEDIATE FSTMT; --  
  END IF; --  
END;
```

```
CREATE TRIGGER SYSTOOLS.OPT_PROFILE_UTRIG AFTER UPDATE ON SYSTOOLS.OPT_PROFILE  
REFERENCING OLD AS O
```

```

FOR EACH ROW
  CALL SYSTOOLS.OPT_FLUSH_CACHE( O.SCHEMA, O.NAME );

CREATE TRIGGER SYSTOOLS.OPT_PROFILE_DTRIG AFTER DELETE ON SYSTOOLS.OPT_PROFILE
REFERENCING OLD AS O
FOR EACH ROW
  CALL SYSTOOLS.OPT_FLUSH_CACHE( O.SCHEMA, O.NAME );

```

SYSTOOLS.OPT_PROFILE 表の管理:

最適化プロファイルは、固有のスキーマ修飾名に関連付けて、SYSTOOLS.OPT_PROFILE 表に保管する必要があります。 **LOAD**、**IMPORT**、および **EXPORT** コマンドを使用して、その表内のファイルを管理できます。

例えば、**IMPORT** コマンドを任意の DB2 クライアントから使用して、SYSTOOLS.OPT_PROFILE 表にデータを挿入するかまたは更新できます。 **EXPORT** コマンドを使用して、プロファイルを SYSTOOLS.OPT_PROFILE 表からファイルにコピーできます。

次の例は、SYSTOOLS.OPT_PROFILE 表に新しい 3 つのプロファイルを挿入する方法を示しています。ファイルが現行ディレクトリーにあることを想定しています。

1. 個々のプロファイルにスキーマ、名前、およびファイル名を別々の行で指定して、入力ファイル (例えば、profiledata) を作成します。

```

"ROBERT","PROF1","ROBERT.PROF1.xml"
"ROBERT","PROF2","ROBERT.PROF2.xml"
"DAVID","PROF1","DAVID.PROF1.xml"

```

2. **IMPORT** コマンドを実行します。

```

import from profiledata of del
modified by lobsinfile
insert into systools.opt_profile

```

既存の行を更新するには、以下のようにして、**IMPORT** コマンドで **INSERT_UPDATE** オプションを使用します。

```

import from profiledata of del
modified by lobsinfile
insert_update into systools.opt_profile

```

ROBERT.PROF1 プロファイルを ROBERT.PROF1.xml にコピーするには、**EXPORT** コマンドを以下のように使用します。このプロファイルの長さは 32 700 バイトより小さいと想定します。

```

export to robert.prof1.xml of del
select profile from systools.opt_profile
where schema='ROBERT' and name='PROF1'

```

32 700 バイトを超えるデータのエクスポート方法を含め詳しくは、『**EXPORT** コマンド』を参照してください。

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・データベース環境では、オプティマイザーは、照会に対する最適のアクセス・プランを判別する際に表のコロケーションを認識し、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・パーティションに置かれるように、データベース・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、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、データをあるデータベース・パーティションから別のデータベース・パーティションに移動できなくなります。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、そのデータが確実に連結されるようにしてください。

データをより多くのデータベース・パーティションに配分すると、表のサイズに応じて照会の実行にかかる見積時間が減少します。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。

正確なカタログ統計の収集 (拡張統計フィーチャーを含む)

正確なデータベース統計は、照会の最適化に不可欠です。照会のパフォーマンス上重要なすべての表で、**RUNSTATS** コマンド操作を定期的に行ってください。

また、アプリケーションがシステム・カタログ表を直接照会しており、重要なカタログ更新アクティビティ (データ定義言語 (DDL) ステートメントの実行によるものなど) がある場合、システム・カタログ表についての統計を収集する必要があるかもしれません。自動統計収集を有効にすると、DB2 データ・サーバーによる **RUNSTATS** コマンド操作を自動的に実行できます。リアルタイム統計収集を有効にすると、DB2 データ・サーバーは、照会が最適化される直前に統計を収集することにより、いっそうタイムリーな統計を提供できます。

RUNSTATS コマンドを使用して手動で統計を収集している場合は、少なくとも以下のオプションを使用してください。

```
RUNSTATS ON TABLE DB2USER.DAILY_SALES  
WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL
```

分散統計において、 옵ティマイザーはデータの偏りを検知します。詳細索引統計により、特定の索引を使用して表にアクセスする際に、データ・ページのフェッチに必要な入出力について、さらに詳細が得られます。大規模な表の場合、詳細索引統計の収集により、相当な処理時間およびメモリーが使用されます。 **SAMPLED** オプションを使用すると、詳細索引統計を同程度の正確さで得られますが、必要とする CPU およびメモリーはわずかで済みます。表のために統計プロファイルが提供されていない場合、自動統計収集はこれらのオプションを使用します。

照会パフォーマンスを向上させるには、列グループ統計または Like 統計といった拡張統計の収集や、統計ビューの作成を検討してください。

統計ビューは、複雑な関係に関する統計を収集するときに便利です。統計ビューへの統計の収集は、DB2 Database for Linux, UNIX, and Windows の自動統計収集フィーチャーにより自動化できます。統計ビューの自動統計収集の有効化または無効化は、**auto_stats_views** データベース構成パラメーターを使用して行います。この機能を有効にするには、次のコマンドを発行します。

```
update db cfg for dbname using auto_stats_views on
```

このフィーチャーを無効にするには、次のコマンドを発行します。

```
update db cfg for dbname using auto_stats_views off
```


このデータベース構成パラメーターは、デフォルトではオフになっています。統計ビューで統計を自動収集するために発行するコマンドは、次のコマンドと同等です。

```
runstats on view view_name with distribution
```

大規模な表または統計ビューの統計を収集する場合、時間がかかることがあります。全データの中のほんの一部のサンプルを考慮することで、同等の品質の統計を収集できる場合がよくあります。すべてのバックグラウンド統計収集について自動サンプリングを有効にすることを検討してください。これにより、統計収集時間を削減できる可能性があります。この機能を有効にするには、次のコマンドを発行します。

```
update db cfg for dbname using auto_sampling on
```

列グループ統計

2 つの表を結合する結合述部が照会に複数含まれる場合、DB2 オプティマイザーは、照会を実行するためのプランを選択する前に、これらの述部の選択可能性をそれぞれ計算します。

例えば、さまざまな色、伸縮性、および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。メーカーは、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、これらの 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に基づく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、例えばネスト・ループ結合とマージ結合のどちらを優先するかなどが選択される場合があります。

しかし、これら 2 つの述部は独立していない可能性があります。例えば、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は少なくなり、照会により戻される行は多くなります。この情報がないと、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。

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))
```

オブティマイザーは、これらの統計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。

照会で `GROUP BY` または `DISTINCT` などのキーワードを使用してデータをグループ化する際、列グループ統計により、オブティマイザーが個々のグループの数を計算することも可能となります。

次の照会を考えてみましょう。

```
SELECT DEPTNO, YEARS, AVG(SALARY)
FROM EMPLOYEE
GROUP BY DEPTNO, MGR, YEAR_HIRED
```

索引または列のグループ統計を使用しない場合、オブティマイザーは、グループの数（およびこの場合、戻される行の数）が `DEPTNO`、`MGR`、および `YEAR_HIRED` にある個々の値の数の積であるとして見積もります。この見積もりは、グループ化キー列が独立していることを前提としています。ただし、各管理者が 1 つの部門だけを管理している場合には、この前提事項に該当しない可能性があります。さらに、各部門が従業員の雇用を毎年行っていない可能性もあります。このように、`DEPTNO`、`MGR`、および `YEAR_HIRED` の個々の値の積は、個々のグループの実際の数よりも多く見積もられている可能性があります。

以下のように、`DEPTNO`、`MGR`、および `YEAR_HIRED` で収集された列グループ統計により、オブティマイザーは、前の照会における個々のグループの正確な数を得ることができます。

```
RUNSTATS ON TABLE EMPLOYEE ON COLUMNS ((DEPTNO, MGR, YEAR_HIRED))
```

`JOIN` 述部相関に加えて、オブティマイザーは、以下のような単純な等価述部に関する相関を管理します。

```
DEPTNO = 'Sales' AND MGR = 'John'
```

この例では、`EMPLOYEE` 表の `DEPTNO` 列に関する述部は、`YEAR` 列に関する述部から独立していると思われます。しかし、`DEPTNO` および `MGR` に関する述部同士は独立していません。なぜなら普通、1 つの部門を一度に管理するのは 1 人の管理者だからです。オブティマイザーは、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、列間の相関を示すカーディナリティー推定値を調整します。

統計ビューでは、列グループ統計も使用できます。列グループ統計は、照会同士の間に関係が複数ある場合に、統計ビュー内の統計の偏りを調整するのに役立ちます。オブティマイザーはこれらの統計を使用してより良いカーディナリティー推定値を取得でき、その結果、より良いアクセス・プランが作成されます。

単純な等価述部の相関

結合述部相関に加えて、オブティマイザーはタイプ `COL =` の単純な等価述部に関する相関も管理します。

例えば、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、またはスポーツ・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、MAKE および MODEL に基づく述部は独立していません。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めったにありません。

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に関する統計情報を使用して、固有値の数の合計を判別し、これら 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。述部がローカル等価述部の場合、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。

統計ビュー

DB2 のコスト・ベースのオプティマイザーは、アクセス・プラン演算子によって処理された行数の推定値を使用して、その演算子を正確にコスト計算します。このカーディナリティー推定値は、オプティマイザーのコスト・モデルへの唯一最重要な入力データであり、その正確性は、**RUNSTATS** コマンドによってデータベースから収集される統計に大きく依存しています。

次に示す関係のように、より複雑な関係を表すには、より洗練された統計が必要です。

- 式を含む比較。例えば `price > MSRP + Dealer_markup` など。
- 複数の表にまたがる関係。例えば、`product.name = 'Alloy wheels'` and `product.key = sales.product_key` など。
- 独立した属性および単純な比較操作を行う述部以外の関係。

統計ビューは、この種の複雑な関係を表すことができます。なぜなら、ビューによって参照される基本表ではなく、ビューによって返される結果セットに、統計が収集されるためです。

照会のコンパイル時に、オプティマイザーは、使用可能な統計ビューに照会を突き合わせます。オプティマイザーは、カーディナリティー推定値を中間結果セットのために計算する際、ビューからの統計を使用してより優れた推定値を算出します。

オプティマイザーが統計ビューを使用するために、照会で統計ビューを直接参照する必要はありません。オプティマイザーは、照会を統計ビューに突き合わせる際、マテリアライズ照会表 (MQT) で使用されるのと同じ突き合わせ手段を使用します。この点、統計ビューは MQT に似ていますが、永続的に格納されないこと、ディスク・スペースを消費しないこと、および保守される必要がないことは異なっています。

統計ビューを作成するには、まずビューを作成し、次いで **ALTER VIEW** ステートメントを使って最適化に使用できるようにします。それから、**RUNSTATS** コマンドを統計ビューに対して実行すると、システム・カタログ表にビューの統計が設定されます。例えば、スター・スキーマで **TIME** ディメンション表とファクト表との結合を表す統計ビューを作成するには、以下のステートメントとコマンドを実行します。

```
CREATE VIEW SV_TIME_FACT AS (  
  SELECT T.* FROM TIME T, SALES S  
  WHERE T.TIME_KEY = S.TIME_KEY)
```

```
ALTER VIEW SV_TIME_FACT ENABLE QUERY OPTIMIZATION
```

```
RUNSTATS ON TABLE DB2DBA.SV_TIME_FACT WITH DISTRIBUTION
```

この統計ビューを使用することにより、カーディナリティー推定値、そして結果として、以下のような照会のアクセス・プランおよび照会パフォーマンスを向上させることができます。

```
SELECT SUM(S.PRICE)
FROM SALES S, TIME T, PRODUCT P
WHERE
  T.TIME_KEY = S.TIME_KEY AND
  T.YEAR_MON = 200712 AND
  P.PROD_KEY = S.PROD_KEY AND
  P.PROD_DESC = 'Power drill'
```

統計ビューがない場合、`TIME` ディメンション `YEAR_MON` の特定の値に対応する、ファクト表の `TIME_KEY` の値すべてが、均等にファクト表内に存在しているとオプティマイザーによってみなされます。しかし、販売高は 12 月が特に多く、他の月よりも販売トランザクションが多かった可能性もあります。

述部に複雑な式を持つ照会について収集された統計をオプティマイザーが使用して正確なカーディナリティー推定値を計算し、より良いアクセス・プランを作成することができます。

多くのスター型結合照会では、多くの統計ビューの作成が必要な場合がありますが、参照整合性制約がある場合、これら多くの統計ビューを絞り込むことができます。他のビューの統計は、参照整合性制約を使用して削減された統計ビューの数から推論できます。

より良いアクセス・プランを獲得するもう 1 つの方法は、統計ビューに列グループ統計を適用する方法です。これらのグループ統計によって、より正確な統計を収集するために役立つフィルター係数を調整することができます。正確な統計によって、オプティマイザーは正確なカーディナリティー推定値を取得することができます。

統計は、DB2 Database for Linux, UNIX, and Windows の自動統計収集フィーチャーを使用して統計ビューから自動的に収集することもできます。この新しいフィーチャーは、`auto_stats_views` データベース構成パラメーターを使用して使用可能にしたり使用不可にしたりすることができます。このデータベース構成パラメーターはデフォルトではオフになっており、`UPDATE DB CFG` コマンドを使用して使用可能にできます。自動統計収集により収集される統計は、コマンド `runstats on view view_name with distribution` を発行するのと同様です。

統計ビューにユーティリティー・スロットルを使用して、ワークロードへのパフォーマンスの影響を制限することができます。例えば、コマンド `runstats on view view_name util_impact_priority 10` では、`util_impact_lim` データベース・マネージャー構成パラメーターにより指定された限度内で、(統計ビュー上で統計が収集されている間に) ワークロードに与える影響が指定されています。

統計ビューの使用

照会を最適化するために統計を使用するには、最適化でビューが使用可能でなければなりません。最適化で使用可能なビューを、*統計ビュー* といいます。

このタスクについて

統計ビューではないビューは、最適化では無効であるとみなされ、通常ビュー といえます。ビューは、最初に作成された時は、最適化で無効になっています。ビューを最適化に使用できるようにするには、ALTER VIEW ステートメントを使用します。このタスクを実行するのに必要な特権および権限については、ALTER VIEW ステートメントの説明を参照してください。ビューに対して RUNSTATS ユーティリティを使用するのに必要な特権および権限については、RUNSTATS コマンドの説明を参照してください。

以下の条件のいずれかが成立する場合、ビューを最適化のために有効にすることはできません。

- ビューが直接または間接にマテリアライズ照会表 (MQT) を参照する。(MQT または統計ビューは統計ビューを参照できます)。
- ビューが作動不能である。
- ビューは、型付きビューである。
- 同じ ALTER VIEW ステートメントの中に別のビュー変更要求がある。

最適化を有効とするために変更されるビューの定義に、以下の項目のいずれかが含まれる場合、警告が返され、 옵ティマイザーはビューの統計を活用しません。

- 集約または DISTINCT 操作
- UNION、EXCEPT、または INTERSECT 操作
- OLAP 仕様

手順

1. 最適化のためにビューを有効にします。

ALTER VIEW ステートメントで ENABLE OPTIMIZATION 節を使用すると、ビューを最適化のために有効とすることができます。最適化のために有効となったビューは、DISABLE OPTIMIZATION 節を使用すると、後で最適化のために無効とすることができます。例えば、最適化のために MYVIEW を有効にするには、次のように入力します。

```
alter view myview enable query optimization
```

2. RUNSTATS コマンドを実行します。例えば、MYVIEW 上で統計を収集するには、次のように入力します。

```
runstats on table db2dba.myview
```

分散統計を含むビュー統計の収集中に、行の 10% からなる行レベルのサンプリングを使用するには、次のように入力します。

```
runstats on table db2dba.myview with distribution tablesample bernoulli (10)
```

分散統計を含むビュー統計の収集中に、ページの 10% からなるページ・レベルのサンプリングを使用するには、次のように入力します。

```
runstats on table db2dba.myview with distribution tablesample system (10)
```

3. オプション: ビュー定義の影響を受ける照会が静的 SQL パッケージの一部である場合、これらのパッケージを再バインドして、新しい統計によって加えられたアクセス・プランに対する変更を利用できるようにします。

最適化に関連するビュー統計

照会最適化の際には、CARD や COLCARD など、統計ビューを定義する照会のデータ分布を特徴づける統計値のみが考慮されます。

収集してオプティマイザーで使用できるのは、ビュー・レコードに関連付けられている次の統計だけです。

- 表統計 (SYSCAT.TABLES、SYSSTAT.TABLES)
 - CARD - ビューの結果における行数です。
- 列統計 (SYSCAT.COLUMNS、SYSSTAT.COLUMNS)
 - COLCARD - ビューの結果における列の特殊な値の数です。
 - AVGCOLLEN - ビューの結果における平均の列長です。
 - HIGH2KEY - ビューの結果における列の 2 番目に大きい値です。
 - LOW2KEY - ビューの結果における列の 2 番目に小さい値です。
 - NUMNULLS - ビューの結果における列の NULL 値の数です。
 - SUB_COUNT - ビューの結果における列の平均のサブエレメントの数です。
 - SUB_DELIM_LENGTH - サブエレメントを分離する各区切り文字の平均長です。
- 列分散統計 (SYSCAT.COLDIST、SYSSTAT.COLDIST)
 - DISTCOUNT - COLVALUE 統計値以下の特殊な変位値の数です。
 - SEQNO - 表内の行を一意的に識別することを助けるためのシーケンス番号の頻度ランキングです。
 - COLVALUE - 頻度または変位値統計を収集する対象となるデータ値です。
 - VALCOUNT - ビュー列においてデータ値が発生する頻度、または変位値に対しては、データ値 (COLVALUE) 以下の値の数。

データ分布を表現しない統計 (NPAGES、および FPAGES など) は収集されますが、オプティマイザーでは無視されます。

シナリオ: 統計ビューを使用してカーディナリティー推定値を向上させる

データウェアハウスでは、ディメンション表データは静的ですが、ファクト表情報はしばしば大きく動的に変化します。これは、次元属性データがファクト表属性データと正または負の相関関係にある可能性を意味します。

現在、オプティマイザーで使用可能な従来の基本表統計では、表間のリレーションシップを識別することは許可されていません。統計ビュー (および MQT) 上での列と表の分散統計を使用して、オプティマイザーに必要な情報を提供し、これらのタイプのカーディナリティー推定エラーを修正することができます。

各年の 7 月期に販売したゴルフ・クラブについて、年間の売り上げを計算する次の照会を考慮します。

```
select sum(f.sales_price), d2.year
  from product d1, period d2, daily_sales f
 where d1.prodkey = f.prodkey
```

```

and d2.perkey = f.perkey
and d1.item_desc = 'golf club'
and d2.month = 'JUL'
group by d2.year

```

オプティマイザーが、PRODUCT および DAILY_SALES を含む半結合、または PERIOD や DAILY_SALES を含む半結合が最も選択的かどうかを判別できるのであれば、場合によっては、この照会ではスター型結合照会の実行プランが最も選択的です。効率的なスター型結合プランを生成するには、オプティマイザーが、索引 ANDing 操作の外部レグのために最も選択的な半結合を選択できる必要があります。

データウェアハウスに、在庫がなくなった製品のレコードが含まれることが多くあります。これが原因で、結合後の PRODUCT 列の分布が結合前の分布と大きく異なって表示されることがあります。高精度の情報が欠落しているため、オプティマイザーは基本表の統計のみに基づいてローカル述部の選択度を判別します。そのため、item_desc = 'golf club' 述部の選択度に関して、オプティマイザーが過度に楽観的になる可能性があります。

例えば、ゴルフ・クラブが従来製造された製品の 1% に相当し、かつ最近の売り上げの 20% を占める場合、オプティマイザーは、item_desc = 'golf club' の選択度を過剰に推定します。結合後の item_desc の分布記述統計がないからです。また、セールスが 12 カ月がすべて同程度だと思われる場合、month = 'JUL' 述部の選択度は約 8% 前後です。したがって、item_desc = 'golf club' 述部の選択度推定におけるエラーは、オプティマイザーが PRODUCT と DAILY_SALES の半結合を、スター型結合プランの索引 ANDing 操作の外部レグとして見かけ上、より選択的に実行する誤りの原因となります。

次の例は、このタイプの問題を解決するための、統計ビューのセットアップ方法を段階的に説明しています。

STORE、CUSTOMER、PRODUCT、PROMOTION、および PERIOD がディメンション表にあり、DAILY_SALES がファクト表にある典型的なデータウェアハウスのデータベースについて考えます。以下の表では、こうした表の定義を示します。

表 62. STORE (63 行)

列	storekey	store_number	city	状態	district	...
属性	integer 非 NULL 主キー	char(2)	char(20)	char(5)	char(14)	...

表 63. CUSTOMER (1 000 000 行)

列	custkey	name	address	age	gender	...
属性	integer 非 NULL 主キー	char(30)	char(40)	smallint	char(1)	...

表 64. PRODUCT (19 450 行)

列	prodkey	category	item_desc	price	cost	...
属性	integer 非 NULL 主キー	integer	char(30)	decimal(11)	decimal(11)	...

表 65. PROMOTION (35 行)

列	promokey	promotype	promodesc	promovalue	...
属性	integer 非 NULL 主キー	integer	char(30)	decimal(5)	...

表 66. PERIOD (2922 行)

列	perkey	calendar_date	month	period	year	...
属性	integer 非 NULL 主キー	date	char(3)	smallint	smallint	...

表 67. DAILY_SALES (754,069,426 行)

列	storekey	custkey	prodkey	promokey	perkey	sales_price	...
属性	integer	integer	integer	integer	integer	decimal(11)	...

再び訪問した消費者に対して割引を提案した場合、その製品を再び買う可能性について、会社のマネージャーが調査すると想定します。さらに、国内に 18 のロケーションがある「01」ストアにのみ調査をします。表 68 は、使用可能なプロモーションの異なるカテゴリーに関する情報を示しています。

表 68. PROMOTION (35 行)

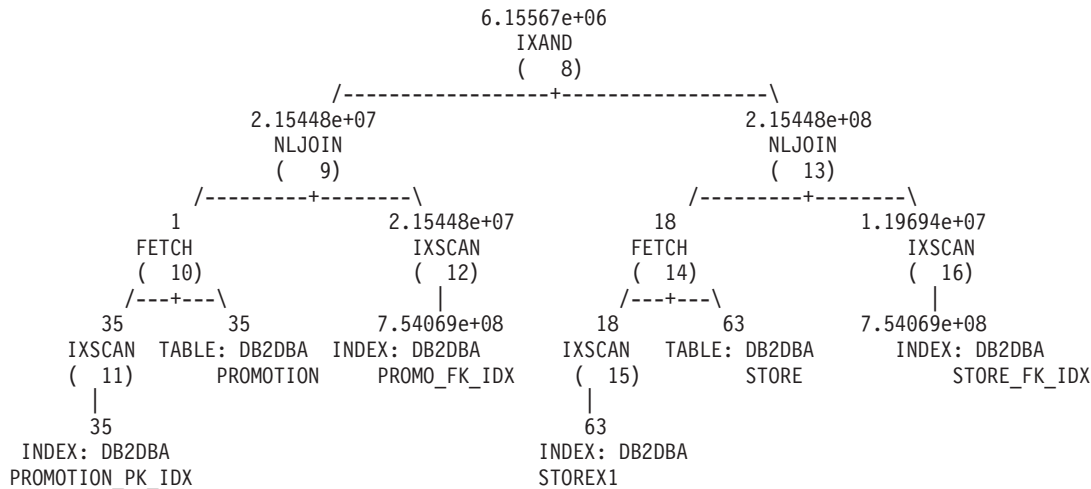
promotype	promodesc	COUNT (promotype)	合計割合
1	登録済みのお客様	1	2.86%
2	クーポン	15	42.86%
3	広告	5	14.29%
4	マネージャーのお勧め	3	8.57%
5	過剰在庫アイテム	4	11.43%
6	エンド・アイル・ディ スプレイ	7	20.00%

この表は、登録済みのお客様の割引が、提供された 35 種類のプロモーションのわずか 2.86% に相当することを示しています。

以下の照会により、カウント 12 889 514 が戻ります。

```
select count(*)
  from store d1, promotion d2, daily_sales f
 where d1.storekey = f.storekey
       and d2.promokey = f.promokey
       and d1.store_number = '01'
       and d2.promotype = 1
```


この照会は、オブティマイザーによって生成された次のプランに従って実行されます。この図の各ノードにおいて、最初の行はカーディナリティー推定値を、2 番目の行は演算子タイプを、3 行目 (括弧中の番号) は演算子 ID を表しています。



ネストされたループ結合 (番号 9) において、オブティマイザーは、販売商品の約 2.86% が割引価格で同じ商品を買ったお客様が戻ってきた結果であると推定します ($2.15448e+07 \div 7.54069e+08 \approx 0.0286$)。これは、PROMOTION 表と DAILY_SALES 表を結合する前後で同じ値であることに注意してください。表 69 は、結合の前後のカーディナリティー推定値とその割合 (フィルタリング効果) を要約します。

表 69. カーディナリティーは、DAILY_SALES との結合の前後を推定します。

述部	結合前		結合後	
	カウント	行修飾の割合	カウント	行修飾の割合
store_number = '01'	18	28.57%	2.15448e+08	28.57%
promotype = 1	1	2.86%	2.15448e+07	2.86%

promotype = 1 の確率は、store_number = '01' の確率より小さいので、オブティマイザーは PROMOTION と DAILY_SALES 間の半結合をスター型結合プランの索引 ANDing 操作の外部レグとして選択します。これにより、タイプ 1 のプロモーションが使用され、商品販売の推定カウントが約 6 155 670 となります。これは不正確なカーディナリティー推定で、実際の数はこの数の 2.09 倍です ($12\ 889\ 514 \div 6\ 155\ 670 \approx 2.09$)。

オブティマイザーが、2 つの述部を満たすレコードの実際数の半分しか推定しないのはなぜでしょうか。「01」ストアは全ストアの約 28.57% に相当します。他のストアが、「01」ストアより多くの売り上げがある場合はどうでしょうか (28.57% 未満の場合)。あるいは、もし「01」ストアが製品の大部分を売り上げた場合はどうでしょうか (28.57% より多い場合)。同様に、表 69 に示されるタイプ 1 のプロモーションを利用した商品販売の 2.86% は誤りの原因になる可能性があります。DAILY_SALES の実際の割合は推定された数と大きく異なる場合があります。

オブティマイザーが見積もりを訂正するために統計ビューを使用することができます。最初に前述の照会内の各半結合を表す 2 つの統計ビューを作成する必要があります。最初の統計ビューは、すべての日常販売活動におけるストアの分布を提供し

ます。2番目の統計ビューは、すべての日常販売活動におけるプロモーション・タイプの分布を表します。各統計ビューが特定のストア・ナンバーやプロモーション・タイプに関する分布情報を提供できるという点に注意してください。この例では、10%のサンプル率を利用して、各ビューのための DAILY_SALES のレコードを取得し、グローバルな一時表に保管しています。それから、必要な統計を収集するためにこれらの表を照会し、2つの統計ビューを更新します。

1. STORE と DAILY_SALES の結合を表すビューを作成します。

```
create view sv_store_dailysales as
(select s.*
 from store s, daily_sales ds
 where s.storekey = ds.storekey)
```

2. PROMOTION と DAILY_SALES の結合を表すビューを作成します。

```
create view sv_promotion_dailysales as
(select p.*
 from promotion.p, daily_sales ds
 where p.promokey = ds.promokey)
```

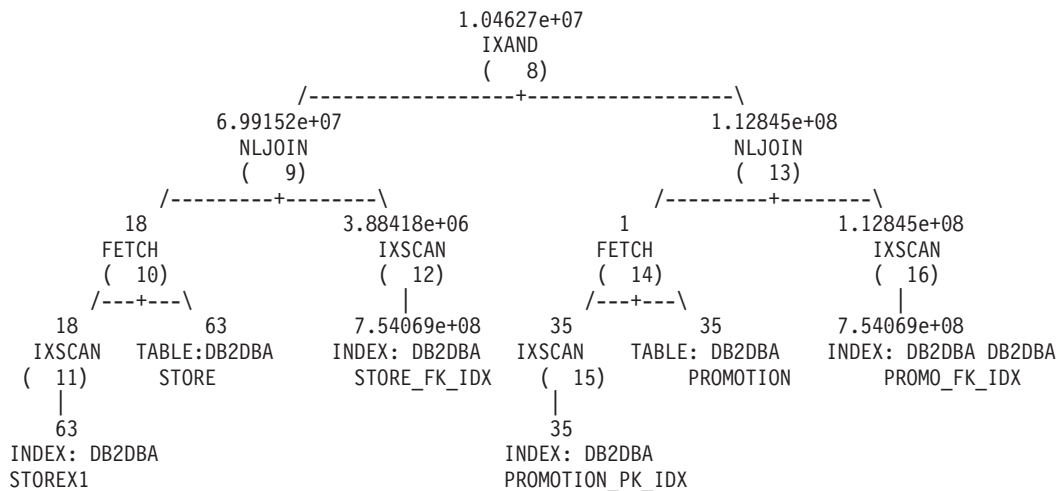
3. 照会最適化を使用可能にしてビューを統計ビューにします。

```
alter view sv_store_dailysales enable query optimization
alter view sv_promotion_dailysales enable query optimization
```

4. RUNSTATS コマンドを実行して、ビュー上の統計を収集します。

```
runstats on table db2dba.sv_store_dailysales with distribution
runstats on table db2dba.sv_promotion_dailysales with distribution
```

5. 再び照会を実行し、再最適化できるようにします。再最適化の際に、オペティマイザーは、SV_STORE_DAILYSALES と SV_PROMOTION_DAILYSALES をこの照会と突き合わせ、ビュー統計を使用してファクト表とディメンション表間の半結合のカーディナリティー推定を調整します。これにより、これらの統計なしで選択された半結合の元の順序を逆転させます。新規プランは次のとおりです。



463 ページの表 70 は、半結合ごとの結合の前後のカーディナリティー推定値とその割合 (フィルタリング効果) を要約します。

表 70. カーディナリティーは、DAILY_SALES との結合の前後を推定します。

述部	結合前		結合後 (統計ビューなし)		結合後 (統計ビューあり)	
	カウント	行修飾の割合	カウント	行修飾の割合	カウント	行修飾の割合
store_number = '01'	18	28.57%	2.15448e+08	28.57%	6.99152e+07	9.27%
promotype = 1	1	2.86%	2.15448e+07	2.86%	1.12845e+08	14.96%

今回は STORE と DAILY_SALES 間の半結合が索引 ANDing プランの外部レグ上で実行されていることに注意してください。2つの統計ビューは、本質的にオプティマイザーに対して store_number = '01' の述部は promotype = 1 の述部より多くの行をフィルタリングするように指示するからです。今回オプティマイザーは、約 10 462 700 の商品販売があると見積もります。この推定の場合の誤差は 1.23 倍 (12 889 514 ÷ 10 462 700 ≈ 1.23) になり、これは 461 ページの表 69 の統計ビューなしの推定に比べて大きな改善です。

統計ビューの式列から使用される統計

DB2 照会オプティマイザーは、統計ビューの式列からの統計を使用して正確なカーディナリティー推定値を取得し、より良いアクセス・プランを作成することができます。

例えば、UCASE スカラー関数を使用する照会を考えてみましょう。

```
SELECT * FROM t1, t2 WHERE t2.c2 = UCASE(t1.c1)
```

DB2 照会オプティマイザーは、このようなタイプの照会について統計ビューから得られた統計を使用して、より良いアクセス・プランを取得します。

前述の照会例では、列 c1 に UCASE スカラー関数を含む統計ビューが役に立ちます。以下に例を示します。

```
CREATE VIEW sv AS (SELECT UCASE(c1) AS c1 FROM t1)
ALTER VIEW sv ENABLE QUERY OPTIMIZATION
RUNSTATS ON TABLE dba.sv WITH DISTRIBUTION
```

このようなタイプの照会に統計を取得するには、述部の片側が、統計ビューの列定義の式と正確に一致する式でなければなりません。

以下に、DB2 照会オプティマイザーが統計ビューからの統計を使用しない例をいくつか挙げてみます。

- 照会の述部の片側が、統計ビューの複数の式列と一致する式であるもの

```
create view SV14(c1, c2) as (select c1+c2, c1*c2 from t1 where c1 > 3);
alter view SV14 enable query optimization;
runstats on table schema.sv1 with distribution;
select * from t1 where (c1+c2) + (c1*c2) > 5 and c1 > 3;
```

この式 (c1+c2) + (c1*c2) は、ビュー SV14 の 2 つの列と一致します。この式の場合、ビュー SV14 の統計は使用されません。

- 照会の述部の片側が、統計ビューの 1 つの式列と部分一致する式であるもの

```
create view SV15(c1, c2) as (select c1+c2, c1*c2 from t1 where c1 > 3);
alter view SV15 enable query optimization;
runstats on table schema.SV15 with distribution;
select * from t1 where (c1+c2) + 10 > 5 and c1 > 3;
```

この式 $(c1+c2) + 10$ は、ビュー SV15 の $c1+c2$ と部分一致します。この式の場合、ビュー SV15 の統計は使用されません。

- 照会の述部の片側が、統計ビューの 1 つの式列に間接的に一致するもの

```
create view SV16(c1, c2) as (select c1+c2, c1*c2 from t1 where c1 > 3);
alter view SV16 enable query optimization;
runstats on table schema.SV16 with distribution;
select * from t3 left join table (select ta.c1 from t2 left join table
  (select c1+c2,c3 from t1 where c1 > 3) as ta(c1,c3) on t2.c1 = ta.c3) as
  tb(c1) on t3.c1= TB.C1;
```

この列 TB.C1 は、ビュー SV16 の式 $c1+c2$ に間接的に一致しています。この式の場合、ビュー SV16 の統計は使用されません。

参照整合性制約により統計ビューの数を削減する

統計ビューを使用すると DB2 照会オプティマイザーに良好なカーディナリティー推定値を取得できますが、保守と処理にはリソースも必要です。しかし、参照整合性制約を使用することにより、統計の収集に使用する統計ビューの数を減らすことができます。

さまざまな表のセットを結合するスター型結合照会に正確な統計を収集しようとする場合、参照整合性制約を使用して、複数の統計ビューまたは 1 つの統計ビューを作成することができます。

例えば、表 D1、D2、D3、D4、D5、および F があるスキーマの場合を考えてみましょう。

```
create table D1 (D1_PK int not null primary key,
                D1_C1 int,
                D1_C2 int,
                D1_C3 int,
                D1_FK int not null);
create table D2 (D2_PK int not null primary key,
                D2_C1 int,
                D2_C2 int,
                D2_C3 int,
                D2_FK int not null);
create table D3 (D3_PK int not null primary key,
                D3_C1 int,
                D3_C2 int,
                D3_C3 int,
                D3_FK int not null);
create table D4 (D4_PK int not null primary key,
                D4_C1 int,
                D4_C2 int,
                D4_C3 int,
                D4_FK int not null);
create table D5 (D5_PK int not null primary key,
                D5_C1 int,
                D5_C2 int,
                D5_C3 int,
                D5_FK int not null);
create table F (F_FK1 int not null,
                F_FK2 int not null,
                F_FK3 int not null,
                F_FK4 int not null,
                F_FK5 int not null,
                F_C1 int, F_C2 int);
```

表 F を次のようにして変更することを検討します。

```

alter table F add foreign key (F_FK1)
  references D1 (D1_PK) on delete cascade;
alter table F add foreign key (F_FK2)
  references D2 (D2_PK) on delete cascade;
alter table F add foreign key (F_FK3)
  references D3 (D3_PK) on delete cascade;
alter table F add foreign key (F_FK4)
  references D4 (D4_PK) on delete cascade;
alter table F add foreign key (F_FK5)
  references D5 (D5_PK) on delete cascade;

```

また、次の照会に対する統計を提供する場合を考えてみましょう。

```

select distinct * from F, D1, D2, D3 where F_FK1 = D1_PK and F_FK2
  = D2_PK and F_FK3 = D3_PK and D1_C1='ON' and D2_C2>='2009-01-01';

```

次のようにすると、正確な統計を収集するために、ビューの完全セットを作成することができます。

```

create view SV1 as (select D1.* from F, D1 where F_FK1 = D1_PK);
alter view SV1 enable query optimization;

```

```

create view SV2 as(select D2.* from F, D2 where F_FK2 = D2_PK);
alter view SV2 enable query optimization;

```

```

create view SV3 as(select D3.* from F, D3 where F_FK3 = D3_PK);
alter view SV3 enable query optimization;

```

```

create view SV4 as(select D1.*, D2.*, D3.* from F, D1, D2, D3 where
  F_FK1 = D1_PK and F_FK2 = D2_PK and F_FK3 = D3_PK);
alter view SV4 enable query optimization;

```

結合列同士の間参照整合性制約が存在する場合、正確な統計を取得するために作成する統計ビューの数を減らすことができます。必要な統計ビューの数を減らすことにより、統計ビューの作成、更新、保守にかかる時間が短縮されます。前述の例で作成した統計ビューの完全セットと同じ統計を取得するためには、次の例に示す単一の統計ビューで十分です。

```

create view SV5 as (select D1.*, D2.*, D3.*, D4.*, D5.* from F, D1, D2, D3, D4, D5
  where
    F_FK1 = D1_PK and F_FK2 = D2_PK and F_FK3 = D3_PK
    and F_FK4 = D4_PK and F_FK5 = D5_PK
);
alter view SV5 enable query optimization;

```

SV4、SV3、SV2、および SV1 の統計は、参照整合性制約に基づいて SV5 から推定されます。F、D1、D2、D3、D4、および D5 の間の参照整合性制約により、これらの間の結合に欠落がなくなります。欠落のないこれらの結合により、次のカーディナリティーが同じであると推定できます。

- F と D1 の間の結合結果のカーディナリティー
- SV1 のカーディナリティー
- 統計ビュー SV5 のカーディナリティー

F と D2 の間、F と D3 の間、および F と D4 の間の結合にも、同じ推論が当てはまります。この推論は、前述の照会に対して、改善されたカーディナリティー推定値を DB2 オプティマイザーが取得するために必要なのは、SV5 というただ 1 つの統計ビューであることを示しています。

NULL 値を許可する参照整合性制約では、外部キー値における NULL 値の割合は、統計ビューから推定される統計に影響を及ぼします。

NULL 値が外部キー値の大きな割合を占める場合、DB2 オプティマイザーが制約を使用して推定する統計とカーディナリティー推定値が不正確になることがあります。統計の推定に参照整合性制約が使用されないようにして、このような不正確性を防止します。統計の推定に制約が使用されないようにするには、ALTER TABLE コマンドの DISABLE QUERY OPTIMIZATION オプションを使用して、照会の最適化を無効にします。

このような状況は、Explain 機能を使用して検出できます。NULL 値の数が多い、かつ統計が統計ビューからの推定である場合、Explain 機能は Explain 診断メッセージを生成して、外部キーにおける NULL 値の比率を表示します。

統計ビュー上での列グループ統計から使用される統計

1 つの照会に対して統計の収集が必要な複雑なシナリオでは、統計ビューを作成したり、表で列グループ統計を収集するだけでは不十分です。統計ビュー上で列グループ統計を収集するには、場合によってはこの 2 つを組み合わせる必要があります。統計ビューと列グループ統計を組み合わせると、DB2 オプティマイザーがより良いアクセス・プランを生成するのに役立ちます。

普通、統計ビュー上での列グループ統計の使用は、統計ビューが通常処理する一般的な状況以外の相関がさらにある場合に発生します。

例えば、次の照会と統計ビューについて考えてみましょう。

```
select * from T1,T2 where T1.C1=T2.D1 and T1.C2=5 and T2.D3=10;
create view SV2 as (select * from T1,T2 where T1.C1=T2.D1);
alter view SV2 enable query optimization;
runstats on table db2.SV2 on all columns with distribution;
```

この照会の実行速度は遅く、カーディナリティー推定値は不正確である可能性があります。アクセス・プランを調べると、統計ビューによりカーディナリティー推定値は調整されたものの、T1.C2 と T2.D3 の間に強い相関があることがわかります。したがって、カーディナリティーは不正確になります。

この状況を解消するために、次のコマンドを発行して、ビュー SV2 の列グループ統計を収集することができます。

```
runstats on table db2.SV2 on all columns and columns((C2,D3)) with distribution;
```

これらの追加統計によりカーディナリティー推定値が改善され、より良いアクセス・プランが生成されます。

注: 統計ビューにおける列のリストに関する分散統計の収集はサポートされていません。

統計ビュー上での列グループ統計の収集は、データを一定の方法でグループ化する必要のある照会について、個々のグループ化の数、すなわちグループ化キー・カーディナリティーを計算するためにも使用できます。グループ化要件は、GROUP BY または DISTINCT などの操作の結果として生じます。

例えば、次の照会と統計ビューについて考えてみましょう。

```
select T1.C1, T1.C2 from T1,T2 where T1.C3=T2.C3 group by T1.C1, T1.C2;
create view SV2 as (select T1.C1, T1.C2 from T1,T2 where T1.C3=T2.C3);
alter view SV2 enable query optimization;
```

結合述部を対象とした統計ビュー上で列グループ統計を収集すると、オプティマイザーがグループ化キー・カーディナリティーをより正確に見積もる上で役に立ちます。列グループ統計を収集するには、次のコマンドを発行します。

```
runstats on table db2.SV2 on columns((C1,C2));
```

カタログ統計

照会コンパイラーが照会プランを最適化する際には、データベースの表、索引、および統計ビューのサイズに関する統計情報が、コンパイラーの判断にかなり影響します。この情報は、システム・カタログ表に保管されます。

さらに、表、索引、および統計ビューの特定の列が行の選択や表の結合に使用される場合、オプティマイザーはそれらの列でのデータ分布情報も使用します。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。

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、表または索引を再編成する時期を決定するために使用できます。

パーティション・データベース環境では、表の統計は、ユーティリティーが実行されるデータベース・パーティション上にある表の部分についてのみ収集されるか、表を含むデータベース・パーティション・グループ内の最初のデータベース・パーティションについてのみ収集されます。統計ビューについての情報は、全データベース・パーティションについて収集されます。

runstats ユーティリティーによって更新される統計

カタログ統計は runstats ユーティリティーによって収集されます。このユーティリティーは、**RUNSTATS** コマンドを発行するか、ADMIN_CMD プロシージャーを呼び出すか、または db2Runstats API を呼び出すことによって開始できます。更新は、手動と自動のいずれでも開始できます。

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: カタログ統計の収集. タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

宣言済み一時表に関する統計はシステム・カタログに保管されるのではなく、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。宣言済み一時表に対して runstats を実行することは可能です (状況によってはこれが役立つ場合があります)。

表および索引に関して、runstats ユーティリティーは以下の情報を収集します。

- 行を含んでいるページの数
- 使用中のページの数
- 表内の行数 (カーディナリティー)
- オーバーフローしている行数

- マルチディメンション・クラスタリング (MDC) 表、および挿入時クラスタリング (ITC) 表の場合は、データを含んでいるブロックの数
- パーティション表の場合、単一のデータ・パーティション内でのデータ・クラスタリングの程度
- データ分散統計。オプティマイザーはこれを使用して、データが均等に分布していない表および統計ビュー、および列に膨大な数の重複値がある表および統計ビューに関して効果的なアクセス・プランを見積もります。
- 詳細な索引統計。オプティマイザーはこれを使用して、索引を介して表のデータにアクセスするときの効率性を判別します。
- LIKE 述部でのサブエレメント統計、特にストリング内のパターンを検索するもの (LIKE %disk% など) も、オプティマイザーによって使用されます。

表のそれぞれのデータ・パーティションごとに、runstats ユーティリティーは以下の統計を収集します。これらの統計は、パーティションが再編成される必要があるかどうかを判別するためにのみ使用されます。

- 行を含んでいるページの数
- 使用中のページの数
- 表内の行の数 (カーディナリティー)
- オーバーフローしている行の数
- MDC 表および ITC 表の場合、データを含んでいるブロックの数

以下の場合、分散統計は収集されません。

- データベース構成パラメーター **num_freqvalues** および **num_quantiles** を 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。例えば、各データ値が固有である場合など
- 列に LONG、LOB、または構造化データ型が含まれる場合
- 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 NULL 以外の値が 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

表または統計ビューの各列、および索引キーの最初の列に関して、runstats ユーティリティーは以下の情報を収集します。

- 列のカーディナリティー
- 列の平均長 (列がデータベース・メモリーまたは一時表に格納される場合に必要となる平均スペース (バイト単位))
- 列内で 2 番目に高い値
- 列内で 2 番目に低い値
- 列内の NULL 値の数

ラージ・オブジェクト (LOB) または LONG データ・タイプが含まれる列では、runstats ユーティリティーは、列の平均の長さとして列に含まれる NULL 値の数のみを収集します。列の平均長は、LOB データがデータ・ページ上のインラインに置かれている場合を除き、データ記述子の長さを表します。ディスク上に列を格納するために必要な平均スペース量は、この統計値とは異なる場合があります。

それぞれの XML 列に関して、runstats ユーティリティーは以下の情報を収集します。

- NULL の XML 文書の数
- 非 NULL の XML 文書の数
- 異なるパスの数
- 異なるパスごとのノード・カウン트의合計
- 異なるパスごとの文書数の合計
- 最大のノード・カウントを持つ (パス、ノード・カウント) の k ペア
- 最大の文書数を持つ (パス、文書数) の k ペア
- 最大のノード・カウントを持つ (パス、値、ノード・カウント) の k トリプル
- 最大の文書数を持つ (パス、値、文書数) の k トリプル
- テキストまたは属性値へ導く異なるパスごとに、以下のとおりです。
 - このパスが取る可能性のある別個の値の数
 - 最大値
 - 最小値
 - テキストまたは属性ノードの数
 - テキストまたは属性ノードを含む文書の数

XML 列の各行は、XML 文書を保管します。パスまたはパスと値のペアのノード・カウントは、そのパスまたはパスと値のペアによって到達可能なノードの数のことです。パスまたはパスと値のペアの文書数は、そのパスまたはパスと値のペアを含む文書の数のことです。

DB2 V9.7 フィックスパック 1 以降のリリースでは、以下の事柄が XML 列の分散統計の収集に適用されます。

- 分散統計は、XML 列で指定された XML データの索引ごとに収集されます。
- runstats ユーティリティーが XML データの索引で分散統計を収集するには、分散統計と表統計の両方を収集しなければなりません。XML 分散統計は表統計と一緒に格納されるため、分散統計を収集するには表統計を収集する必要があります。

索引統計だけを収集したり、索引作成時に索引統計を収集したりしても、XML データの索引の分散統計は収集されません。

デフォルトでは、runstats ユーティリティーによって、XML データの索引ごとに分散統計の最大で 250 の変位値が収集されます。列の変位値の最大数は、runstats ユーティリティーの実行時に指定できます。

- 分散統計は、タイプ VARCHAR、DOUBLE、TIMESTAMP、および DATE の XML データの索引に対して収集されます。タイプ VARCHAR HASHED の XML データの索引に対しては、XML 分散統計は収集されません。
- 自動表の runstats 操作の実行時に、XML 分散統計が収集されます。
- STATISTICS オプションを使用してデータをロードする場合には、XML 分散統計は作成されません。
- パーティション表で定義された XML データのパーティション索引の場合、XML 分散統計は収集されません。

列グループに関して、runstats ユーティリティーは以下の情報を収集します。

- 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

索引に関して、runstats ユーティリティーは以下の情報を収集します。

- 索引項目の数 (索引のカーディナリティー)
- リーフ・ページの数
- 索引レベルの数
- 索引への表データのクラスタリングの程度
- データ・パーティションに関しての索引キー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にあるリーフ・ページの数比率
- 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
- 平均のリーフ・キーのサイズ (組み込み列なし)
- 平均のリーフ・キーのサイズ (組み込み列あり)
- すべてのレコード ID (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 だけ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

詳細な索引統計を要求した場合、索引に対する表データのクラスタリングの程度に関する追加情報、およびさまざまなバッファ・サイズに関するページ・フェッチの見積もりが収集されます。

パーティション索引の場合、これらの統計は単一の索引パーティションを表します。ただし、索引の最初の列の固有値、索引の最初の 2 つ、3 つ、および 4 つの列の固有値、および索引のすべての列の固有値は例外です。索引パーティションごとの統計も、索引パーティションが再編成される必要があるかどうかを判別する目的で収集されます。

統計収集を行うと、統計が収集された表を参照するキャッシュ済み動的ステートメントは無効になります。このようにすることで、キャッシュに入れられた動的ステートメントを最新の統計で再最適化できます。

カタログ統計の表

データベースの表、索引、および統計ビューのサイズに関する統計情報が、システム・カタログ表に格納されます。

以下の表では、この統計情報の簡単な説明と、その格納場所を示します。

- 『表』列は、**RUNSTATS** コマンドに **FOR INDEXES** または **AND INDEXES** パラメーターが指定されていない場合に、特定の統計が収集されるかどうかを示します。
- 『索引』列は、**FOR INDEXES** または **AND INDEXES** パラメーターが指定されている場合に、特定の統計が収集されるかどうかを示します。

統計によって、表のみで提供されるもの、索引のみで提供されるもの、および両方で提供されるものがあります。

- 表 1. 表統計 (SYSCAT.TABLES および SYSSTAT.TABLES)
- 表 2. 列統計 (SYSCAT.COLUMNS および SYSSTAT.COLUMNS)
- 表 3. 複数列統計 (SYSCAT.COLGROUPS および SYSSTAT.COLGROUPS)
- 表 4. 複数列分散統計 (SYSCAT.COLGROUPDIST および SYSSTAT.COLGROUPDIST)
- 表 5. 複数列分散統計 (SYSCAT.COLGROUPDISTCOUNTS および SYSSTAT.COLGROUPDISTCOUNTS)
- 表 6. 索引統計 (SYSCAT.INDEXES および SYSSTAT.INDEXES)
- 表 7. 列分散統計 (SYSCAT.COLDIST および SYSSTAT.COLDIST)

表 4. 複数列分散統計 (SYSCAT.COLGROUPDIST および SYSSTAT.COLGROUPDIST) と 表 5. 複数列分散統計 (SYSCAT.COLGROUPDISTCOUNTS および SYSSTAT.COLGROUPDISTCOUNTS) のリストにある複数列分散統計は、**RUNSTATS** ユーティリティーでは収集されません。それらを手動で更新することはできません。

表 71. 表統計 (SYSCAT.TABLES および SYSSTAT.TABLES)

統計	説明	RUNSTATS オプション	
		表	索引
FPAGES	表が使用しているページの数	はい	はい
NPAGES	行が含まれているページの数	はい	はい
OVERFLOW	オーバーフローしている行の数	はい	いいえ
CARD	表内の行の数 (カーディナリティー)	はい	はい (注 1)
ACTIVE_BLOCKS	MDC 表の場合、占有されているブロックの合計数	はい	いいえ
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 72. 列統計 (SYSCAT.COLUMNS および SYSSTAT.COLUMNS)

統計	説明	RUNSTATS オプション	
		表	索引
COLCARD	列カーディナリティー	はい	はい (注 1)
AVGCOLLEN	列の平均長	はい	はい (注 1)
HIGH2KEY	列内で 2 番目に高い値	はい	はい (注 1)
LOW2KEY	列内で 2 番目に低い値	はい	はい (注 1)
NUMNULLS	列内の NULL 値の数	はい	はい (注 1)

表 72. 列統計 (SYSCAT.COLUMNS および SYSSTAT.COLUMNS) (続き)

統計	説明	RUNSTATS オプション	
		表	索引
SUB_COUNT	サブ要素の平均数	はい	いいえ (注 2)
SUB_DELIM_LENGTH	サブ要素を分ける各区切り文字の平均長	はい	いいえ (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要素を含む列の、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、1 バイト文字セット (SBCS) のコード・ページ属性、FOR BIT DATA、または UTF-8 のタイプ CHAR および VARCHAR の列の場合だけです。			

表 73. 複数列統計 (SYSCAT.COLGROUPS および SYSSTAT.COLGROUPS)

統計	説明	RUNSTATS オプション	
		表	索引
COLGROUPCARD	列グループのカードィナリティー	はい	いいえ

表 74. 複数列分布統計 (SYSCAT.COLGROUPDIST および SYSSTAT.COLGROUPDIST)

統計	説明	RUNSTATS オプション	
		表	索引
TYPE	F = 頻出値 Q = 変位値	はい	いいえ
ORDINAL	グループ内の列の序数	はい	いいえ
SEQNO	n 番目の TYPE 値を表す、シーケンス番号 n 。	はい	いいえ
COLVALUE	文字リテラルまたは NULL 値としてのデータ値	はい	いいえ

表 75. 複数列分布統計 (SYSCAT.COLGROUPDISTCOUNTS および SYSSTAT.COLGROUPDISTCOUNTS)

統計	説明	RUNSTATS オプション	
		表	索引
TYPE	F = 頻出値 Q = 変位値	はい	いいえ
SEQNO	n 番目の TYPE 値を表す、シーケンス番号 n 。	はい	いいえ

表 75. 複数列分布統計 (SYSCAT.COLGROUPDISTCOUNTS および
SYSSTAT.COLGROUPDISTCOUNTS) (続き)

統計	説明	RUNSTATS オプション	
		表	索引
VALCOUNT	TYPE = F の場合、 VALCOUNT は、こ の SEQNO を持つ列 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。	はい	いいえ
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。	はい	いいえ

表 76. 索引統計 (SYSCAT.INDEXES および SYSSTAT.INDEXES)

統計	説明	RUNSTATS オプション	
		表	索引
NLEAF	索引リーフ・ページ の数	いいえ	はい
NLEVELS	索引レベルの数	いいえ	はい
CLUSTERRATIO	表データのクラスタリ ングの程度	いいえ	はい (注 2)
CLUSTERFACTOR	クラスタリングの度合 いの詳細	いいえ	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)	いいえ	はい
FIRSTKEYCARD	索引の最初の列の固有 値の数	いいえ	はい
FIRST2KEYCARD	索引の最初の 2 つの 列の固有値の数	いいえ	はい
FIRST3KEYCARD	索引の最初の 3 つの 列の固有値の数	いいえ	はい

表 76. 索引統計 (SYSCAT.INDEXES および SYSSTAT.INDEXES) (続き)

統計	説明	RUNSTATS オプション	
		表	索引
FIRST4KEYCARD	索引の最初の 4 つの列の固有値の数	いいえ	はい
FULLKEYCARD	索引のすべての列の固有値の数 (ただし、すべてのレコード ID (RID) が削除対象としてマークされている索引のキー値はすべて除く)	いいえ	はい
PAGE_FETCH_PAIRS	異なるバッファ・サイズでのページ取り出し見積もり	いいえ	詳細説明 (注 1、2)
AVGPARTITION_CLUSTERRATIO	単一のデータ・パーティション内でのデータ・クラスタリングの程度	いいえ	はい (注 2)
AVGPARTITION_CLUSTERFACTOR	単一のデータ・パーティション内での、クラスタリングの度合いのより精密な測定	いいえ	詳細説明 (注 1、2)
AVGPARTITION_PAGE_FETCH_PAIRS	単一のデータ・パーティションを基にして生成された異なるバッファ・サイズのページ取り出し見積もり	いいえ	詳細説明 (注 1、2)
DATAPARTITION_CLUSTERFACTOR	索引スキャン中のデータ・パーティション参照の数	いいえ (注 6)	はい (注 6)
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	いいえ	はい
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。	いいえ	はい
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	いいえ	はい
AVERAGE_SEQUENCE_GAP	シーケンス間のギャップ	いいえ	はい

表 76. 索引統計 (SYSCAT.INDEXES および SYSSTAT.INDEXES) (続き)

統計	説明	RUNSTATS オプション	
		表	索引
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。	いいえ	はい (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	いいえ	はい (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	いいえ	はい (注 4)
NUMRIDS	索引内の RID の数 (削除対象の RID を含む)	いいえ	はい
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 RID 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)	いいえ	はい
NUM_EMPTY_LEAFS	すべての RID 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	いいえ	はい
INDCARD	索引項目の数 (索引のカーディナリティー)	いいえ	はい

表 76. 索引統計 (SYSCAT.INDEXES および SYSSTAT.INDEXES) (続き)

統計	説明	RUNSTATS オプション	
		表	索引
注:			
<p>1. 詳細索引統計は、RUNSTATS コマンドに DETAILED 節を指定して収集します。</p> <p>2. 表のサイズが相当大きい (おおよそ 25 ページより大きい) ものでない限り、DETAILED 節を指定しても CLUSTERFACTOR および PAGE_FETCH_PAIRS は収集されません。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 ユーティリティーで CLUSTERRATIO だけが収集され、CLUSTERFACTOR および PAGE_FETCH_PAIRS は収集されません。DETAILED 節を指定しないと、CLUSTERRATIO だけが収集されます。</p> <p>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、DENSITY は 100 になるはずですが、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、オブティマイザーが見積もるのに使用されます。</p> <p>4. この統計は、表が DMS 表スペースにある場合は計算できません。</p> <p>5. プリフェッチ統計は、統計収集がロードまたは索引の作成コマンドの呼び出し時に指定されている場合でも、その操作の実行中には収集されません。プリフェッチ統計は、seqdetect データベース構成パラメーターが NO に設定されている場合も収集されません。</p> <p>6. 表の RUNSTATS オプションが「いいえ」なら、表統計の収集時に統計が収集されず、索引の RUNSTATS オプションが「はい」なら、INDEXES オプションを指定した RUNSTATS コマンドが使用されるときに統計が収集されます。</p>			

表 77. 列分散統計 (SYSCAT.COLDIST および SYSSTAT.COLDIST)

統計	説明	RUNSTATS オプション	
		表	索引
DISTCOUNT	TYPE = Q の場合、 DISTCOUNT は COLVALUE 統計以下の固有値の数	分散 (注 2)	いいえ
TYPE	行の統計が頻出値統計であるか変位値統計であるかの標識	分散	いいえ
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	分散	いいえ
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	分散	いいえ
VALCOUNT	列内でデータ値が発生する頻度。変位値の場合は、データ値 (COLVALUE) 以下の数値	分散	いいえ

表 77. 列分散統計 (SYSCAT.COLDDIST および SYSSTAT.COLDDIST) (続き)

統計	説明	RUNSTATS オプション	
		表	索引
注:			
1. 列分散統計は、 RUNSTATS コマンドに WITH DISTRIBUTION 節を指定することにより収集されます。列の値が十分に不均一でないかぎり、分散統計は収集できません。			
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。			

自動統計収集

DB2 オプティマイザーは、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。日付が古い、あるいは不完全な表または索引に関する統計を使用すると、オプティマイザーが最適ではないプランを選択するため、照会の実行がスローダウン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 の自動表保守フィーチャーの一部である自動統計収集を使えば、統計の更新が必要かどうかをデータベース・マネージャーに判断させることができます。自動統計収集は、リアルタイム統計 (RTS) フィーチャーを使用してステートメント・コンパイル時に同期的に行うこともできますし、**RUNSTATS** コマンドを使用して単にバックグラウンドで非同期的に収集することもできます。バックグラウンド統計収集は、リアルタイム統計収集が使用不可な場合にも有効にすることができますが、リアルタイム統計収集が生じるためにはバックグラウンド統計収集が有効になっていなければなりません。デフォルトでは、データベースの作成時には、自動バックグラウンド統計収集 **auto_runstats** と自動リアルタイム統計収集 **auto_stmt_stats** が有効になります。

DB2 バージョン 9 以降では、新しいデータベースの初期構成 (**auto_stmt_stats** データベース構成パラメーターの適切な設定値も含む) を決定するのに構成アドバイザーを使用できます。

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: 自動統計収集の構成. タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

非同期統計収集およびリアルタイム統計収集について

リアルタイム統計が使用可能である場合は、メタデータを使用して統計を作成することが可能です。ファブリケーションとは、通常の **RUNSTATS** コマンド・アクティビティーの一部として統計を収集するのではなく、統計を派生させること、または作成することです。例えば、表のページ数、ページ・サイズ、および平均の行幅が分かれば、表の行数を派生させることができます。場合によっては、統計は派生させられず、索引およびデータ・マネージャーによって保守され、カタログに直接保管されることもあります。例えば、索引マネージャーは索引ごとのリーフ・ページの数とレベルのカウントを保守します。

照会オプティマイザーは、照会のニーズおよび表更新アクティビティーの量 (更新、挿入、または削除操作の数) に基づいて統計を収集する方法を決定します。

リアルタイム統計収集の方がよりタイムリーで、より正確な統計となります。正確な統計を収集できると、照会実行プランをより良いものにできますし、パフォーマンスを向上させることが可能です。リアルタイム統計が使用可能かどうかに関わらず、2 時間のインターバルで非同期統計収集が行われます。アプリケーションによっては、正確な統計を出すために、このインターバルでは不十分な場合があります。

また、以下の場合にもリアルタイム統計収集の非同期収集要求が開始します。

- 表アクティビティーが同期収集を必要とするほど高くはないものの、非同期収集を十分必要とする高さである場合。
- 表が大きいために、同期統計収集でサンプリングを使用した場合。
- 同期統計を作成した場合。
- 同期統計収集の収集時間が超過したために失敗した場合。

非同期要求は多くて 2 つ同時に処理できますが、それは対象となる表が異なる場合に限りです。一方の要求はリアルタイム統計収集によって開始され、もう一方の要求は非同期統計収集の検査によって開始されます。

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。

- 非同期統計収集は、スロットル調整した **RUNSTATS** ユーティリティーを使用して実行されます。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。
- 同期統計収集は、照会 1 つにつき 5 秒に制限されています。この値は、RTS 最適化ガイドラインによって制御できます。同期収集が時間制限を超えると、非同期収集要求が送信されます。
- 同期統計収集では、統計はシステム・カタログ内に格納されません。代わりに、統計は統計キャッシュに格納され、後ほど非同期操作によってシステム・カタログ内に格納されます。この格納手順により、システム・カタログの更新時に関係したオーバーヘッドを回避でき、ロック競合の可能性も回避できます。統計キャッシュ内の統計は、後続の SQL コンパイル要求で使用できます。
- 1 つの表で行われる同期統計収集の操作は、1 つだけです。同期統計収集を必要とする他のエージェントは、可能な場合には統計を作成し、引き続きステートメント・コンパイルを行います。この動作は、パーティション・データベース環境でも強制されます。この環境では、異なるデータベース・パーティション上にあるエージェントが同期統計を必要とする場合があります。
- この収集する統計のタイプは、統計プロファイルを使用可能にすることによってカスタマイズできます。統計プロファイルでは、直前のデータベース・アクティビティーに関する情報を使用してデータベースのワークロードに必要な統計を決定します。また独自の統計プロファイルを特定の表に作成してカスタマイズすることもできます。

- 統計が欠落している表またはアクティビティーのレベルが高い表 (更新、挿入、および削除の操作の数で測る) だけを、統計収集の対象として考慮します。表が統計収集の基準を満たしている場合であっても、照会を最適化するために必要とされない限りは同期統計は収集されません。場合によっては、照会オプティマイザーは統計なしでアクセス・プランを選択することもできます。
- 非同期統計収集の検査に関しては、大きい表 (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。
- 非同期統計収集の場合、メンテナンス・ポリシーで指定されるオンライン保守時間枠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集合も指定するので、不必要なリソース消費をさらにおさえます。
- 同期統計収集および作成は、メンテナンス・ポリシーで指定されたオンライン保守時間枠には従いません。同期統計収集および作成は、自動統計収集の有効範囲内の表集合を指定するポリシーに従います。
- 自動統計収集の実行中も、影響を受ける表に対し、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、または削除の操作) が可能です。
- リアルタイム統計 (同期、または作成されたもの) は、ニックネームに関しては収集されません。同期統計収集の場合、システム・カタログ内でニックネーム統計をリフレッシュするには、**SYSPROC.NNSTAT** プロシージャーを呼び出します。非同期統計収集の場合、DB2 Database for Linux, UNIX, and Windows は自動的に **SYSPROC.NNSAT** プロシージャーを呼び出し、システム・カタログ内でニックネーム統計をリフレッシュします。
- リアルタイム統計 (同期、または作成されたもの) は、統計ビューに関しては収集されません。
- 宣言済み一時表 (DGTT) では、リアルタイム統計しか収集できません。

リアルタイム統計収集は統計収集のオーバーヘッドを最小化するように設計されていますが、まずテスト環境で試行してパフォーマンスに悪影響を与えないことを確かめてください。一部のオンライン・トランザクション処理 (OLTP) のシナリオ、特に照会実行時間に上限がある場合に、パフォーマンスに悪影響を与えることがあります。

リアルタイム同期統計収集の実行対象となるのは、通常表、マテリアライズ照会表 (MQT)、およびグローバル一時表です。グローバル一時表に関しては、非同期統計は収集されません。

自動統計収集 (同期または非同期) は、以下の対象に対しては行われません。

- **VOLATILE** というマークが付けられている表 (**SYSCAT.TABLES** カタログ・ビューに **VOLATILE** フィールドが設定されている表)
- 作成済みの一時表 (CGTT)
- **SYSSTAT** カタログ・ビューに対して **UPDATE** ステートメントを直接発行して、統計が手動で更新された表

表の統計を手動で変更すると、データベース・マネージャーは、変更したユーザーがその統計の保守を行うものと見なします。統計を手動で更新した表の統計

を、データベース・マネージャーが保守するようにするには、**RUNSTATS** コマンドを使用して統計を収集するか、**LOAD** コマンドの使用時に統計を収集するように指定します。バージョン 9.5 より前のバージョンで作成された表で、アップグレード前に統計を手動で更新した表は、影響を受けません。それらの表の統計は、統計を手動で更新するまで、データベース・マネージャーによって自動的に保守されます。

以下のものに対しては、統計作成は行われません。

- 統計ビュー
- **SYSSTAT** カタログ・ビューに対して **UPDATE** ステートメントを直接発行して、統計が手動で更新された表。ただし、リアルタイムの統計収集が有効にされていない場合は、統計が手動で更新された表に対して幾らかの統計作成が行われます。

パーティション・データベース環境では、統計は単一のデータベース・パーティションで収集されてから、外挿されます。データベース・マネージャーは、必ずデータベース・パーティション・グループの最初のデータベース・パーティションで統計 (同期および非同期の両方) を収集します。

データベースがアクティブにされてから最低 5 分間は、リアルタイム統計収集アクティビティは行われません。

リアルタイム統計プロセスは、静的および動的 SQL の両方で生じます。

TRUNCATE ステートメントまたは **IMPORT** コマンドを使用して切り捨てられた表は、失効した統計を持っているものとして自動的に認識されます。

自動統計収集 (同期と非同期の両方) は、統計収集の対象の表を参照する、キャッシュに入れられた動的ステートメントを無効にします。このようにすることで、キャッシュに入れられた動的ステートメントを最新の統計で再最適化できます。

データベースが非アクティブ化されると、非同期自動統計収集操作が中断する可能性があります。 **ACTIVATE DATABASE** コマンドまたは **API** を使用してデータベースを明示的に活動化しなかった場合、最後のユーザーがデータベースから切断したときに、データベースは非アクティブ化されます。操作が中断された場合、**DB2** 診断ログ・ファイルにエラー・メッセージが記録されることがあります。非同期自動統計収集操作が中断されないようにするには、データベースを明示的に活動化してください。

リアルタイム統計および Explain プロセス

Explain 機能を使用して **Explain** しただけの照会 (実行していないもの) に、リアルタイム処理はありません。以下の表では、**CURRENT EXPLAIN MODE** 特殊レジスターの種々の値の動作を要約しています。

表 78. **CURRENT EXPLAIN MODE** 特殊レジスターの値の関数としてのリアルタイム統計収集

CURRENT EXPLAIN MODE 値	リアルタイム統計収集を考慮するかどうか
YES	はい
EXPLAIN	いいえ

表 78. CURRENT EXPLAIN MODE 特殊レジスタの値の関数としてのリアルタイム統計収集 (続き)

CURRENT EXPLAIN MODE 値	リアルタイム統計収集を考慮するかどうか
NO	はい
REOPT	はい
RECOMMEND INDEXES	いいえ
EVALUATE INDEXES	いいえ

自動統計収集および統計キャッシュ

統計キャッシュは、同期的に収集された統計をすべての照会で使用できるようにするために、DB2 バージョン 9.5 で導入されました。このキャッシュは、カタログ・キャッシュの一部です。パーティション・データベース環境では、各データベース・パーティションにカタログ・キャッシュがありますが、統計キャッシュはカタログ・データベース・パーティションにのみ置かれます。リアルタイム統計収集が使用可能である場合は、カタログ・キャッシュの要件が高くなります。リアルタイム統計収集が使用可能である場合は、**catalogcache_sz** データベース構成パラメータ値を調整することを検討してください。

DB2 バージョン 9 以降では、新しいデータベースの初期構成を決定するのに構成アドバイザーを使用できます。構成アドバイザーは、**auto_stmt_stats** データベース構成パラメータを ON に設定することを推奨します。

自動統計収集および統計プロファイル

同期統計および非同期統計は統計プロファイル (事実上、表に対するもの) に基づいて収集されますが、以下の例外があります。

- 同期統計収集のオーバーヘッドを最小化するため、データベース・マネージャーはサンプリングを使用して統計を収集する場合があります。その場合のサンプリング率とサンプリング方法は、統計プロファイルで指定されているサンプリング率とサンプリング方法とは異なる可能性があります。
- 同期統計収集において統計を作成することを選択できますが、統計プロファイルで指定されている統計をすべて作成することは不可能な場合があります。例えば、COLCARD、HIGH2KEY、および LOW2KEY などの列統計を作成することは、列が索引と関連付けられていない場合には不可能です。

統計プロファイルで指定されているすべての統計を同期統計収集で収集できない場合、非同期収集要求がサブミットされます。

統計の自動収集を使用可能にする:

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フィーチャーを使用してください。

このタスクについて

単一データベース・パーティションがシングル・プロセッサで作動する環境では、この機能の拡張として、照会データを収集し、統計プロファイルを生成することができます。この機能により、DB2 サーバーはワークロードが必要とする十分な統計のセットを自動的に収集することができるようになります。このオプションはパーティション・データベース環境、特定のフェデレーテッド・データベース環境では使用できず、パーティション内並列処理が有効な環境でも使用できません。

自動統計収集を使用可能にするには、まずデータベースを構成して **auto_maint** および **auto_tbl_maint** データベース構成パラメーターを ON に設定する必要があります。

手順

auto_maint および **auto_tbl_maint** データベース構成パラメーターを ON に設定した後は、次の選択肢があります。

- バックグラウンド統計収集を使用可能にするには、**auto_runstats** データベース構成パラメーターを ON に設定します。
- 統計ビューに対するバックグラウンド統計収集を使用可能にするには、**auto_stats_views** および **auto_runstats** データベース構成パラメーターを両方とも ON に設定します。
- バックグラウンド統計収集で、大規模な表および統計ビューの自動サンプリングを使用可能にするには、**auto_sampling** データベース構成パラメーターも ON に設定します。この設定は、**auto_runstats** (表のみ) と一緒に、あるいは **auto_runstats** および **auto_stats_views** (表および統計ビュー) と一緒に使用します。
- リアルタイム統計収集を使用可能にするには、**auto_stmt_stats** および **auto_runstats** データベース構成パラメーターを、両方とも ON に設定します。

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、特定の表にどのタイプの統計を収集するかを指定します (例えば、表統計、索引統計、または分散統計)。このフィーチャーは、将来使用するために **RUNSTATS** オプションを格納できるようにして、統計収集を簡略化します。

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** パラメーターを指定して実行します。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** パラメーターを指定して実行します。すでに登録してある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** パラメーター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、**SYSCAT.TABLES** カタログ・ビューを照会します。例えば、

```
SELECT STATISTICS_PROFILE FROM SYSCAT.TABLES WHERE TABNAME = 'EMPLOYEE'
```

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フィーチャーで、統計プロファイルを自動的に生成することもできます。このフィーチャーを使用可能にすると、データベースのアクティビティー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。その後、統計プロファイルがこのデータに基づいて生成されます。このフィーチャーを使用可能にすると、特定のワークロードとどの統計が関連しているのかについての不確かさが軽減されます。

重要: 自動統計プロファイルは、バージョン 10.1 で非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「DB2 バージョン 10.1 の新機能」の『自動統計プロファイルが推奨されなくなった』を参照してください。

自動統計プロファイル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操作をスケジュールに入れます。

自動統計プロファイルを有効にするには、適切なデータベース構成パラメーターを設定して、表自動保守が既に使用可能になっていることを確認します。詳しくは『`auto_maint` - 自動保守構成パラメーター』を参照してください。

`auto_stats_prof` 構成パラメーターは照会フィードバック・データの収集を活性化し、`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性化し、自動統計収集で使用できるようにします。

パーティション・データベース環境の場合や、一部のフェデレーテッド・データベース環境、DB2 pureScale環境の場合、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、自動統計プロファイル生成がサポートされません。

`section_actuals` データベース構成パラメーターが使用可能になっている場合には、自動統計プロファイル生成を使用可能にすることはできません (SQLCODE -5153)。

自動統計プロファイルが最適なのは、多くの述部がある複雑で大規模な照会が実行されていたり、大きな結合を使用したり、大規模なグループ化が指定されていたりするシステムです。主にトランザクション・ワークロードから成るシステムには適していません。

ランタイム・モニターのパフォーマンス・オーバーヘッドが簡単に許容される開発環境では、`auto_stats_prof` と `auto_prof_upd` 構成パラメーターを ON に設定します。テスト・システムで現実のデータと照会を使用している場合、適切な統計プロファイルを実動システムに転送することが可能で、その場合は照会にさらにモニター・オーバーヘッドを取らずに済みます。

実稼働環境で、特定の照会セットにパフォーマンス上の問題 (原因が統計の不良である可能性がある問題) が検出される場合、`auto_stats_prof` 構成パラメーターを ON 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` 表に推奨を作成します。これらの推奨をよく読み、必要に応じて手動で統計プロファイルを改良できます。この推奨に基づいて DB2 サーバーが統計プロファイルを自動で更新するようにするには、`auto_stats_prof` を使用可能にするときに `auto_prof_upd` も使用可能にします。

照会フィードバック・ウェアハウスの作成

自動統計プロファイルに必要な照会フィードバック・ウェアハウスは、SYSTOOLSスキーマの5つの表で構成されています。これらの表には、照会実行時に検出される述部に関する情報と、統計収集の推奨事項が格納されます。5つの表は以下のとおりです。

- OPT_FEEDBACK_PREDICATE
- OPT_FEEDBACK_PREDICATE_COLUMN
- OPT_FEEDBACK_QUERY
- OPT_FEEDBACK_RANKING
- OPT_FEEDBACK_RANKING_COLUMN

SYSINSTALLOBJECTS プロシージャを使用して、照会フィードバック・ウェアハウスを作成します。SYSTOOLSスキーマでオブジェクトを作成またはドロップするのに使用されるこのプロシージャについて詳しくは、『SYSINSTALLOBJECTS』を参照してください。

自動統計収集とプロファイルによって使用されるストレージ:

自動統計収集および再編成のフィーチャーは、作業データをご使用のデータベースの一部である表に格納します。それらの表は、SYSTOOLSPACE 表スペースの中に作成されます。

SYSTOOLSPACE は、データベースがアクティブになった時点でデフォルト・オプションにより自動作成されます。それらの表のストレージ要件は、データベース中の表の数に比例し、1つの表に対して約1KBと見積もることができます。それがデータベースの重要なサイズである場合は、表スペースを自分でドロップしてから再作成し、ストレージを適切に割り振ることができます。表スペースの自動保守およびヘルス・モニター表は自動的に再作成されますが、こうした表でキャプチャーされた履歴は表スペースのドロップ時に失われます。

自動統計収集アクティビティのロギング:

統計ログは、特定のデータベースに対して生じたすべての統計収集アクティビティ（手動と自動の両方）に関する記録です。

統計ログのデフォルト名は、db2optstats.number.log です。これは \$diagpath/events ディレクトリーにあります。統計ログは回転ログです。ログの動作は、DB2_OPTSTATS_LOG レジストリー変数によって制御します。

統計ログは直接表示することもできますし、SYSPROC.PD_GET_DIAG_HIST 表関数を使用して照会することもできます。この表関数は、タイム・スタンプ、DB2 インスタンス名、データベース名、プロセス ID、プロセス名、およびスレッド ID などのログ・イベントに関する標準情報が含まれる複数の列を戻します。またログには、別のログ機能で使用するための汎用列も含まれます。以下の表で、統計ログにおけるこうした汎用列の使用法について説明します。

表 79. 統計ログ・ファイルでの汎用列

列名	データ・タイプ	説明
OBJTYPE	VARCHAR(64)	<p>イベントが適用されるオブジェクトのタイプ。統計ロギングの場合、これは収集される統計のタイプです。OBJTYPE は、プロセス開始時または停止時の統計収集バックグラウンド・プロセスを指すことがあります。さらに、自動統計収集によって実行されるアクティビティ (サンプリング・テスト、初期サンプリング、表評価など) のこともあります。</p> <p>統計収集アクティビティに可能な値は、以下のとおりです。</p> <p>TABLE STATS 表統計が収集されます。</p> <p>INDEX STATS 索引統計が収集されます。</p> <p>TABLE AND INDEX STATS 表統計と索引統計の両方が収集されます。</p> <p>自動統計収集に可能な値は、以下のとおりです。</p> <p>EVALUATION 自動統計バックグラウンド収集プロセスで、評価フェーズが開始されました。このフェーズでは、統計の更新を必要とするかどうかを判別するために表を検査し、必要な場合には統計が収集されます。</p> <p>INITIAL SAMPLING 表の統計を、サンプリングを使用して収集しています。このサンプル統計は、システム・カタログに格納されます。これにより、自動統計収集は、統計のない表に迅速に取り掛かることができます。その後の操作では、サンプリングなしに統計が収集されます。初期サンプリングは、自動統計収集の評価フェーズで実行されます。</p> <p>SAMPLING TEST 表の統計を、サンプリングを使用して収集しています。このサンプル統計は、システム・カタログに格納されません。このサンプル統計は現在のカタログ統計と比較されて、この表に対して完全統計を収集すべきかどうか、またいつ収集すべきかが判別されます。このサンプリングは、自動統計収集の評価フェーズで実行されます。</p> <p>STATS DAEMON 統計デーモンは、リアルタイム統計プロセスによってサブミットされた要求を処理するために使用されるバックグラウンド・プロセスです。このオブジェクト・タイプは、バックグラウンド・プロセスの開始および停止時にログに記録されます。</p>
OBJNAME	VARCHAR(255)	<p>該当する場合には、イベントが適用されるオブジェクトの名前。統計ロギングの場合、これは表または索引の名前です。OBJTYPE が STATS DAEMON または EVALUATION の場合、OBJNAME はデータベース名で、OBJNAME_QUALIFIER は NULL になります。</p>

表 79. 統計ログ・ファイルでの汎用列 (続き)

列名	データ・タイプ	説明
OBJNAME_QUALIFIER	VARCHAR(255)	統計ログGINGの場合、これは表または索引のスキーマです。
EVENTTYPE	VARCHAR(24)	<p>イベント・タイプは、このイベントに関連付けられているアクションです。統計ログGINGに可能な値は、以下のとおりです。</p> <p>COLLECT このアクションは、統計収集操作に関連してログに記録されます。</p> <p>START このアクションは、リアルタイム統計バックグラウンド・プロセス (OBJTYPE = STATS DAEMON) または自動統計収集の評価フェーズ (OBJTYPE = EVALUATION) が開始されるとログに記録されます。</p> <p>STOP このアクションは、リアルタイム統計バックグラウンド・プロセス (OBJTYPE = STATS DAEMON) または自動統計収集の評価フェーズ (OBJTYPE = EVALUATION) が停止されるとログに記録されます。</p> <p>ACCESS このアクションは、統計収集のために表に対するアクセスが試行された際にログに記録されます。このイベント・タイプは、オブジェクトが使用できないために失敗したアクセス試行をログに記録するために使用されます。</p> <p>WRITE このアクションは、統計キャッシュに保管されている収集済みの統計がシステム・カタログに書き込まれたときにログに記録されます。</p>
FIRST_EVENTQUALIFIERTYPE	VARCHAR(64)	最初のイベント修飾子のタイプです。イベント修飾子は、イベントによって影響を受けた対象について記述するために使用します。統計ログGINGの場合、最初のイベント修飾子はイベントが発生した際のタイム・スタンプです。最初のイベント修飾子タイプの値は、AT です。
FIRST_EVENTQUALIFIER	CLOB(16k)	イベントの最初の修飾子です。統計ログGINGの場合、最初のイベント修飾子は統計イベントが発生した際のタイム・スタンプです。統計イベントのタイム・スタンプは、TIMESTAMP 列によって表されるログ・レコードのタイム・スタンプとは異なる場合があります。
SECOND_EVENTQUALIFIERTYPE	VARCHAR(64)	2 番目のイベント修飾子のタイプです。統計ログGINGの場合、可能な値は BY または NULL です。その他のイベント・タイプではこのフィールドは使用されません。

表 79. 統計ログ・ファイルでの汎用列 (続き)

列名	データ・タイプ	説明
SECOND_EVENTQUALIFIER	CLOB(16k)	<p>イベントの 2 番目の修飾子です。統計ロギングの場合、COLLECT イベント・タイプで統計が収集された方法を示します。可能な値は以下のとおりです。</p> <p>User DB2 ユーザーが、LOAD、REDISTRIBUTE、または RUNSTATS コマンドを起動して、あるいは CREATE INDEX ステートメントを発行して、統計収集が実行されました。</p> <p>Synchronous 統計収集は、DB2 によって SQL ステートメント・コンパイル時に実行されました。統計は、システム・カタログ内ではなく統計キャッシュに格納されます。</p> <p>Synchronous sampled 統計収集は、DB2 によって SQL ステートメント・コンパイル時にサンプリングを使用して実行されました。統計は、システム・カタログ内ではなく統計キャッシュに格納されます。</p> <p>Fabricate 統計は、SQL ステートメント・コンパイル時に、データおよび索引マネージャーが維持している情報を使用して作成されました。統計は、システム・カタログ内ではなく統計キャッシュに格納されます。</p> <p>Fabricate partial 一部の統計に限り、SQL ステートメント・コンパイル時に、データ・マネージャーおよび索引マネージャーが保守している情報を使用して作成されました。特に、特定の列の HIGH2KEY 値と LOW2KEY 値だけは作成されたものです。統計は、システム・カタログ内ではなく統計キャッシュに格納されます。</p> <p>Asynchronous 統計は DB2 バックグラウンド・プロセスによって収集され、システム・カタログに格納されました。</p> <p>その他のイベント・タイプではこのフィールドは使用されません。</p>
THIRD_EVENTQUALIFIERTYPE	VARCHAR(64)	<p>3 番目のイベント修飾子のタイプです。統計ロギングの場合、可能な値は DUE TO または NULL です。</p>

表 79. 統計ログ・ファイルでの汎用列 (続き)

列名	データ・タイプ	説明
THIRD_EVENTQUALIFIER	CLOB(16k)	<p>イベントの 3 番目の修飾子です。統計ロギングの場合、統計アクティビティを完了できなかった理由を示します。可能な値は以下のとおりです。</p> <p>Timeout 同期統計収集の予定時間を超過しました。</p> <p>Error エラーが原因で統計アクティビティが失敗しました。</p> <p>RUNSTATS error RUNSTATS エラーが原因で同期統計収集が失敗しました。一部のエラーでは、統計は収集できないものの SQL ステートメント・コンパイルは正常に完了するという場合があります。例えば、統計を収集するにはメモリーが不十分な場合、SQL ステートメント・コンパイルは続行されます。</p> <p>Object unavailable データベース・オブジェクトにアクセスできなかったため、その統計を収集できませんでした。考えられる理由には、以下のものが含まれます。</p> <ul style="list-style-type: none"> • オブジェクトが超排他 (Z) モードでロックされている • オブジェクトが存在する表スペースを使用できない • 表索引を再作成する必要がある <p>Conflict 別のアプリケーションが既に同期統計を収集していたため、同期統計収集が実行されませんでした。</p> <p>エラーの詳細については、FULLREC 列または db2diag ログ・ファイルを確認してください。</p>
EVENTSTATE	VARCHAR(255)	<p>イベントの結果のオブジェクトまたはアクションの状態。統計ロギングの場合、統計操作の状態を示します。可能な値は以下のとおりです。</p> <ul style="list-style-type: none"> • Start • Success • Failure

例

以下の例に挙げる照会では、PD_GET_DIAG_HIST を呼び出し、最大で、現在のタイム・スタンプからさかのぼって過去 1 年分のイベントの統計ログ・レコードを返します。

```
select pid, tid,
       substr(eventtype, 1, 10),
       substr(objtype, 1, 30) as objtype,
       substr(objname_qualifier, 1, 20) as objschema,
       substr(objname, 1, 10) as objname,
       substr(first_eventqualifier, 1, 26) as event1,
       substr(second_eventqualifiertype, 1, 2) as event2_type,
       substr(second_eventqualifier, 1, 20) as event2,
       substr(third_eventqualifiertype, 1, 6) as event3_type,
       substr(third_eventqualifier, 1, 15) as event3,
       substr(eventstate, 1, 20) as eventstate
from table(sysproc.pd_get_diag_hist
```

```

('optstats', 'EX', 'NONE',
 current_timestamp - 1 year, cast(null as timestamp))) as sl
order by timestamp(varchar(substr(first_eventqualifier, 1, 26), 26));

```

結果は、FIRST_EVENTQUALIFIER 列に格納されているタイム・スタンプ順に並べられます。この列は、統計イベントの時間を表します。

PID	TID	EVENTTYPE	OBJTYPE	OBJSHEMA	OBJNAME	EVENT1	EVENT2_	EVENT2	EVENT3_	EVENT3	EVENTSTATE
							TYPE		TYPE		
28399	1082145120	START	STATS DAEMON	-	PROD_DB	2007-07-09-18.37.40.398905	-	-	-	-	success
28389	183182027104	COLLECT	TABLE AND INDEX STATS	DB2USER	DISTRICT	2007-07-09-18.37.43.261222	BY	Synchronous	-	-	start
28389	183182027104	COLLECT	TABLE AND INDEX STATS	DB2USER	DISTRICT	2007-07-09-18.37.43.407447	BY	Synchronous	-	-	success
28399	1082145120	COLLECT	TABLE AND INDEX STATS	DB2USER	CUSTOMER	2007-07-09-18.37.43.471614	BY	Asynchronous	-	-	start
28399	1082145120	COLLECT	TABLE AND INDEX STATS	DB2USER	CUSTOMER	2007-07-09-18.37.43.524496	BY	Asynchronous	-	-	success
28399	1082145120	STOP	STATS DAEMON	-	PROD_DB	2007-07-09-18.37.43.526212	-	-	-	-	success
28389	183278496096	COLLECT	TABLE STATS	DB2USER	ORDER_LINE	2007-07-09-18.37.48.676524	BY	Synchronous sampled	-	-	start
28389	183278496096	COLLECT	TABLE STATS	DB2USER	ORDER_LINE	2007-07-09-18.37.53.677546	BY	Synchronous sampled	DUE TO	Timeout	failure
28389	1772561034	START	EVALUATION	-	PROD_DB	2007-07-10-12.36.11.092739	-	-	-	-	success
28389	8231991291	COLLECT	TABLE AND INDEX STATS	DB2USER	DISTRICT	2007-07-10-12.36.30.737603	BY	Asynchronous	-	-	start
28389	8231991291	COLLECT	TABLE AND INDEX STATS	DB2USER	DISTRICT	2007-07-10-12.36.34.029756	BY	Asynchronous	-	-	success
28389	1772561034	STOP	EVALUATION	-	PROD_DB	2007-07-10-12.36.39.685188	-	-	-	-	success
28399	1504428165	START	STATS DAEMON	-	PROD_DB	2007-07-10-12.37.43.319291	-	-	-	-	success
28399	1504428165	COLLECT	TABLE AND INDEX STATS	DB2USER	CUSTOMER	2007-07-10-12.37.43.471614	BY	Asynchronous	-	-	start
28399	1504428165	COLLECT	TABLE AND INDEX STATS	DB2USER	CUSTOMER	2007-07-10-12.37.44.524496	BY	Asynchronous	-	-	failure
28399	1504428165	STOP	STATS DAEMON	-	PROD_DB	2007-07-10-12.37.45.905975	-	-	-	-	success
28399	4769515044	START	STATS DAEMON	-	PROD_DB	2007-07-10-12.48.33.319291	-	-	-	-	success
28389	4769515044	WRITE	TABLE AND INDEX STATS	DB2USER	CUSTOMER	2007-07-10-12.48.33.969888	BY	Asynchronous	-	-	start
28389	4769515044	WRITE	TABLE AND INDEX STATS	DB2USER	CUSTOMER	2007-07-10-12.48.34.215230	BY	Asynchronous	-	-	success

大きい統計ログの照会パフォーマンスの向上:

統計ログ・ファイルが大きい場合、ログ・レコードを表にコピーし、索引を作成し、それから統計を収集することにより、照会のパフォーマンスを向上させることができます。

手順

1. ログ・レコードのために適切な列を含む表を作成します。

```

create table db2user.stats_log (
  pid          bigint,
  tid          bigint,
  timestamp    timestamp,
  dbname       varchar(128),
  retcode      integer,
  eventtype    varchar(24),
  objtype      varchar(30),
  objschema    varchar(20),
  objname      varchar(30),
  event1_type  varchar(20),
  event1       timestamp,
  event2_type  varchar(20),
  event2       varchar(40),
  event3_type  varchar(20),
  event3       varchar(40),
  eventstate   varchar(20))

```

2. 照会のカーソルを SYSPROC.PD_GET_DIAG_HIST に対して宣言します。

```

declare c1 cursor for
select pid, tid, timestamp, dbname, retcode, eventtype,
  substr(objtype, 1, 30) as objtype,
  substr(objname_qualifier, 1, 20) as objschema,
  substr(objname, 1, 30) as objname,
  substr(first_eventqualifiertype, 1, 20),
  substr(first_eventqualifier, 1, 26),
  substr(second_eventqualifiertype, 1, 20),
  substr(second_eventqualifier, 1, 40),
  substr(third_eventqualifiertype, 1, 20),
  substr(third_eventqualifier, 1, 40),
  substr(eventstate, 1, 20)

```

```

from table (sysproc.pd_get_diag_hist
('optstats', 'EX', 'NONE',
current_timestamp - 1 year, cast(null as timestamp ))) as s1

```

3. この表に統計ログ・レコードをロードします。

```
load from c1 of cursor replace into db2user.stats_log
```

4. 索引を作成し、表の統計を収集します。

```

create index s1_ix1 on db2user.stats_log(eventtype, event1);
create index s1_ix2 on db2user.stats_log(objtype, event1);
create index s1_ix3 on db2user.stats_log(objname);

```

```

runstats on table db2user.stats_log
with distribution and sampled detailed indexes all;

```

統計の収集および更新のガイドライン

RUNSTATS ユーティリティは、表、索引、および統計ビューに関する統計を収集し、アクセス・プランの選択のための正確な情報をオプティマイザーに提供します。

以下のような状態のときに、**RUNSTATS** ユーティリティ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、適切な索引が作成された後
- 表に索引を作成した後
- 表が **REORG** ユーティリティによって再編成された後
- 更新、挿入、または削除操作によって、表とその索引が大幅に変更された後
- 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
- 現在の統計と前の統計を比較する場合
- プリフェッチ値が変更された場合
- **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを実行した後
- XML 列がある場合。**RUNSTATS** が XML 列の統計を収集するためだけに使用される場合、ロード操作または前の **RUNSTATS** 操作の際に収集された非 XML 列の既存の統計は保持されます。いくつかの XML 列の統計が以前に収集されている場合、それらの統計は置換されます。現行の **RUNSTATS** 操作でそれらの列が含まれない場合にはドロップされます。

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。

RUNSTATS を実行した後は、アプリケーション・プログラムを再バインドする必要があります。新しい統計を利用できる場合には、異なるアクセス・プランが照会オプティマイザーによって選択されることがあります。

一度に一連のすべての統計を収集することができない場合には、オブジェクトのサブセットに対して **RUNSTATS** ユーティリティを使用します。これらのオブジェクトに対して継続的に行われるアクティビティの結果として不整合が発生した場合は、照会の最適化中に警告メッセージ (SQL0437W、理由コード 6) が返されます。このような場合、**RUNSTATS** を再度使用し、分散統計を更新してください。

索引統計を対応する表と確実に同期化させるために、表と索引の両方の統計を同時に収集してください。統計を最後に収集した時点以降に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更新すると、それら 2 つの統計セットの間の同期が失われます。

実動システムで **RUNSTATS** ユーティリティーを使用すると、ワークロード・パフォーマンスに悪影響を与える可能性があります。この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。

パーティション・データベース環境で表の統計を収集する際に、**RUNSTATS** は、このユーティリティーの実行元のデータベース・パーティション上でのみ作動します。このデータベース・パーティションの結果は、他のデータベース・パーティションに外挿されます。このデータベース・パーティションにこの表の必要な部分が含まれていない場合、その要求は、そのデータベース・パーティション・グループの中の、必要なデータが含まれる最初のデータベース・パーティションに送られます。

統計ビューの統計は、そのビューで参照される基本表が入っている全データベース・パーティションについて収集されます。

RUNSTATS の効率と統計の有用性を向上させるため、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**、あるいは照会の類似の節で参照される列に関する統計のみを収集します。これらの列に索引が作成されている場合には、これらの列を **RUNSTATS** コマンドの **ONLY ON KEY COLUMNS** 節で指定できます。
- データベース構成パラメーター **num_freqvalues** および **num_quantiles** の値を、特定の表および列に合わせてカスタマイズします。
- 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** 節を使用します。
- 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** を再使用して統計を更新してください。
- **RUNSTATS** は単一データベース・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データベース・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。データ分散が偏っていると疑われる場合は、**RUNSTATS** ユーティリティーを使用する前に **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを使用して、全データベース・パーティションにデータを再分散させることを考慮してください。
- DB2 V9.7 フィックスパック 1 以降のリリースでは、分散統計が XML 列で収集されます。分散統計は、XML 列で指定された XML データの索引ごとに収集されます。デフォルトでは、XML データの索引ごとに最大 250 変位値が分散統計に関して使用されます。

XML 列で分散統計を収集する場合、変位値の最大数を変更できます。変位値の最大数を少なめにすると、特定のデータ・サイズに基づいて XML 分散統計のスペース所要量を減らすことができますし、XML データの索引に関するデータ・

セットの分散統計をキャプチャーするために 250 変位値では不十分な場合にはこの変位値の最大数を増やすこともできます。

カタログ統計の収集:

RUNSTATS ユーティリティーを使用すると、表、索引、および統計ビューに関するカタログ統計を収集できます。照会オプティマイザーは、この情報を使用して、最適な照会のアクセス・プランを選択します。

このタスクについて

このユーティリティーを使用するのに必要な特権および権限については、**RUNSTATS** コマンドの説明を参照してください。

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、索引、または統計ビューを含んでいるデータベースに接続します。
2. 以下のいずれかの方法を使用して、表、索引、または統計ビューに対して実行される照会に関する統計を収集します。
 - 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、索引、または統計ビューに対して実行される照会に関して収集される統計を調整できます。
 - IBM Data Studio から、**RUNSTATS** コマンドのタスク・アシスタントを開きます。
3. `runstats` オペレーションが完了したら、**COMMIT** ステートメントを発行してロック解除します。
4. 統計情報を更新した表、索引、または統計ビューにアクセスするパッケージのすべてを再バインドします。

タスクの結果

注:

1. **RUNSTATS** コマンドでは、ニックネームの使用がサポートされません。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、**RUNSTATS** を使用してすべてのデータベース内にある表の統計を更新し、次いでリモート表にアクセスするニックネームをドロップおよび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。
2. パーティション・データベース環境で表の統計を収集する際に、**RUNSTATS** は、このユーティリティーの実行元のデータベース・パーティション上でのみ作動します。このデータベース・パーティションの結果は、他のデータベース・パーティションに外挿されます。このデータベース・パーティションにこの表の必要な部分が含まれていない場合、その要求は、そのデータベース・パーティション・グループの中の、必要なデータが含まれる最初のデータベース・パーティションに送られます。

統計ビューの統計は、そのビューで参照される基本表が入っている全データベース・パーティションについて収集されます。

3. DB2 V9.7 フィックスパック 1 以降のリリースでは、以下の事柄が XML タイプの列における分散統計の収集に適用されます。
- 分散統計は、XML 列で指定された XML データの索引ごとに収集されます。
 - **RUNSTATS** コマンドが XML データの索引で分散統計を収集するには、分散統計と表統計の両方を収集しなければなりません。
 - デフォルトでは、**RUNSTATS** コマンドによって、XML データの索引ごとに分散統計の最大で 250 の変位値が収集されます。列の変位値の最大数は、**RUNSTATS** コマンドの実行時に指定できます。
 - 分散統計は、タイプ VARCHAR、DOUBLE、TIMESTAMP、および DATE の XML データの索引で収集されます。タイプ VARCHAR HASHED の XML データの索引では、XML 分散統計は収集されません。
 - パーティション表で定義された XML データのパーティション索引の場合、分散統計は収集されません。

関連情報:

データのサンプルでの統計の収集:

表統計は、照会に最適なアクセス・プランを選択するときに照会オプティマイザーによって使用されるので、統計が最新のものであることは重要です。データベースのサイズが増加し続けると、効果的な統計収集を行うのはますます難しくなります。

表および索引データの無作為抽出で統計を収集するのが有効な方法です。入出力制約やプロセッサ制約があるシステムの場合、こうしたパフォーマンス上の利点は計り知れません。

DB2 製品を使用すると、統計収集データのサンプリングを効率的に行えます。これによって、高度の正確さを維持しながら、**RUNSTATS** ユーティリティのパフォーマンスを何十倍も改善できる可能性があります。

2 つのサンプリング方式を使用できます。行レベルのサンプリングと、ページ・レベルのサンプリングです。こうしたサンプリング方式の説明については、『照会でのデータ・サンプリング』を参照してください。

サンプルに組み込まれるページごとに 1 つの入出力操作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では、各表ページを完全な表または索引の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、統計収集はプロセッサ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・レベルの表サンプリングよりも行レベルの表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・レベルの表サンプリングと比べると、行レベルの表サンプル・セットでは各データ・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良く反映することができます。ページ・レベルの表サンプリングでは、 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 (前の表サンプルが再生成される) **REPEATABLE** パラメーターを指定せずに何回も呼び出すと、それぞれの表サンプルがランダムに生成されます。このオプションが有用なのは、データが一定のままの表で一貫性のある統計が必要な場合です。

REPEATABLE は索引サンプリング (**INDEXSAMPLE**) には適用されず、同様の機能もありません。

IBM Data Studio バージョン 3.1 以降では、次のタスクのためにタスク・アシスタントを使用できます: 統計の収集、タスク・アシスタントは、オプションの設定、タスク実行のために自動生成されたコマンドの確認、およびそれらのコマンドの実行のプロセスをガイドします。詳しくは、タスク・アシストを使用したデータベースの管理を参照してください。

サブエレメント統計:

パターンの末尾を除く任意の箇所で % ワイルドカード文字を使用して **LIKE** 述部を指定すると、サブエレメント構造に関する基本情報を収集できるでしょう。

ワイルドカード **LIKE** 述部 (例: `SELECT...FROM DOCUMENTS WHERE KEYWORDS LIKE '%simulation%'`) に加え、列および照会も、サブエレメント統計を役立たせるために特定の基準を満たしていなければなりません。

表列には、ブランクで区切られたサブフィールドまたはサブエレメントが含まれている必要があります。例えば、4 行の表 **DOCUMENTS** に **KEYWORDS** 列があり、この列には、テキスト検索の目的で使用する関連キーワードのリストが含まれているとします。 **KEYWORDS** 内の値は以下のとおりです。

```
'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ています。各エレメントは、ブランクで区切られたワード (キーワード) です。

照会では、これらの列を **WHERE** 文節において参照する必要があります。

オプティマイザーは、各述部と一致する行数を常に見積もります。これらのワイルドカード **LIKE** 述部の場合、オプティマイザーは、一致する列に、連結された一連のエレメントが含まれていると推定し、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を収集するなら、オプティマイザーは各サブエレメントの長さおよび区切り文字についての情報を取得します。この追加情報を使用して、述部と一致する行数をより正確に見積もることができます。

サブエレメント統計を収集するには、**LIKE STATISTICS** パラメーターを使用して、**RUNSTATS** コマンドを実行します。

サブエレメントについての **RUNSTATS** 統計:

RUNSTATS ユーティリティで **LIKE STATISTICS** 節を指定した場合、1 バイト文字セット (SBCS)、FOR BIT DATA、または UTF-8 のコード・ページ属性を持つタイプ CHAR および VARCHAR の列に関して、統計が収集されます。

SUB_COUNT

サブエレメントの平均数。

SUB_DELIM_LENGTH

サブエレメントを分ける各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。

サブエレメント統計の値を調べるには、SYSCAT.COLUMNS カタログ・ビューを照会します。例えば、

```
select substr(colname, 1, 16), sub_count, sub_delim_length
from syscat.columns where tabname = 'DOCUMENTS'
```

LIKE STATISTICS 節を使用する場合、**RUNSTATS** ユーティリティの完了には時間がかかることがあります。このオプションの使用を考慮している場合、このオーバーヘッドが追加されても照会パフォーマンスが向上するかを評価してください。

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用の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。例えば、SYSSTAT.COLUMNS 内の COLCARD は SYSSTAT.TABLES 内の CARD より小さくしなければなりません (列内の固有値の数は、表の行の数より大き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 に減らすと仮定します。このとき最初に SYSSTAT.TABLES を更新したとすると、エラーが返されます (CARD が COLCARD より小さくなってしまったため)。

状況によっては、矛盾を検出するのが難しく、エラーが返されない場合もあります。影響する統計が別々のカタログ表に保管されている場合には特にそう言えます。

カタログ統計を更新する前に、(少なくとも) 次の点を確認してください。

- 数値統計が -1 であるか、0 以上である。
- パーセンテージを表す数値統計 (例えば SYSSTAT.INDEXES 内の CLUSTERRATIO) が 0 から 100 までの範囲である。

表が作成される時、カタログ統計は表に統計がないことを示す -1 に設定されます。統計が収集されるまで、DB2 サーバーでは SQL または XQuery ステートメントのコンパイルおよび最適化にデフォルト値を使用します。新しい値がデフォルト値と不整合の場合、表または索引の統計の更新が失敗することがあります。このため、表の作成後、かつ表またはその索引の統計の更新を試みる前に、runstats ユーティリティを使用することをお勧めします。

注:

1. 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。
2. パーティション・レベルの表および索引の統計は、更新されません。

手動での列統計の更新の規則:

SYSSTAT.COLUMNS カタログ・ビュー内の統計を更新するには、以下の指針に従ってください。

- HIGH2KEY または LOW2KEY の値を手動で更新する場合、次の点を確認してください。
 - これらが、対応するユーザー列のデータ・タイプの有効値であること。
 - これらの値の長さは、33 またはターゲット列のデータ・タイプの最大長の、いずれか小さい方でなければならない (追加された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 または 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。
 - これらの値が、コスト計算のためだけでなく、UPDATE ステートメントの SET 節でも使用できるような方法で格納されること。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ことを意味します。HIGH2KEY および LOW2KEY のユーザー列値および対応する値の例を表 80 に示します。

表 80. HIGH2KEY および LOW2KEY のデータ・タイプ別の値

ユーザー列のデータ・タイプ	ユーザー・データ	対応する HIGH2KEY または 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 よりも大きい値になります。
- 列のカーディナリティー (SYSSTAT.COLUMNS 内の COLCARD) は、対応する表または統計ビューのカーディナリティー (SYSSTAT.TABLES 内の CARD) より大きくすることはできません。
- 列の NULL 値の数 (SYSSTAT.COLUMNS 内の NUMNULLS) は、対応する表または統計ビューのカーディナリティー (SYSSTAT.TABLES 内の CARD) より大きくすることはできません。
- LONG またはラージ・オブジェクト (LOB) データ・タイプで定義された列の統計はサポートされません。

手動での表およびニックネーム統計の更新の規則:

SYSSTAT.TABLES カタログ・ビュー内の統計を更新するには、以下の指針に従ってください。

- SYSSTAT.TABLES で更新できる統計値は、CARD、FPAGES、NPAGES、AVGCOMPRESSEDROWSIZE、

AVGROWCOMPRESSIONRATIO、PCTROWSCOMPRESSED、OVERFLOW、およびマルチディメンション・クラスタリング (MDC) 表の場合の ACTIVE_BLOCKS だけです。

- CARD 統計の値は以下の基準を満たしている必要があります。
 - SYSSTAT.COLUMNS 内の対応する表のすべての COLCARD 統計値以上でなければならない。
 - NPAGES 統計値よりも大きくなければならない。
 - いかなる索引の PAGE_FETCH_PAIRS 列内のいかなる「フェッチ」値に対しても、その値以下であってはならない。(CARD 統計が索引に関連していることを想定しています)。
- FPAGES 統計の値は、NPAGES 統計の値よりも大きくなければなりません。
- NPAGES 統計の値は、いかなる索引の PAGE_FETCH_PAIRS 列内のいかなる「フェッチ」値に対しても、その値以下でなければなりません。
- AVGCOMPRESSEDROWSIZE 統計の有効値は -1 か、または 0 から AVGROWSIZE 統計値の範囲です。
- AVGROWCOMPRESSIONRATIO 統計の有効値は -1 か、または 1 より大きい値です。
- PCTROWSCOMPRESSED 統計の有効値は -1、または 0 以上 100 以下です。

フェデレーテッド・データベース・システムでは、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動で更新するときには、注意が必要です。ニックネームが返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誤った判断をすることがあります。ただし、場合によっては統計の更新がリモート・ビューで役立つこともあります。例えば、SELECT リストに列関数が適用されていない単一の基本表に対してリモート・ビューを定義する場合などです。複合ビューでは、照会ごとに調整が必要となる複雑なチューニング・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それらの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。

詳細索引統計

DETAILED パラメーターを指定して索引に **RUNSTATS** 操作を実行したときに収集される統計情報を使用すると、オプティマイザーは、バッファー・プール・サイズに基づいて必要なデータ・ページ・フェッチの数を見積もることができます。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。

詳細な統計は、完全な索引スキャンがさまざまなバッファー・プール・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** ユーティリティー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を見積もります。例えば、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。最悪の状況として、各行が別のページを参照する場合、入出力の数が、多い場合でも索引付けされた表内の行数と同じになります。他方の極端な例を挙げると、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、すべての表ページが一度に読み取られます。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。クラスタリングが少なければ少ないほど、索引を介して表の行にアクセスするために必要な入出力が多くなります。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。

次のような場合に、詳細な索引統計を収集します。

- 索引に含まれていない列を照会で参照する場合
- 表にクラスタリングの程度が異なる複数の非クラスター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、これらの条件を識別するのは困難です。これらの状態が存在しているかどうかを最もコストをかけずに判別する方法は、索引に関する詳細な統計を収集し、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持するという方法でしょう。

詳細な索引統計を収集する場合、`RUNSTATS` 操作にはより多くのメモリーを必要とし、完了までの処理時間も長くなります。例えば `DETAILED` オプション (`SAMPLED DETAILED` パラメーターと同等) では、2 MB の統計ヒープが必要です。この追加メモリー要件用に、488 (4 KB) 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` ユーティリティーは統計の収集を試行する前にエラーを戻します。

表のサイズが相当大きい (おおよそ 25 ページより大きい) ものでない限り、`CLUSTERFACTOR` および `PAGE_FETCH_PAIRS` は収集されません。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) です。表が比較的小さいと、`RUNSTATS` ユーティリティーで `CLUSTERRATIO` (0 から 100 までの値) だけが収集され、`CLUSTERFACTOR` および `PAGE_FETCH_PAIRS` は収集されません。`DETAILED` 節を指定しないと、`CLUSTERRATIO` だけが収集されます。

索引統計の収集:

索引統計を収集すると、照会の解決に索引を使用すべきかどうかを、オプティマイザーが判断するのに役立ちます。

このタスクについて

以下の例は、`SALES` という名前のデータベースに基づいており、この中には索引 `CUSTIDX1` および `CUSTIDX2` を備えた `CUSTOMERS` 表が含まれています。

`RUNSTATS` ユーティリティーを使用するのに必要な特権および権限については、`RUNSTATS` コマンドの説明を参照してください。

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 コマンド行から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収集するには、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収集するものの、各索引項目ごとの詳細な計算ではなくサンプリングによるものにするには、以下を行います。

```
runstats on table sales.customers and sampled detailed indexes all
```

SAMPLED DETAILED パラメーターでは、2 MB の統計ヒープが必要です。この追加メモリー要件用に、488 (4 KB) 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** ユーティリティーは統計の収集を試行する前にエラーを戻します。

- 索引および表の統計に整合性を持たせるために、表の分散統計に加えてサンプリング済み索引に関する詳細な統計を収集するには、以下を行います。

```
runstats on table sales.customers  
with distribution on key columns  
and sampled detailed indexes all
```

手動での索引統計の更新の規則:

SYSSTAT.INDEXES カタログ・ビュー内の統計を更新する際には、以下の指針に従ってください。

- 以下の規則は、PAGE_FETCH_PAIRS に適用されます。
 - 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 (2 147 483 647) より小さい値でなければなりません。
 - PAGE_FETCH_PAIRS 統計内の個々の値は、ブランク区切り文字で区切る必要があります。
 - CLUSTERFACTOR が 0 より大きい場合は、必ず有効な PAGE_FETCH_PAIRS 統計が存在していなければなりません。
 - 単一の PAGE_FETCH_PAIRS 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 統計のバッファー・サイズ値 (各ペアの最初の値) は、昇順に並んでいなければなりません。
 - PAGE_FETCH_PAIRS 統計のバッファー・サイズ値は、32 ビット・オペレーティング・システムの場合は MIN(NPAGES, 524 287)、64 ビット・オペレーティング・システムの場合は MIN(NPAGES, 2 147 483 647) 以下にする必要があります。ここで NPAGES (SYSSTAT.TABLES に保管されている) は、対応する表のページ数です。
 - PAGE_FETCH_PAIRS 統計のページ・フェッチ値 (各ペアの 2 番目値) は、降順に並んでいなければなりません。この場合、個々の値は、対応する表の NPAGES より小さくしたり CARD より大きくしたりすることはできません。

- 2つの連続したペアにおいてバッファー・サイズ値が同一である場合は、両方のペアのページ・フェッチ値も同一でなければなりません。

有効な PAGE_FETCH_PAIRS 統計の例を示します。

```
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
```

- 以下の規則は、CLUSTERRATIO および CLUSTERFACTOR に適用されます。
 - 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_PAIRS 値が伴わなければなりません。
- リレーショナル索引の場合、以下の規則は、FIRSTKEYCARD、FIRST2KEYCARD、FIRST3KEYCARD、FIRST4KEYCARD、FULLKEYCARD、および INDCARD に適用されます。
 - 単一列索引の場合、FIRSTKEYCARD は FULLKEYCARD と等しくなければなりません。
 - FIRSTKEYCARD は、対応する列の SYSSTAT.COLUMNS 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 に設定します。例えば、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、それらの間の関係は以下のようにならなければなりません。

```
FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
<= FULLKEYCARD <= INDCARD == CARD
```

- XML データの索引の場合、FIRSTKEYCARD、FIRST2KEYCARD、FIRST3KEYCARD、FIRST4KEYCARD、FULLKEYCARD、および INDCARD の間の関係は以下のようになる必要があります。

```
FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
<= FULLKEYCARD <= INDCARD
```

- 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。

分散統計

頻出値統計および変位値統計という 2 種類のデータ分散統計を収集できます。

- **頻出値統計** は、重複の数が最も多い列およびデータ値、その次に重複の数が多いうように、`num_freqvalues` データベース構成パラメーターの値で指定したレベルまで、それらの情報を提供します。頻出値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。 `RUNSTATS` コマンドで、特定の表、統計ビュー、または列に対して `NUM_FREQVALUES` 節を使用することもできます。
- **変位値統計** は、データ値が他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、値を昇順に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」(quantiles) 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 から 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` 述部で最大プラス・マイナス 5% のエラーです。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` は、特定の表、統計ビュー、または列に対して設定できます。

注: `num_freqvalues` および `num_quantiles` の値を大きくすると、`RUNSTATS` ユーティリティーで消費される処理リソースとメモリー (`stat_heap_sz` データベース構成パラメーターで指定される) が増加します。

分散統計を収集すべきとき

表または統計ビューに関する分散統計が役立つかどうかを判断するには、まず以下の点を考慮してください。

- アプリケーションの照会でホスト変数が使用されるか。

分散統計は、ホスト変数を使用しない動的照会および静的照会の場合に最も役立ちます。オプティマイザーは、ホスト変数を含む照会を評価するときには、分散統計を限定的に使用します。

- 列内のデータの分布が均一か。

表のうちの少なくとも 1 つの列に、かなり「不均一」なデータ分散があり、その列が下記の節のような等価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、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、高度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、以下の列を考慮してください。

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ために、照会が役立つ場合があります。例えば、

```
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。

表 81. 列におけるデータ値の頻度

データ値	頻度
20	5
30	10
40	10
50	25
60	25
70	20
80	5

頻出値と変位値の両方の統計は、オプティマイザーが多数の重複値を処理するために役立ちます。

索引統計のみを収集すべきとき

次の状況では、索引データにのみ基づく統計の収集を考慮できます。

- **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。

指定する統計精度のレベル

分散統計を保管する精度を指定するには、データベース構成パラメーター **num_quantiles** および **num_freqvalues** を使用します。精度は、表または列の統計を収集するときに、**RUNSTATS** コマンドの対応するオプションで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** ユーティリティーが使用する精度が高くなります。ただし、**RUNSTATS** 操作自体の間と、カタログ表により多くのデータを格納するための両方において、精度を高くするとより多くのリソースが必要になります。

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 から 100 までの値を指定します。頻出値統計は、頻出値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に作成するのが理想です。データベース・マネージャーが収集する回数はこの数より少ない場合があります。その理由は、複数回数出現す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。

変位値の数を指定するには、**num_quantiles** データベース構成パラメーターを、20 から 50 までの値に設定します。

- 最初に、範囲照会の行数を見積もるときの最大許容エラー P をパーセント単位で求めます。
- 変位値の数は、BETWEEN 述部の場合は約 $100/P$ にし、その他の種類の範囲述部 (<, <=, >, または >=) の場合は約 $50/P$ にします。

例えば、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% という結果になるはずですが。一般に、変位値は 10 以上を指定します。50 を超える変位値が必要になるのは、極端に不均一なデータの場合のみです。頻出値統計のみを必要とする場合、**num_quantiles** を 0 に設定してください。このパラメーターを 1 に設定すると、値の範囲全体が 1 つの変位値内に収まるため、変位値統計は収集されません。

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、さまざまな照会アクセス・プランのコストをより正確に見積もることができます。

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。

次の例を考えてみましょう。コストが最低のアクセス・プランを選択するために、等価述部または範囲述部を満たす列値を持つ行の数をオプティマイザーで見積もる必要があります。見積もりが正確になるほど、オプティマイザーが最適なアクセス・プランを選択する可能性が大きくなります。次の照会を考えてみましょう。

```
select c1, c2
  from table1
  where c1 = 'NEW YORK'
  and c2 <= 10
```

C1 と C2 の両方の列に索引が 1 つあるとします。この場合に可能なアクセス・プランの 1 つとして、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 であるかどうかを調べるという方法があります。別のプランとして、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' であるかどうかを調べる、という方法があります。通常、照会を実行するための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ことは、各述部を満たす行の数を見積もることを意味します。

使用できる分散統計はなくても、runstats ユーティリティーが表または統計ビューに対して使用されている場合、オプティマイザー使用できる情報は、ある列における、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。等価述部または範囲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 LOW2KEY と HIGH2KEY の間で均等に分布する、という仮定の下に見積もられます。特に、等価述部 (C1 = KEY) を満たす行の数は CARD/COLCARD として見積もられ、範囲述部 (C1 BETWEEN KEY1 AND KEY2) を満たす行の数は次の公式を使用して見積もられます。

$$\frac{\text{KEY2} - \text{KEY1}}{\text{HIGH2KEY} - \text{LOW2KEY}} \times \text{CARD}$$

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、データ値の頻度が大きく異なっているか、またはデータ値の分布が非常に不均一である場合には、見積もりは桁違いのものになる場合があります、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見積もるために頻出値統計を使用し、範囲述部を満たす行数を見積もるために頻出値統計と変位値統計の両方を使用することによって、そのようなエラーの可能性を大幅に減らすことができます。

特定の列に関する分散統計の収集:

RUNSTATS 操作および続く照会プラン分析を効率的に行うには、照会の WHERE、GROUP BY、および類似の節で参照される列に関してのみ、分散統計を収集します。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。

このタスクについて

以下の例は、SALES という名前のデータベースに基づいており、この中には索引 CUSTIDX1 および CUSTIDX2 を備えた CUSTOMERS 表が含まれています。

RUNSTATS ユーティリティーを使用するのに必要な特権および権限については、**RUNSTATS** コマンドの説明を参照してください。

パーティション・データベース環境で表の統計を収集する際に、**RUNSTATS** は、このユーティリティーの実行元のデータベース・パーティション上でのみ作動します。このデータベース・パーティションの結果は、他のデータベース・パーティションに外挿されます。このデータベース・パーティションにこの表の必要な部分が含まれていない場合、その要求は、そのデータベース・パーティション・グループの中の、必要なデータが含まれる最初のデータベース・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 コマンド行から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、異なる分散オプションで収集するには、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列の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
```

- 列 ZIP および YTDTOTAL と、REGION および TERRITORY を含む列グループに関する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)
```

- 非 XML 列の統計が、**STATISTICS** パラメーターを指定した **LOAD** コマンドを使用して、前もって収集されているとします。XML 列の MISCINFO に関する統計を収集するには、以下を行います。

```
runstats on table sales.customers
on columns (miscinfo)
```

- 非 XML 列のみに関する統計を収集するには、以下を行います。

```
runstats on table sales.customers
excluding xml columns
```

EXCLUDING XML COLUMNS 節は、XML 列を指定する他のすべての節に優先します。

- DB2 V9.7 フィックスパック 1 以降のリリースでは、以下のコマンドによって、XML 列 MISCINFO の最大 50 変位値を使用する分散統計が収集されます。表内の他のすべての列に関しては、デフォルトの 20 変位値が使用されません。

```
runstats on table sales.customers
with distribution on columns ( miscinfo num_quantiles 50 )
default num_quantiles 20
```

注: XML 列 MISCINFO で分散統計が収集されるためには、以下が必要となります。

- 表統計と分散統計の両方が収集されなければなりません。
- その列に対して XML データの索引が定義されていなければならず、その索引に指定されているデータ・タイプが VARCHAR、DOUBLE、TIMESTAMP、または DATE でなければなりません。

分散統計の使用の拡張例:

分散統計は、表データの頻度と分布に関する情報を提供します。これは、データの分布が均一でなく多くの重複がある場合に、オプティマイザーが照会アクセス・プランを作成する上で役立ちます。

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ために、以下の例が役立ちます。

頻出値統計の例

C1 = KEY の形式の等価述部を含む照会について考えます。頻出値統計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・プランを選択できます。

- KEY が N 個の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 個の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布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、次の公式 (1) によって見積もられます。

$$\frac{\text{CARD} - \text{NUM_FREQ_ROWS}}{\text{COLCARD} - N}$$

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。

例えば、列 C1 のデータ値の頻度が以下のようにになっている場合を考慮します。

データ値	頻度
1	2
2	3
3	40
4	4
5	1

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想定すると、オプティマイザーは、この述部を満たす行の数は 50/5 = 10 として見積もり、-75% のエラーになります。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、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。

$$\frac{\text{見積もられた行数} - \text{実際の行数}}{\text{実際の行数}} \times 100$$

$$\frac{10 - 2}{2} \times 100 = 400\%$$

頻出値統計 ($N = 1$) を使用すると、オブティマイザーは、この値を持つ行の数を、前述の公式 (1) を使用して以下のように見積もります。

$$\frac{(50 - 40)}{(5 - 1)} = 3$$

また、エラー率は、次のように 1 桁減ります。

$$\frac{3 - 2}{2} = 50\%$$

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 変位値を計算するには、列値を昇順にソートします。 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、オブ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 C1 があるとします。

```
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

- ちょうど 7 つの行が述部 $C \leq 8.5$ を満たします。データ分布が均一であると仮定し、次の公式 (2) を考えます。

$$\frac{\text{KEY2} - \text{KEY1}}{\text{HIGH2KEY} - \text{LOW2KEY}} \times \text{CARD}$$

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。

$$\frac{8.5 - 5.1}{93.6 - 5.1} \times 10 \approx 0$$

≈ 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、8.5 (変位値の 1 つにおける最大値) に対応する K の値 7 を使用して、この述部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。

- ちょうど 8 つの行が述部 $C \leq 10$ を満たします。オプティマイザーにおいてデータ分布が均一であると想定して公式 (2) が使用される場合、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 $C \leq 8.5$ を満たす行の数、 r_2 は、述部 $C > 8.5$ かつ $C \leq 10$ 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間を使用します。

$$r_2 \approx \frac{10 - 8.5}{100 - 8.5} \times (8.5 \text{ より大きく } 100.0 \text{ 以下である値を持つ行の数})$$

$$r_2 \approx \frac{10 - 8.5}{100 - 8.5} \times (10 - 7)$$

$$r_2 \approx \frac{1.5}{91.5} \times (3)$$

$$r_2 \approx 0$$

最終的な見積もりは、 $r_1 + r_2 \approx 7$ であり、エラーは -12.5% のみです。

これら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 5 から 10 までの範囲で「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 から 100 まで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します。

データ値	頻度
20	5
30	5
40	15
50	50
60	15
70	5
80	5

次のように、 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。

ちょうど 10 個の行が述部 C BETWEEN 20 AND 30 を満たします。データ分布が均一であると仮定し、公式 (2) 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。

$$\frac{30 - 20}{70 - 30} \times 100 = 25$$

エラーは 150% です。

頻出値統計と変位値統計を使用すると、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 (1) を使用すると、 r_1 は次のように見積もられます。

$$\frac{100 - 80}{7 - 3} = 5$$

線形補間を使用すると、 r_2 は次のように見積もられます。

$$\begin{aligned} & \frac{30 - 20}{40 - 20} \times (20 \text{ より大きく } 40 \text{ 以下である値を持つ行の数}) \\ &= \frac{30 - 20}{40 - 20} \times (25 - 5) \\ &= 10 \end{aligned}$$

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。

手動での分散統計の更新の規則:

SYSSTAT.COLDIST カタログ・ビュー内の統計を更新する際には、以下の指針に従ってください。

- 頻出値統計:

- 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。
- COLVALUE の値の数は、その列の固有値の数 (この数は SYSSTAT.COLUMNS.COLCARD に保管されている) 以下でなければなりません。
- VALCOUNT の値の合計数は、その列の行の数 (SYSSTAT.TABLES.CARD に保管されている) 以下でなければなりません。

- ほとんどの場合、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 SYSSTAT.COLUMNS の HIGH2KEY および LOW2KEY に保管されている) との間にあるはずですが、HIGH2KEY より大きい頻出値が 1 つと LOW2KEY より小さい頻出値が 1 つ存在することができます。
- 変位値統計:
 - 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。
 - VALCOUNT の値は、SEQNO の値が増加するにつれて、増加しなければなりません。
 - COLVALUE の最大値に対応する VALCOUNT 内の項目は、その列の行数と等しくなければなりません。
 - ほとんどの場合、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 SYSSTAT.COLUMNS の HIGH2KEY および LOW2KEY に保管されている) との間にあるはずですが、

行数が R 個である列 $C1$ で分散統計を使用できる場合に、データ値の相対比率を同じに保ったまま、行数を $(F \times R)$ にした列に対応するように、統計を変更するとします。頻出値統計または変位値統計を F 倍するには、VALCOUNT の各項目を F 倍します。

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.ROUTINES カタログ・ビューを編集します。

runstats ユーティリティでは、UDF の統計は収集されません。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。使用できる統計がないなら、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。

表 82 のリストは、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、カタログ・ビューの列を示しています。ユーザーが変更できるのは、SYSSTAT.ROUTINES (SYSCAT.ROUTINES ではない) の列値だけである点に注意してください。

表 82. 関数統計 (SYSCAT.ROUTINES および SYSSTAT.ROUTINES)

統計	説明
IOS_PER_INVOC	関数が呼び出されるたびに実行される読み取りまたは書き込み要求の数の見積値
INSTS_PER_INVOC	関数が呼び出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	入力引数バイトごとに実行される読み取りまたは書き込み要求の数の見積値
INSTS_PER_ARGBYTE	入力引数バイトごとに実行されるマシン語命令数の見積値
PERCENT_ARGBYTES	関数が実際に処理する入力引数バイトの平均比率 (%) の見積値

表 82. 関数統計 (SYSCAT.ROUTINES および SYSSTAT.ROUTINES) (続き)

統計	説明
INITIAL_IOS	関数が最初または最後に呼び出されるときに実行される読み取りまたは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

例えば、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 である EU_SHOE について考えます。この UDF の場合、SYSSTAT.ROUTINES にある統計列の値を次のように設定します。

- INSTS_PER_INVOC。次のことを行うために必要なマシン語命令数の見積値に設定します。
 - 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値に設定します。
- PERCENT_ARGBYTES。100 に設定します。入力ストリング全体を変換することになります。
- INITIAL_INSTS、IOS_PER_INVOC、IOS_PER_ARGBYTE、および INITIAL_IOS。この UDF は計算しか実行しないため、それぞれ 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。例えば、2 つの引数を入力とし、第 2 引数の中で、第 1 引数が現れる最初の出現の開始位置を返す UDF である LOCATE について考えます。第 1 引数の長さが第 2 引数と比べると十分に小さく、第 2 引数の平均 75 % が検索されるものとします。この情報と次の想定に基づいて、PERCENT_ARGBYTES は 75 に設定されます。

- 2 回に 1 回は、最初の引数は検出されず、2 番目の引数全体が検索される
- 最初の引数は 2 番目の引数内のどの場所にも現れる可能性があるため、最初の引数が検出されるとき、2 番目の引数の半分 (平均) が検索される

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。この数は、スクラッチパッド域の設定などのためのコストを表すことがあります。

UDF 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、プログラミング言語コンパイラによって、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します。

RUNSTATS 操作の進行状況のモニター

LIST UTILITIES コマンドまたは **db2pd** コマンドを使用して、データベース上の **RUNSTATS** 操作の進行状況をモニターできます。

手順

LIST UTILITIES コマンドを発行して、**SHOW DETAIL** パラメーターを指定します。

```
list utilities show detail
```

または、**db2pd** コマンドを発行して、**-runstats** パラメーターを指定します。

```
db2pd -runstats
```

タスクの結果

次に示すのは、**LIST UTILITIES** コマンドを使用して **RUNSTATS** 操作でのパフォーマンスをモニターするときの出力例です。

```
ID = 7
Type = RUNSTATS
Database Name = SAMPLE
Partition Number = 0
Description = YIWEIANG.EMPLOYEE
Start Time = 08/04/2011 12:39:35.155398
State = Executing
Invocation Type = User
Throttling:
  Priority = Unthrottled
```

次に示すのは、**db2pd** コマンドを使用して **RUNSTATS** 操作でのパフォーマンスをモニターするときの出力例です。

```
db2pd -runstats
```

```
Table Runstats Information:
```

```
Retrieval Time: 08/13/2009 20:38:20
TbspaceID: 2      TableID: 4
Schema: SCHEMA   TableName: TABLE
Status: Completed Access: Allow write
Sampling: No      Sampling Rate: -
Start Time: 08/13/2009 20:38:16 End Time: 08/13/2009 20:38:17
Total Duration: 00:00:01
Cur Count: 0      Max Count: 0
```

```
Index Runstats Information:
```

```
Retrieval Time: 08/13/2009 20:38:20
TbspaceID: 2      TableID: 4
Schema: SCHEMA   TableName: TABLE
Status: Completed Access: Allow write
Start Time: 08/13/2009 20:38:17 End Time: 08/13/2009 20:38:18
Total Duration: 00:00:01
Prev Index Duration [1]: 00:00:01
Prev Index Duration [2]: -
Prev Index Duration [3]: -
Cur Index Start: 08/13/2009 20:38:18
Cur Index: 2      Max Index: 2      Index ID: 2
Cur Count: 0      Max Count: 0
```

モデル化および what-if の計画のカatalog統計

計画に役立てるため、システム・カATALOG内の特定の統計情報を変更したときのデータベース・パフォーマンスへの影響を調べることができます。

選択されたシステム・カATALOG統計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。

- 実動システム統計を使用した開発システム上での照会パフォーマンスのモデル化
- 「what-if」照会パフォーマンス分析の実行

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。それを行ってしまうと、動的 SQL または XQuery ステートメント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。

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。DATAACCESS 権限を保持しているユーザーは、SYSSTAT スキーマで定義されているビューに対して UPDATE ステートメントを実行して、それらの統計列の値を変更できます。

DATAACCESS 権限のないユーザーは、CONTROL 特権があるオブジェクトの統計が入った行しか表示できません。DATAACCESS 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特権があれば、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。これらの更新が影響を与えるのは、ローカル・メタデータだけであり (データ・ソース表統計は変更されません)、DB2 オプティマイザーが生成するグローバル・アクセス戦略だけに影響を与えるという点に注意してください。
- ユーザー定義関数 (UDF) の所有権

以下のコードは、EMPLOYEE 表の統計を更新するときの一例です。

```
update sysstat.tables
set
  card = 10000,
  npages = 1000,
  fpages = 1000,
  overflow = 2
where tabschema = 'MELNYK'
and tabname = 'EMPLOYEE'
```

カATALOG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。以下の方法のいずれかを使用して、開発システム上の統計を整合状態に戻すことができます。

- 手動で変更が加えられた作業単位をロールバックする (その作業単位がまだコミットされていないことが前提です)。
- runstats ユーティリティを使用してカATALOG統計をリフレッシュする

- カタログ統計を更新して、統計が収集されていないことを明示する。例えば NPAGES 列の値を -1 に設定すると、この統計が収集されていないことが明示されます。
- 行った変更を取り消す。この方法は、**db2look** コマンドを使って変更前の統計をキャプチャーした場合に限り可能です。

何らかの値または値の組み合わせが無効と判断された場合、オプティマイザーはデフォルト値を使用し、警告を返します。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にしかありません。

実動データベースのモデル化の統計:

開発システムに、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、開発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。

場合によっては、開発システムのカタログ統計と構成が実動システムのカタログ統計と構成に一致するように更新することが必要です。

db2look コマンドを模擬モード (-m オプションを指定) で使用すると、開発データベースと実動データベースのカタログ統計を一致させるために必要なデータ操作言語 (DML) ステートメントを生成できます。

db2look によって生成された UPDATE ステートメントを開発システムに対して実行すると、そのシステムを使用して、実動システム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。オプティマイザーは表スペースの構成を使用して入出力コストを見積もるので、開発システム上のそれぞれの表スペースは実動システムと同じタイプ (SMS か DMS のいずれか) でなければならず、それらを構成するコンテナの数も実動システムと同じでなければなりません。テスト・システムの物理メモリーは、実動システムよりも少ないことがあります。テスト・システムのメモリー関連の構成パラメーターを、実動システムのそうした構成パラメーターの値と同じように設定することはできない場合があります。**db2fopt** コマンドを使用して、ステートメント・コンパイル時にオプティマイザーが使用する値を割り当てることができます。例えば、実動システムが **sortheap=20000** という設定で稼働し、テスト・システムは **sortheap=5000** という設定でのみ動作できる場合は、テスト・システムで **db2fopt** を使用することによって **opt_sortheap** を 20000 に設定できます。ステートメント・コンパイル時に照会オプティマイザーがアクセス・プランを評価する際に、**sortheap** の代わりに **opt_sortheap** が使用されるようになります。

カタログ統計への手動更新の回避

DB2 データ・サーバーでは、カタログ統計の手動での更新がサポートされます。この更新を行うには、SYSSTAT スキーマ内のビューに対して UPDATE ステートメントを発行します。

このフィーチャーは、照会アクセス・プランを検討するために、テスト・システム上で実動データベースを模倣する際に役立ちます。**db2look** ユーティリティーは、別のシステムで再生するために、SYSSTAT スキーマ内のビューに対する DDL および UPDATE ステートメントをキャプチャーするのに非常に役立ちます。

不正確な統計を手動で加えて特定の照会アクセス・プランを強制することにより、照会オプティマイザーに影響を与えることがないようにしてください。この手法により、いくつかの照会のパフォーマンスが改善されることもあります。他の照会でのパフォーマンス低下を招きかねません。この手法に頼る前に、他のチューニング・オプション（最適化ガイドラインおよびプロファイルの使用など）を考慮してください。それでも、この手法が必要になる場合は、復元の必要に備えて必ず元の統計を記録しておいてください。

RUNSTATS の影響の最小化

RUNSTATS のパフォーマンスを改善するのに使用可能な方法がいくつかあります。

このユーティリティーのパフォーマンスへの影響を最小化するには、次のようにします。

- **COLUMNS** 節を使用して統計が収集される列を制限します。照会ワークロードにおいて、列の多くは述部によって参照されることがないため、統計を必要としません。
- データの分散が均一である傾向があるなら、分散統計が収集される列を制限します。分散統計の収集は、基本的な列統計の収集よりも多くの CPU およびメモリーを必要とします。しかし、列の値が均一に分散しているかどうかを判断するには、既存の統計があること、またはデータを照会することが求められます。また、この方法では、表が変更されてもデータが均一に分散したままであることが前提となります。
- ページ・レベルまたは行レベルの表サンプリングを使用し (**TABLESAMPLE SYSTEM** または **TABLESAMPLE BERNOULLI** 節を指定)、またページ・レベルまたは行レベルの索引サンプリングを使用して (**INDEXSAMPLE SYSTEM** または **INDEXSAMPLE BERNOULLI** 節を指定) 処理される、ページおよび行の数を制限します。
TABLESAMPLE SYSTEM(10) および **INDEXSAMPLE SYSTEM(10)** を指定して、10% のページ・レベル・サンプルから始めます。統計の正確性を確認し、アクセス・プランの変更によりシステム・パフォーマンスが低下していないかどうかを確認します。低下しているなら、**TABLESAMPLE BERNOULLI(10)** を使用して、10% の行レベル・サンプルを代わりに試します。同様に、索引サンプリングに適正な比率を取得するために、**INDEXSAMPLE** パラメーターを試します。統計の正確性が不十分であるなら、サンプルの量を増やします。**RUNSTATS** ページ・レベルまたは行レベルのサンプルを使用する場合、結合する表についても同じサンプリング・レートを使用してください。これは、結合列の統計で同レベルの正確性を確保する上で重要です。
- **CREATE INDEX** ステートメントで **COLLECT STATISTICS** オプションを指定することにより、索引の作成中に索引統計を収集します。これは、索引の作成後に **RUNSTATS** 操作を別に行うよりも速い方法です。また、作成直後に生成された統計を新規索引で確保し、索引使用のコストをオプティマイザーによって正確に見積もることが可能となります。
- **LOAD** コマンドの実行時に、**REPLACE** オプションを使用して統計を収集します。これは、ロード操作後に **RUNSTATS** 操作を別に行うよりも速い方法です。また、データのロード直後に最新の統計を表で確保し、表を使用するコストをオプティマイザーによって正確に見積もることが可能となります。

パーティション・データベース環境において、**RUNSTATS** ユーティリティーは、単一のデータベース・パーティションから統計を収集します。表が存在するデータベ

ス・パーティションで **RUNSTATS** コマンドを発行すると、そこで統計が収集されます。表が存在しない場合、表のデータベース・パーティション・グループにおける 1 番目のデータベース・パーティションで、統計が収集されます。統計の整合性を取るには、同じデータベース・パーティションから結合表の統計を収集するようにしてください。

データ圧縮とパフォーマンス

データ圧縮を使用すると、ディスクから読み取らなければならないデータ量やディスクに書き込まなければならないデータ量を減らして、入出力コストを削減することができます。

一般的に、データ行中に存在する反復パターンが多ければ多いほど、圧縮率は高くなります。データに反復ストリングが含まれていない場合 (数値データが主である場合や、インライン BLOB データが入っている場合など) は、圧縮してもストレージの節約の点でそれほど効果が得られない可能性があります。さらに、データがあまり圧縮されないにも関わらず、データを圧縮しようとするによりデータベース・マネージャーのパフォーマンスに影響が及び、ストレージに関する利点がコンプレッション・ディクショナリーを保持する負担に見合わない場合があります。圧縮によって得られるストレージ推定節約量を確認するには、**ADMIN_GET_TAB_COMPRESS_INFO** 表関数を使用します。

以下の 2 つの形式のデータ圧縮を使用できます。

行圧縮 以下の 2 つのタイプの行圧縮を使用できます。

クラシック行圧縮

クラシック行圧縮では、1 つの行の中で複数の列値にまたがる反復パターンをより短いシンボル・ストリングに置き換えます。反復データのインスタンスを見つけるために、サンプリングした行がスキャンされます。このスキャンから、表レベルのコンプレッション・ディクショナリーが作成されます。このディクショナリーを使用して、反復データをより短いシンボル・ストリングで置き換えます。

表レベルのコンプレッション・ディクショナリーは静的です。つまり、これらのディクショナリーは、最初に作成してしまうとその後は、従来の表再編成中に再作成しない限り変更されません。

アダプティブ行圧縮

アダプティブ行圧縮では、クラシック行圧縮とページ・レベルの圧縮の 2 つの圧縮手法が使用されます。ページ・レベルの圧縮では、データの単一ページ内の複数の列値にまたがる反復パターンが、より短いシンボル・ストリングで置き換えられます。ページがデータで埋まると、反復データがないかページ圧縮ロジックがそのページをスキャンします。このスキャンから、ページ・レベルのコンプレッション・ディクショナリーが作成されます。このディクショナリーを使用して、反復データをより短いシンボル・ストリングで置き換えます。

ページ・レベルのディクショナリーは動的であり、これらは、必要に応じて自動的に再作成されます。

値圧縮 値圧縮では、値の重複項目を除去し、1つのコピーだけを保管し、格納された値への参照の場所に関するトラッキングを保持します。

行圧縮は一時表にも使用されます。一時表を圧縮すると、大規模で複雑な照会に必要な一時ディスク・スペースの量が減り、照会のパフォーマンスが向上します。DB2 Storage Optimization Feature では、一時表の圧縮が自動的に使用可能になります。行圧縮に適格な各一時表には、それぞれの表のコンプレッション・ディクショナリーの作成用にさらにメモリーが 2 から 3 MB 必要になります。このメモリーは、コンプレッション・ディクショナリーの作成が完了するまで割り振られたままになります。

索引オブジェクトを圧縮することもできます。これにより、ストレージ・コストが削減されます。この種の圧縮が特に役立つのは、非常に大きな索引を数多く使用することが一般的である大規模なオンライン・トランザクション処理 (OLTP) 環境やデータウェアハウス環境の場合です。どちらの場合も、索引圧縮を行うことにより、入出力制約のある環境では大幅にパフォーマンスが改善され、CPU 制約のある環境ではパフォーマンスの低下はほとんど、あるいは全く見られません。

XML 列のある表で圧縮が使用可能である場合、XDA オブジェクトに保管されている XML データも圧縮されます。XML データの別個のコンプレッション・ディクショナリーは、XDA オブジェクトに保管されています。DB2 バージョン 9.7 以降を使用して、DB2 バージョン 9.5 を使用して作成された表に XML 列を追加すると、これらの XML 列は圧縮されます。ただし、DB2 バージョン 9.5 を使用して表に追加された XML 列は圧縮されません。そのような表ではデータ・オブジェクトだけが圧縮されます。

DML のパフォーマンスを向上させるためのロギング・オーバーヘッドの低減

データベース・マネージャーは、データベースのすべての変更内容を記録するログ・ファイルを維持します。ロギング・ストラテジーには、循環ロギングとアーカイブ・ロギングの 2 つがあります。

- 循環ロギング を使用すると、使用可能なファイルが満杯になると、ログ・ファイルが再利用 (初期ログ・ファイルを始めとして) 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。
- アーカイブ・ロギング を使用すると、ログ・ファイルがログ・レコードで満杯になるとアーカイブされます。ログを保持するとロールフォワード・リカバリーが可能になります。ロールフォワード・リカバリーによって、ログ・ファイルに記録されたデータベースに対する変更内容 (完了した作業単位またはトランザクション) を災害時リカバリーに再び適用できます。

正規データおよび索引ページになされた変更はすべてがログ・バッファーに書き込まれてから、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、SQL ステートメントの処理は、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まで。これは DB2 サーバーが書き込み先行ロギングを使用するためです。この場合、COMMIT ステートメン

トでトランザクションが完了しても、変更されたすべてのデータおよび索引ページをディスクに書き込む必要はありません。

- 変更 (ほとんどはデータ定義言語ステートメントの実行結果によって生じる) がメタデータに加えられるまで。
- ログ・バッファが満杯になるとき

データベース・マネージャーは、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・データをディスクに書き込みます。たくさんの短いトランザクションが同時に処理される場合、ほとんどの遅延は、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みます。ログ入出力によってさらに遅延が生じます。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** データベース構成パラメーターの値を 1 より大きく設定します。この設定により、いくつかのアプリケーションの COMMIT の遅延はさらに長くなりますが、1 つの操作で書き込まれるログ・データの量が多くなる場合があります。

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。ログ保持が指定され、LOB 列が CREATE TABLE ステートメントで NOT LOGGED 節なしで定義されていないかぎり、LOB 列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。インライン LOB 値は、VARCHAR 値であるかのように、ロギングの更新、挿入、または削除に全面的に関与します。

インライン LOB によってパフォーマンスを改善する

一部のアプリケーションでは、ラージ・オブジェクト (LOB) を広範囲に使用します。多くの場合、こうした LOB のサイズはそれほど大きくなく、ほとんどの場合 2、3 K バイトです。LOB データ・アクセスのパフォーマンスは、そうした LOB データを LOB ストレージ・オブジェクト内ではなく、データ・ページ上のフォーマット済み行内に配置することによって改善できるようになりました。

そうした LOB は、インライン LOB と呼ばれます。これまでは、そうした LOB の処理によってアプリケーションのボトルネックが作成される可能性がありました。インライン LOB では、LOB データのフェッチ、挿入、または更新のために追加の入出力は不要なので、LOB データにアクセスする照会のパフォーマンスが改善されます。また、インライン LOB データは行圧縮にも適しています。

このフィーチャーは、CREATE TABLE ステートメントまたは ALTER TABLE ステートメントの INLINE LENGTH オプションを使用して有効にします。INLINE LENGTH オプションは、構造化タイプ、XML タイプ、または LOB 列に適用されます。LOB 列の場合、インライン長は基本表行に格納できる LOB 値の最大バイト・サイズ (オーバーヘッド用の 4 バイトを含む) を示します。

このフィーチャーは、新規表または既存の表 (LOB 列が追加されるとき) のすべての LOB 列、およびデータベースのアップグレードにおけるすべての既存の LOB 列でも暗黙的に使用可能です。すべての LOB 列は、定義された最大サイズに基づ

いた行スペースを予約しています。LOB 列ごとの暗黙的な `INLINE LENGTH` 値は、自動的に定義され、明示的に指定されたかのように保管されます。

インラインに保管できない LOB 値は、LOB ストレージ・オブジェクトに別個に保管されます。

表にインライン LOB を持つ列がある場合は 1 ページに収まる行数が減り、非 LOB データのみを戻す照会のパフォーマンスが好ましくない影響を受けることに注意してください。LOB のインライン化は、ほとんどのステートメントに 1 つ以上の LOB 列が含まれているワークロードに役立ちます。

LOB データは必ずしもログに記録されるわけではありませんが、インライン LOB は常にログに記録されます。従って、ロギングのオーバーヘッドが大きくなる可能性があります。

第 4 章 パフォーマンス・チューニング手段の設定

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、どの索引、マテリアライズ照会表 (MQT)、クラスタリング・ディメンション、またはデータベース・パーティションを作成するかを選択する作業は、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
- 新規クラスタリング索引
- 新規 MQT
- マルチディメンション・クラスタリング (MDC) 表への変換
- 表の再配分

設計アドバイザーは、こうした推奨の一部またはそのすべてをすぐにインプリメントすることもできますし、後で実行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・ユーティリティを起動するには、**db2adv** コマンドを使用します。

設計アドバイザーでは、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およびセットアップ

データベースの設計の際に、設計アドバイザーを使用して、索引作成、MQT、MDC 表、またはデータベース・パーティション作成のための代替の設計をテスト環境に生成します。

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ます。

- データベースにデータをロードする前に、適切なデータベース・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・データベースから複数パーティション・データベースへのアップグレードを支援する
- 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

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

- IBM InfoSphere Optim Performance Manager など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
- 新しい照会のパフォーマンスを最適化する方法を判別する
- 共有メモリー・ユーティリティーに関する Data Studio ヘルス・モニターからの推奨、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同ヘルス・モニターからの推奨に応じた処置をする
- 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

IBM InfoSphere Optim Query Workload Tunerは、単一の SQL ステートメントのパフォーマンスと SQL ステートメント・グループのパフォーマンス (照会ワークロードと呼ばれる) を向上するツールを提供しています。この製品について詳しくは、<http://www.ibm.com/software/data/optim/query-workload-tuner-db2-luw/index.html>の製品概要のページを参照してください。また、バージョン 3.1.1 以降では、ワークロード設計アドバイザーを使用して、DB2 設計アドバイザー・ウィザードで利用できた多くの操作を実行できます。詳しくは、<http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.datatools.qrytune.workloadtunedb2luw.doc/topics/genrecsdsgn.html>のワークロード設計アドバイザーの資料を参照してください。

設計アドバイザーの出力

設計アドバイザーの出力はデフォルトでは標準出力に書き込まれ、次のように ADVISE_* 表に保存されます。

- ADVISE_INSTANCE 表は、設計アドバイザーが実行されるたびに、次のように 1 行が新しく更新されます。
 - START_TIME フィールドおよび END_TIME フィールドは、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 フィールドに値 COMPLETED が入ります。
 - MODE フィールドは、**db2advise** コマンド上で **-m** パラメーターが使用されたかどうかを示します。
 - COMPRESSION フィールドは、使用された圧縮タイプを示します。
- MQT、MDC 表、またはデータベース・パーティション化ストラテジーの推奨が出された場合には、ADVISE_TABLE 表の USE_TABLE 列には値 Y が入ります。

MQT 推奨は ADVISE_MQT 表にあります。MDC 推奨は ADVISE_TABLE 表にあります。データベース・パーティション化ストラテジー推奨は ADVISE_PARTITION 表にあります。これらの表の RUN_ID 列には ADVISE_INSTANCE 表の行の START_TIME 値に対応する値が入るため、ADVISE_INSTANCE 表は、設計アドバイザーによる同一の実行に関してリンクされます。

MQT、MDC、またはデータベース・パーティションの推奨が提供されている場合、関連する ALTER TABLE ストアード・プロシージャ呼び出しが ADVISE_TABLE 表の ALTER_COMMAND 列に置かれます。ALTER TABLE ストアード・プロシージャ呼び出しは、ALTOBJ ストアード・プロシージャに関する表の制限のために成功しない場合があります。

- 索引の推奨が出された場合には、ADVISE_INDEX 表の USE_INDEX 列には、値 Y (索引が推奨されたか評価された) または R (既存のクラスタリング RID 索引を非クラスタ化するように推奨された) が入ります。
- ADVISE_MQT 表の COLSTATS 列には、MQT の列統計が入ります。これらの統計は、以下のように XML 構造内に組み込まれます。

```
<?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>
```

db2adv コマンドで **-o** パラメーターを使用すると、設計アドバイザーの推奨をファイルに保存できます。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。

- 索引、MQT、MDC 表、またはデータベース・パーティション化の新しいストラテジーに関連した CREATE ステートメント
- MQT の REFRESH ステートメント
- 新規オブジェクトの **RUNSTATS** コマンド

この出力の例は以下のとおりです。

```
--<?xml version="1.0"?>
--<design-advisor>
--<mqt>
--<identifier>
--<name>MQT612152202220000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<statementlist>3</statementlist>
--<benefit>1013562.481682</benefit>
--<overhead>1468328.200000</overhead>
--<diskspace>0.004906</diskspace>
--</mqt>
.....
--<index>
--<identifier>
--<name>IDX612152221400000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<table><identifier>
--<name>PART</name>
--<schema>TPCD </schema>
--</identifier></table>
--<statementlist>22</statementlist>
--<benefit>820160.000000</benefit>
--<overhead>0.000000</overhead>
--<diskspace>9.063500</diskspace>
--</index>
.....
```

```

--<statement>
--<statementnum>11</statementnum>
--<statementtext>
--
-- select
-- c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice,
-- sum(l_quantity) from tpcd.customer, tpcd.orders,
-- tpcd.lineitem where o_orderkey in( select
-- l_orderkey from tpcd.lineitem group by l_orderkey
-- having sum(l_quantity) > 300 ) and c_custkey
-- = o_custkey and o_orderkey = l_orderkey group by
-- c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
-- order by o_totalprice desc, o_orderdate fetch first
-- 100 rows only
--</statementtext>
--<objects>
--<identifier>
--<name>MQT612152202490000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>ORDERS</name>
--<schema>TPCD </schema>
--</identifier>
--<identifier>
--<name>CUSTOMER</name>
--<schema>TPCD </schema>
--</identifier>
--<identifier>
--<name>IDX612152235020000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>IDX612152235030000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>IDX612152211360000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--</objects>
--<benefit>2091459.000000</benefit>
--<frequency>1</frequency>
--</statement>

```

この XML 構造には複数の列を含めることができます。列カーディナリティー (つまり、各列内の値の数) が組み込まれ、オプションで HIGH2KEY および LOW2KEY の値が組み込まれます。

索引が定義されている基本表も組み込まれます。索引および MQT のランキングは利点値を使用して行えます。また、ランク付けを行うこともできます。索引の場合は (利点 - オーバーヘッド) を使用し、MQT の場合は (利点 - 0.5 * オーバーヘッド) を使用します。

索引および MQT のリストの後に、ワークロードに含まれるステートメントのリストが続きます。これには、SQL テキスト、ステートメントのステートメント番号、推奨値から見積もられるパフォーマンスの改善 (利点)、およびステートメントによって使用された表、索引、および MQT のリストが含まれます。この出力例では SQL テキストの元のスペーシングが保たれていますが、通常、SQL テキストは、読みやすさを向上させるために 80 文字のコメント行に分割されています。

ワークロードの実行に既存の索引または MQT が使用されている場合、それらは出力に組み込まれます。

MDC およびデータベース・パーティション化の推奨については、この XML 出力例に明示的に示されていません。

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 の代わりに、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データ定義言語 (DDL) を、除去またはコメント化する

設計アドバイザーの使用

db2adv コマンドを呼び出すことにより、設計アドバイザーを実行できます。

手順

1. ワークロードを定義します。『設計アドバイザーのワークロードの定義』を参照してください。
2. このワークロードに対して **db2adv** コマンドを実行します。

注: データベース上の統計が最新のものではない場合、生成される推奨の信頼性が低下します。

3. **db2adv** からの出力を解釈し、必要な変更を行います。
4. 適宜、設計アドバイザーの推奨を実際に設定します。

設計アドバイザーのワークロードの定義

設計アドバイザーは、特定のワークロードを分析する際、ワークロードに組み込まれた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。

このタスクについて

ワークロードとは、データベース・マネージャーが特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。設計アドバイザーの実行対象は以下のとおりです。

- **db2adv** コマンドによりインラインで入力した、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

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・ファイルを変更することができます。以下に示すような種々のソースから、ステートメントをファイルに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

- EXPLAIN_STATEMENT 表内の EXPLAIN されたステートメント
- 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
- ワークロード・マネージャー・アクティビティー表
- ワークロード・マネージャー・イベント・モニター表 (コマンド行から **-wlm** オプションを使用する)

SQL ステートメントをワークロード・ファイルにインポートした後で、ステートメントの追加、変更、修正、または削除、ならびにステートメントの頻度の変更を実行できます。

手順

- 設計アドバイザーを動的 SQL ステートメントに対して実行するには、以下のようになります。
 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. データベースに対して動的 SQL ステートメントを実行できるよう、しばらく待機します。
 3. **-g** パラメーターを使用して、**db2adv** コマンドを実行します。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** パラメーターも使用します。
- ワークロード・ファイル内にある一式の SQL ステートメントに対して設計アドバイザーを実行するには、以下のようになります。
 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・ファイル内にあるすべての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。例えば、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す。これを行うには、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. **-i** パラメーターとワークロード・ファイルの名前を使用して、**db2adv** コマンドを実行します。
- ADVISE_WORKLOAD 表に含まれるワークロードに対して設計アドバイザーを実行するには、**-w** パラメーターとワークロード名を指定して **db2adv** コマンドを実行します。

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・データベースへの変換

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・データベースへの変換の際に、設計アドバイザーを役立てることができます。

このタスクについて

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) 表についての推奨に加えて、設計アドバイザーはデータの分散についても推奨できます。

手順

1. **db2licm** コマンドを使用して、パーティション・データベース環境のライセンス・キーを登録します。
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データ再配分のみ推奨できません。

3. **db2adv** コマンドにパーティション・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. **db2adv** 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。設計アドバイザーによって生成された DDL スクリプトを実行できるようにするには、まずデータベース・パーティションをセットアップする必要があるため、返されるスクリプトで推奨内容はコメント化されています。推奨内容にしたがって表をトランスフォームする操作は、ご自分で行ってください。

設計アドバイザーの制限と制約事項

索引、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、およびデータベース・パーティションについての設計アドバイザーの推奨に関連した特定の制限と制約事項が存在します。

索引に関する推奨事項の制限

- 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。
- クラスタリング RID 索引は、MDC 表に対してのみ推奨されています。設計アドバイザーは、表の MDC 構造を作成するのではなく、クラスタリング RID 索引をオプションとして組み込みます。
- バージョン 9.7 の設計アドバイザーは、パーティション表上のパーティション索引を推奨しません。すべての索引は、明示的な NOT PARTITIONED 節とともに推奨されます。

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、選択したステージング表でインクリメンタル MQT 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。
- ワークロードに更新、挿入、または削除操作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索引を 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。

MDC に関する推奨事項の制限

- 設計アドバイザーが表の MDC を考慮する前に、既存の表には十分なデータが移植されなければなりません。最小でも 24 から 30 MB のデータを入れておくことが勧められています。12 エクステントより小さい表は、考慮対象外です。
- サンプリング・オプション `r` が `db2advvis` コマンドに指定されていない場合、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、型付き表、一時表、また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。
- `db2advvis` コマンドの実行中に使用するサンプリング・データ用に十分な大きさのストレージ・スペース (大きな表の場合は表データの約 1%) が必要です。
- 収集された統計がない表は、考慮対象外です。
- 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。

データベース・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 でのデータベース・パーティション化のみを推奨します。

追加の制限

一時的なシミュレーション・カタログ表が、設計アドバイザーの実行時に作成されます。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、この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 ステートメントのみを生成します。実際には表を削除しません。

注: バージョン 9.5 の時点で、`-f` オプションはデフォルトです。これは、MQT を選択して `db2advvis` を実行する場合、データベース・マネージャーはスキーマ名と同じユーザー ID を使用して、自動的にすべてのローカル・シミュレーション・カタログ表をドロップすることを意味します。

これらのシミュレーション済みのカタログ表を保管するためにカタログ・データベース・パーティション上に別個の表スペースを作成し、CREATE または ALTER TABLESPACE ステートメントの DROPPED TABLE RECOVERY オプションを 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。

第 2 部 問題のトラブルシューティングのための技法

良い問題分析の最初のステップは、問題を完全に記述することです。問題記述なしでは、問題の原因を突き止めるためにどこから始めたらいかがが分かりません。

このステップでは、以下のような基本的な質問を自問自答します。

- どのような症状か
- 問題が発生しているのはどこか
- 問題が発生するのはいつか
- どのような条件で問題が発生するか
- 問題は再現可能か

これらの質問や他の質問に答えることは、ほとんどの問題に対する正しい記述に結びつき、問題解決し始めるための最良の方法です。

どのような症状か

問題を記述し始める際にまず考えられる質問は「何が問題か」というものです。これは平易な質問のように思えます。しかし、問題をより明確にするには、これをさらにいくつかの質問に分割することができます。以下のような質問です。

- 問題を報告しているのは誰または何か
- エラー・コードおよびエラー・メッセージはどのようなものか
- どのような障害が起きるか (ループ、ハング、停止、性能低下、結果の誤りなど)
- ビジネスにどのような影響があるか

問題が発生しているのはどこか

問題の発生箇所の判別は必ずしも容易ではありませんが、問題の解決において最も重要なステップの 1 つです。報告を行うコンポーネントと障害が起きているコンポーネントの間には、多くのテクノロジーの層が存在していることがあります。問題を突き止める際に考慮すべきコンポーネントをほんのいくつか挙げるだけでも、ネットワーク、ディスク、ドライバーなどがあります。

- 問題はプラットフォームに固有のものか、それとも複数のプラットフォームに共通のものか
- 現行の環境および構成はサポートされているか
- アプリケーションはデータベース・サーバー上でローカルに実行しているか、それともリモート・サーバー上で実行しているか
- ゲートウェイが関係しているか
- データベースは個々のディスク上に保管されているか、それとも RAID ディスク・アレイに保管されているか

これらのタイプの質問は、問題の層を切り分けるのに役立つとともに、問題の発生源を判別するのに不可欠です。1 つの層が問題を報告しているからといって、根本原因がそこに存在するとは限らないことに注意してください。

問題がどこで発生しているかを識別することには、問題が存在している環境を理解することが含まれます。必ずいくらかの時間を取って、問題の発生している環境を完全に記述する必要があります。それには、オペレーティング・システムとそのバージョン、該当するすべてのソフトウェアとバージョン、およびハードウェア情報が含まれます。実行環境が、サポートされている構成になっているかどうかを確認してください。というのは、一緒に実行するよう意図されていないソフトウェア・レベル、または一緒に十分にテストされていないソフトウェア・レベルを突き止めることで解明可能な問題は少なくないからです。

問題が発生するのはいつか

障害につながる詳細なイベントを時系列上に並べることは、問題分析における別の不可欠なステップです。一回限りの現象である場合には特にそうです。この作業の最も簡単な方法は、過去にさかのぼって行くことです。すなわち、エラーが報告された時刻（ミリ秒単位にまで至る、可能な限り正確な時刻）を起点に、入手可能なログと情報を過去にさかのぼって作業を進めていきます。通常はいずれかの診断ログで最初に見つかる疑わしいイベントまで調べれば十分です。しかし、これは必ずしも容易に行えることではなく、経験を積む必要があります。特に、テクノロジーの層が複数あり、それぞれに独自の診断情報がある場合には、どこでやめるかを判断するのは困難です。

- 問題は、昼間または夜間の特定の時刻にのみ発生するか
- どれほどの頻度で発生するか
- 問題の報告される時刻まで、どのような一連のイベントが生じているか
- 問題が発生するのは、既存のソフトウェアまたはハードウェアのアップグレードや、新規のソフトウェアまたはハードウェアのインストールといった環境の変更後か

このような質問に答えることにより、イベントを時系列に並べるのに役立ち、調査の基準枠を持つことができます。

どのような条件で問題が発生するか

問題の発生時に、他に何が実行されていたかを知ることは、完全な問題記述のために重要です。特定の環境または特定の条件下で問題が発生する場合、それが問題の原因のかぎとなる標識であることがあります。

- 同じタスクを実行すると、常にその問題が発生するか
- 問題が表面化するには、特定の順序でイベントが発生する必要があるか
- 同時に他のアプリケーションにも障害が起きるか？

これらのタイプの質問に答えることにより、問題が発生する環境を解明し、依存関係を明らかにするのに役立ちます。複数の問題がほぼ同時に発生するというだけでは、必ずしもそれらが関連していることにはならないことを覚えておいてください。

問題は再現可能か

問題記述および調査の観点から見て「理想的」な問題とは、再現可能な問題です。再現可能な問題の場合はたいてい、調査の助けになるツールやプロシーチャーのセットがより多く利用できます。その結果、再現可能な問題は、通常デバッグおよび解決をより容易にします。

しかし、再現可能な問題にも欠点があります。すなわち、ビジネス上の大きな影響を伴う問題の場合、これを繰り返し発生させるわけにはいきません。この場合、可能であれば、テストまたは開発環境で問題を再現するほうが望ましいでしょう。

- テスト・マシン上で問題を再現できるか
- 複数のユーザーまたはアプリケーションが、同じタイプの問題に遭遇するか
- 単一コマンド、コマンドのセット、特定のアプリケーション、または独立型アプリケーションのいずれかを実行して、問題を再現できるか
- DB2 コマンド行から等価のコマンドまたは照会を入力して、その問題を再現できるか

多くの場合、テストまたは開発環境で単一の問題を再現するのが望ましいでしょう。通常、そのほうが、調査を行う際により一層の柔軟性と制御が得られるからです。

第 5 章 トラブルシューティングのためのツール

診断データの収集、フォーマット、または分析に役立つツールがあります。

db2cklog ツールを使用したアーカイブ・ログ・ファイルの検査

アーカイブ・ログ・ファイルを検査すると、ロールフォワード・リカバリーが必要になった場合でも既知の適切なログ・ファイルが確実に使用可能になります。こうして、ログ・ファイルの問題のためにリカバリー操作が失敗することがなくなります。このトピックでは、**db2cklog** ツールを使ってログ・ファイルを検査する方法、またログ・ファイルの妥当性検査が失敗した場合にどうすべきかについての情報を示します。

始める前に

db2cklog ツールによってログ・ファイルを読み取って検査を実行するには、アーカイブ・ログ・ファイルに対する読み取り権限が必要です。(アーカイブ・ログ・ファイルのような) 閉じたログ・ファイルだけを、正常に妥当性検査することができます。依然としてアクティブなログ・ファイルに対してこのツールを実行した場合、ツールはそのファイルを正確に検査することができません。ファイルが依然としてアクティブであることを通知する警告を受け取ります。

このタスクについて

db2cklog ツールは単一のログ・ファイル、または番号の付いた複数のログ・ファイルから成る範囲を読み取って、ファイルの内部的な妥当性を検査します。エラー・メッセージや警告のない、妥当性検査に合格したログ・ファイルは既知の適切なファイルとなり、ロールフォワード・リカバリー操作でこれらを使用できます。エラー・メッセージが出てアーカイブ・ログ・ファイルの妥当性検査が失敗した場合、または警告が戻された場合には、ロールフォワード・リカバリーでそのログ・ファイルを使用してはなりません。妥当性検査に失敗したアーカイブ・ログ・ファイルを修復することはできません。『次の作業』で説明されている対応策に従ってください。

アーカイブ・ログ・ファイルを検査することは、次のようなシナリオで役立ちます。

- ロールフォワード操作を始める直前。ロールフォワード・リカバリー操作を実行することが必要になった場合、まず最初に、ロールフォワード操作の実行に必要なすべてのアーカイブ・ログ・ファイルに対して **db2cklog** ツールを実行して、それらのログ・ファイルが妥当であることを確認できます。ログ・ファイルに対して予め **db2cklog** ツールを実行しておくこと、ログ・ファイルの問題のためにリカバリー操作が途中で失敗して追加のリカバリー操作が必要になるような事態を防ぐのに役立ちます。
- ログ・ファイルが閉じられてログ・アーカイブ・ディレクトリーにコピーされるたびに毎回。日常的な操作の 1 つとしてアーカイブ・ログ・ファイルを検査すると、既知の適切なログ・ファイルを常に使用可能にするための付加的な予防措置となります。このような予防手段を使用すると、ログ・ファイルのコピーを見つ

ける必要があるかどうか、あるいは新しいリカバリー・ポイントを確立するためのフル・データベース・バックアップが必要かどうか直ちにわかります。こうして、ロールフォワード・リカバリーが必要になった場合の操作の遅れを軽減できます。

手順

アーカイブ・ログ・ファイルを検査するには、コマンド行から **db2cklog** コマンドを発行して、検査対象の 1 つまたは複数のログ・ファイルを含めます。**db2cklog** コマンドではログ・ファイル名の全体を指定するのではなく、ログ・ファイル名の一部である数値 ID だけを指定することに注意してください。例えば S0000001.LOG というログ・ファイルの数値 ID は 1 です。このログ・ファイルを検査するには **db2cklog 1** と指定します。アーカイブ・ログ・ファイルが現行ディレクトリーに存在しない場合は、オプションの **ARCHLOGPATH** パラメーターと共にログ・ファイルの相対パスまたは絶対パスを含めてください。

1. 単一のアーカイブ・ログ・ファイルの妥当性を検査するには、コマンドでそのログ・ファイルの数値 ID を *log-file-number1* として指定します。例えば `/home/amytang/tests` ディレクトリー内のログ・ファイル `S0000000.LOG` の妥当性を検査するには、`db2cklog 0 ARCHLOGPATH /home/amytang/tests` というコマンドを発行します。
2. 複数のアーカイブ・ログ・ファイルから成る範囲を妥当性検査するには、その範囲の最初と最後の数値 ID (*log-file-number1* から *log-file-number2* まで) をコマンドに含めます。範囲に含まれるすべてのログ・ファイルが検査されます。ただし、*log-file-number2* で指定された範囲の上限が (*log-file-number1* で指定された) 範囲の始まりより小さい数値である場合を除きます。そのような場合には *log-file-number1* だけが検査されます。例えば `/home/nrichers/tests` ディレクトリー内の `S0000000.LOG` から `S0000005.LOG` までの範囲のログ・ファイルを妥当性検査するには、`db2cklog 0 TO 5 ARCHLOGPATH /home/nrichers/tests` というコマンドを発行します。

タスクの結果

db2cklog ツールで妥当性検査に合格したすべてのファイルに対して、戻りコード 0 が戻されます。番号付きアーカイブ・ログ・ファイルの範囲を指定した場合、

db2cklog ツールは各ファイルを順番に読み取り、検査を実行して、各ファイルの戻りコードを発行します。最初のエラーが検出された時点でツールは停止します (ログ・ファイルの範囲が指定され、まだ検査されていないファイルが存在する場合であっても、ツールの動作は同じです)。エラー検出時に戻される DBT メッセージには、アーカイブ・ログ・ファイルの妥当性検査が失敗した理由について情報が示される場合がありますが、無効なログ・ファイルを修正することはできません。ログ・ファイルが依然としてアクティブである可能性を示す DBT 警告メッセージを受け取った場合、そのファイルがアーカイブ・ログ・ファイルであることが確実に分かっているならば、それを無効なアーカイブ・ログ・ファイルとして扱い、『次の作業』で説明されている対応策に従ってください。

例

以下の例は、**db2cklog** コマンドでログ・ファイル (この場合は S0000002.LOG) を構文解析したときの標準的な出力を示しています。このファイルは妥当性検査に合格し、戻りコード 0 が戻されます。

```
$ db2cklog 2
```

```
-----
                D B 2 C K L O G
                -----
                DB2 Check Log File tool
                  I   B   M

                The db2cklog tool is a utility can be used to test the integrity
                of an archive log file and to determine whether or not the log file
                can be used in the rollforward database command.

                -----

=====
"db2cklog": Processing log file header of "S0000002.LOG"

"db2cklog": Processing log pages of "S0000002.LOG" (total log pages: "316840")
==> page "1" ...
==> page "25001" ...
==> page "50001" ...
==> page "75001" ...
==> page "100001" ...
==> page "125001" ...
==> page "150001" ...
==> page "175001" ...
==> page "200001" ...
==> page "225001" ...
==> page "250001" ...
==> page "275001" ...
==> page "300001" ...

"db2cklog": Finished processing log file "S0000002.LOG". Return code: "0".
=====
```

次のタスク

アーカイブ・ログ・ファイルの妥当性検査が失敗した場合に取るべき対応策は、**db2cklog** ツールでの妥当性検査に合格するようなログ・ファイルのコピーがあるかどうかに応じて異なります。ログ・ファイルのコピーがあるかどうか分からない場合は、(各ログ・ファイルの 2 次コピーがデータベース・サーバーによってアーカイブされるかどうかを決定する) **logarchmeth2** 構成パラメーターの設定値を確認してください。アーカイブされるログをすぐに妥当性検査している場合、データ・サーバーでログ・ミラーリングも構成されているならば、ログ・ミラー・パスでログ・ファイルのコピーが見つかる可能性があります。これは、データ・サーバーがアーカイブの直後にログ・ファイルをリサイクルすることはないためです。

- アーカイブ・ログ・ファイルのコピーがある場合は、そのコピーに対して **db2cklog** コマンドを使用してください。ログ・ファイルのコピーが妥当性検査に合格した場合、読み取り不能なログ・ファイルを、妥当なログ・ファイル・コピーで置き換えます。
- アーカイブ・ログ・ファイルのコピーが 1 つだけで、そのコピーが妥当性検査に合格しない場合は、ログ・ファイルを修復することができず、ロールフォワード・リカバリーには使用できません。この場合、フル・データベース・バックアップをできるだけ早急に取りする必要があります。こうすることで、ロールフォワード・リカバリーに使用できないログ・ファイルに依存しない、より新しいリカバリー・ポイントを確立します。

db2dart ツールの概要

db2dart コマンドは、データベースとその内部のオブジェクトの体系的な正確さを検査するのに使用できます。他の方法ではアクセスできない表からデータを抽出するために、データベース制御ファイルの内容を表示するのにも使用できます。

使用可能なオプションすべてを表示するには、パラメーターを何も指定せず、**db2dart** コマンドを発行します。表スペース ID などのパラメーターを必要とするオプションによっては、コマンド行で明示的に指定されていない場合、プロンプトが出されます。

デフォルトでは、**db2dart** ユーティリティーは `databaseName.RPT` という名前のレポート・ファイルを作成します。単一パーティションのデータベース・パーティション環境の場合、ファイルは現行ディレクトリーで作成されます。複数パーティションのデータベース・パーティション環境の場合、ファイルは診断ディレクトリー中のサブディレクトリーの下に作成されます。サブディレクトリーは `DART####` で、#### はデータベース・パーティション番号です。

DB2 pureScale環境では、いくつかのメタデータ・ファイル (例えばバッファー・プール構成ファイル) がメンバーごとに存在し、(データベースごとではなく) メンバーごとに妥当性検査または更新されます。

db2dart ユーティリティーは、ディスクから直接読み取ることによってデータベース中のデータおよびメタデータにアクセスします。そのため、まだアクティブな接続があるデータベースに対してこのツールを実行してはなりません。接続が存在する場合、このツールはバッファー・プール中のページまたはメモリー内の制御構造などを認識しないので、結果として誤ったエラーが報告される可能性があります。同様に、クラッシュ・リカバリーが必要か、またはロールフォワード・リカバリーが完了していないデータベースに対して **db2dart** を実行すると、ディスク上のデータの不整合の性質のため、同様の不整合が発生する場合があります。

INSPECT と db2dart の比較

INSPECT コマンドは、データベースのアーキテクチャーの保全性を検査して、データベース内のページの整合性をチェックします。**INSPECT** コマンドでは、表オブジェクトと表スペースの構造が有効であることをチェックします。オブジェクト間妥当性検査により、オンラインで索引とデータの整合性検査を実行できます。**db2dart** コマンドは、データベースのアーキテクチャーの正確さを検査し、検出されたあらゆるエラーを報告します。

INSPECT コマンドは、**db2dart** コマンドと同じく、データベース、表スペース、表をチェックします。この 2 つのコマンドの大きな違いは、**db2dart** を実行する前にはデータベースを非アクティブ化する必要がありますが、**INSPECT** ではデータベース接続が必要で、データベースへのアクティブ接続が同時に存在する場合でも実行できるという点です。

データベースを非アクティブ化しなかった場合、**db2dart** の実行結果は信頼できません。

db2dart コマンドと **INSPECT** コマンドが実行するテストの違いを以下の表にまとめます。

表 83. 表スペースに対する *db2dart* と *INSPECT* のフィーチャーの比較

実行されるテスト	db2dart	INSPECT
SMS 表スペース		
表スペース・ファイルの検査	YES	NO
内部ページ・ヘッダー・フィールドの内容の妥当性検査	YES	YES
DMS 表スペース		
複数のオブジェクトが指し示すエクステンツ・マップの検査	YES	NO
すべてのエクステンツ・マップ・ページでの整合性ビット・エラーの検査	NO	YES
すべてのスペース・マップ・ページでの整合性ビット・エラーの検査	NO	YES
内部ページ・ヘッダー・フィールドの内容の妥当性検査	YES	YES
エクステンツ・マップが表スペース・マップと一致することの検査	YES	NO

表 84. データ・オブジェクトに対する *db2dart* と *INSPECT* のフィーチャーの比較

実行されるテスト	db2dart	INSPECT
データ・オブジェクトでの整合性ビット・エラーの検査	YES	YES
特殊制御行の内容の検査	YES	NO
可変長列の長さや位置の検査	YES	NO
表の行での LONG VARCHAR、LONG VARGRAPHIC、およびラージ・オブジェクト (LOB) の記述子の検査	YES	NO

表 84. データ・オブジェクトに対する db2dart と INSPECT のフィーチャーの比較 (続き)

実行されるテスト	db2dart	INSPECT
サマリー合計ページ、使用済みページ、およびフリー・スペースの割合の検査	NO	YES
内部ページ・ヘッダー・フィールドの内容の妥当性検査	YES	YES
各行レコード・タイプとその長さの検査	YES	YES
行がオーバーラップしていないことの検査	YES	YES

表 85. 索引オブジェクトに対する db2dart と INSPECT のフィーチャーの比較

実行されるテスト	db2dart	INSPECT
整合性ビット・エラーの検査	YES	YES
索引キーの位置と長さ、およびオーバーラップがあるかどうかの検査	YES	YES
索引におけるキーの順序付けの検査	YES	NO
サマリー合計ページと使用済みページの判別	NO	YES
内部ページ・ヘッダー・フィールドの内容の妥当性検査	YES	YES
ユニーク・キーの一意性の検査	YES	NO
指定された索引項目のデータ行が存在するかどうかの検査	NO	YES
データ値に照らした各キーの検査	NO	YES

表 86. ブロック・マップ・オブジェクトに対する db2dart と INSPECT のフィーチャーの比較

実行されるテスト	db2dart	INSPECT
整合性ビット・エラーの検査	YES	YES
サマリー合計ページと使用済みページの判別	NO	YES
内部ページ・ヘッダー・フィールドの内容の妥当性検査	YES	YES

表 87. 長いフィールドおよび LOB オブジェクトに対する db2dart と INSPECT のフィーチャーの比較

実行されるテスト	db2dart	INSPECT
割り振り構造の検査	YES	YES

表 87. 長いフィールドおよび LOB オブジェクトに対する db2dart と INSPECT のフィーチャーの比較 (続き)

実行されるテスト	db2dart	INSPECT
サマリー合計ページと使用済みページの判別 (LOB オブジェクトのみ)	NO	YES

さらに、**db2dart** コマンドを使用すると、以下のアクションを実行できます。

- データ・ページのフォーマット設定とダンプ
- 索引ページのフォーマット設定とダンプ
- 区切り付き ASCII でのデータ行のフォーマット設定
- 索引に無効のマークを付ける

INSPECT コマンドを使用して、これらのアクションを実行することはできません。

db2diag ツールを使用した db2diag ログ・ファイルの分析

データベース管理者とシステム管理者が使用するための 1 次ログ・ファイルは管理通知ログです。**db2diag** ログ・ファイルは、トラブルシューティングの目的で IBM ソフトウェア・サポートが使用するためのものです。

管理通知ログ・メッセージも、標準化されたメッセージ・フォーマットを使用して **db2diag** ログ・ファイルに記録されます。

db2diag ツールによって、**db2diag** ログ・ファイルで使用可能な大量の情報をフィルターに掛けたり、フォーマットすることが可能です。**db2diag** ログ・ファイルのレコードをフィルター処理することで、問題のトラブルシューティングに必要なレコードを見つけるための時間を短縮できます。

例 1: データベース名による db2diag ログ・ファイルのフィルター処理

インスタンス内にデータベースがいくつかあり、データベース SAMPLE に関連するメッセージだけを参照する場合には、以下のように **db2diag** ログ・ファイルをフィルターに掛けることができます。

```
db2diag -g db=SAMPLE
```

そうすると、以下のように、"DB: SAMPLE" を含んだ **db2diag** ログ・ファイルのレコードだけが表示されます。

```
2006-02-15-19.31.36.114000-300 E21432H406          LEVEL: Error
PID      : 940                TID   : 660          PROC  : db2syscs.exe
INSTANCE: DB2                NODE  : 000          DB    : SAMPLE
APPHDL  : 0-1056             APPID : *LOCAL.DB2.060216003103
FUNCTION: DB2 UDB, base sys utilities, sqliteDatabaseQuiesce, probe:2
MESSAGE : ADM7507W Database quiesce request has completed successfully.
```

例 2: プロセス ID による db2diag ログ・ファイルのフィルター処理

以下のコマンドを使用して、パーティション 0、1、2、または 3 で実行している、プロセス ID (PID) 2200 のプロセスによって生成されるすべての重大エラー・メッセージを表示できます。

```
db2diag -g level=Severe,pid=2200 -n 0,1,2,3
```

このコマンドは 2 種類の異なる方法で書けることに注意してください。すなわち、**db2diag -l severe -pid 2200 -n 0,1,2,3** と書くこともできます。-g オプションは大/小文字の区別のある検索を指定するので、この場合は "Severe" なら問題ありませんが、"severe" が使用されていると失敗することにも注意してください。これらのコマンドは、以下のような、これらの要件にかなう db2diag ログ・ファイルのレコードを正常に検索します。

```
2006-02-13-14.34.36.027000-300 I18366H421          LEVEL: Severe
PID      : 2200                TID : 660          PROC : db2syscs.exe
INSTANCE: DB2                 NODE : 000         DB   : SAMPLE
APPHDL   : 0-1433             APPID: *LOCAL.DB2.060213193043
FUNCTION: DB2 UDB, data management, sqlPoolCreate, probe:273
RETCODE  : ZRC=0x8002003C=-2147352516=SQLB_BAD_CONTAINER_PATH
          "Bad container path"
```

例 3: db2diag ツール出力のフォーマット

以下のコマンドでは、2006 年 1 月 1 日より後に生じた、非重大エラーと重大エラーを含む、パーティション 0、1、または 2 に関して記録されたレコードすべてをフィルタリングします。これによって適合するレコードが出力されます。最初の行にはタイム・スタンプ、パーティション番号、およびレベルが表示され、2 番目の行には pid、tid、およびインスタンス名が表示され、その後エラー・メッセージが表示されます。

```
db2diag -time 2006-01-01 -node "0,1,2" -level "Severe, Error" | db2diag -fmt
"Time: %{ts}
Partition: %node Message Level: %{level} %nPid: %{pid} Tid: %{tid}
Instance: %{instance}%nMessage: @{msg}%n"
```

以下に、作成される出力の例を挙げます。

```
Time: 2006-02-15-19.31.36.099000 Partition: 000 Message Level: Error
Pid: 940 Tid:940 Instance: DB2
Message: ADM7506W Database quiesce has been requested.
```

詳細については、以下のコマンドを発行してください。

- **db2diag -help** は、すべての使用可能なオプションの簡略説明を示します。
- **db2diag -h brief** は、すべてのオプションの記述を示します (例は表示しません)。
- **db2diag -h notes** は、使用上の注意および制約事項を示します。
- **db2diag -h examples** は、始めに参照できるいくつかの例を示します。
- **db2diag -h tutorial** は、使用可能なすべてのオプションの例を示します。
- **db2diag -h all** は、オプションの最も完全なリストを示します。

例 4: 各種機能から送られてくるメッセージのフィルター処理

以下の例では、データベース・マネージャー内の特定の機能から送られてくるメッセージだけ（またはすべての機能から送られてくるメッセージ）を表示する方法を示します。サポートされている機能は、以下のとおりです。

- ALL (すべての機能からのレコードを返します)
- MAIN (**db2diag** ログ・ファイルなどの DB2 一般診断ログや管理通知ログからのレコードを返します)
- OPTSTATS (オプティマイザー統計に関連したレコードを返します)

MAIN 機能から送られてくるメッセージを読み取るには、以下のようになります。

```
db2diag -facility MAIN
```

OPTSTATS 機能から送られてくるメッセージを表示し、フィルターによって Severe レベルのレコードに絞り込むには、以下のようになります。

```
db2diag -fac OPTSTATS -level Severe
```

使用可能なすべての機能から送られてくるメッセージを表示し、フィルターによって instance=harmistr と level=Error のレコードに絞り込むには、以下のようになります。

```
db2diag -fac all -g instance=harmistr,level=Error
```

OPTSTATS 機能から送られてくる Error レベルのメッセージをすべて表示してから、Timestamp と PID フィールドを特定の形式で出力するには、以下のようになります。

```
db2diag -fac optstats -level Error -fmt " Time :%{ts} Pid :%{pid}"
```

例 5: ファイルのマージおよびタイム・スタンプによるレコードのソート

この例では、複数の **db2diag** ログ・ファイルをマージし、タイム・スタンプに従ってレコードをソートする方法を示します。

マージする 2 つの **db2diag** ログ・ファイルは、次のとおりです。

- db2diag.0.log。これには、以下のタイム・スタンプを持つ Level:Error のレコードが含まれています。
 - 2009-02-26-05.28.49.822637
 - 2009-02-26-05.28.49.835733
 - 2009-02-26-05.28.50.258887
 - 2009-02-26-05.28.50.259685
- db2diag.1.log。これには、以下のタイム・スタンプを持つ Level:Error のレコードが含まれています。
 - 2009-02-26-05.28.11.480542
 - 2009-02-26-05.28.49.764762
 - 2009-02-26-05.29.11.872184
 - 2009-02-26-05.29.11.872968

2 つの診断ログ・ファイルをマージし、タイム・スタンプに従ってレコードをソートするには、以下のコマンドを実行します。

```
db2diag -merge db2diag.0.log db2diag.1.log -fmt %{ts} -level error
```

レコードのマージおよびソートの結果は、次のとおりです。

- 2009-02-26-05.28.11.480542
- 2009-02-26-05.28.49.764762
- 2009-02-26-05.28.49.822637
- 2009-02-26-05.28.49.835733
- 2009-02-26-05.28.50.258887
- 2009-02-26-05.28.50.259685
- 2009-02-26-05.29.11.872184
- 2009-02-26-05.29.11.872968

タイム・スタンプがマージされ、時系列的にソートされます。

例 6: 単一ホスト上の診断ディレクトリー・パスにあるファイルのマージとタイム・スタンプによるレコードのソート

デフォルトでは、メンバーと CF はそれぞれ異なる **db2diag** ログ・ファイルに記録します。以下は、マージする 3 つの **db2diag** ログ・ファイルのリストです。

- ~/sql1lib/db2dump/DIAG0000/db2diag.log
- ~/sql1lib/db2dump/DIAG0001/db2diag.log
- ~/sql1lib/db2dump/DIAG0002/db2diag.log

3 つの診断ログ・ファイルをマージし、タイム・スタンプに従ってレコードをソートするには、以下のコマンドを実行します。

```
db2diag -merge
```

例 7: 複数のホストおよびデータベース・パーティション上の診断ディレクトリー・パスにあるファイルのマージ

この例では、すべての診断ログから全レコードの出力を取得し、**bower** と **horton** という 2 つのホストのそれぞれに 3 つあるデータベース・パーティション上の診断ログ・ファイルをマージする方法を示します。以下は、6 つの **db2diag** ログ・ファイルのリストです。

- ~/sql1lib/db2dump/HOST_bower/DIAG0000/db2diag.log
- ~/sql1lib/db2dump/HOST_bower/DIAG0001/db2diag.log
- ~/sql1lib/db2dump/HOST_bower/DIAG0002/db2diag.log
- ~/sql1lib/db2dump/HOST_horton/DIAG0003/db2diag.log
- ~/sql1lib/db2dump/HOST_horton/DIAG0004/db2diag.log
- ~/sql1lib/db2dump/HOST_horton/DIAG0005/db2diag.log

6 つの **db2diag** ログ・ファイルすべてのレコードを出力するには、次のコマンドを実行します。

```
db2diag -global
```

ホストの `bower` と `horton` のそれぞれに 3 つあるデータベース・パーティションすべての診断データ・ディレクトリー・パスの全部で 6 つの `db2diag` ログ・ファイルをマージし、タイム・スタンプに基づいて出力をフォーマット設定するには、次のコマンドを実行します。

```
db2diag -global -merge -sdir /temp/keon -fmt %{ts}
```

`/temp/keon` は、ホストの `bower` と `horton` が共有する共有ディレクトリーであり、処理中に各ホストからのマージ済み一時ファイルを保管するために使用されます。

例 8: 最新の診断ログ項目のみのフィルター処理およびマージ

この例では、`db2diag` ログ・ファイル・レコードがフィルターに掛けられ、特定の数の最新項目のみ表示されます。パーティション・データベース環境の 3 つのパーティションごとに最新の 5 つのレコードを、マージタイム・スタンプでフォーマットして表示するには、以下のように入力します。

```
db2diag -lastrecords 5 -global -merge -sdir /home/vbmithun -fmt %{ts}
```

```
2010-10-08-04.46.02.092192
2010-10-08-04.46.02.092821
2010-10-08-04.46.02.093497
2010-10-08-04.46.02.094431
2010-10-08-04.46.02.095317
2010-10-08-04.46.05.068648
2010-10-08-04.46.05.069212
2010-10-08-04.46.05.069900
2010-10-08-04.46.05.071008
2010-10-08-04.46.05.071831
2010-10-08-04.46.07.302051
2010-10-08-04.46.07.302727
2010-10-08-04.46.07.303544
2010-10-08-04.46.07.304647
2010-10-08-04.46.07.305391
```

最新の診断ログ・レコードをさらにフィルターに掛け、特定のレベルのメッセージのみ戻すこともできます。例えば、10 個の最新レコードのうち、重大メッセージ・レベルのレコードのみ戻すには、以下のように入力します。

```
$ db2diag db2diag.log -lastrecords 10 -level Severe -fmt %{ts}
```

```
2010-08-11-04.11.33.733807
2010-08-11-04.11.33.735398
```

例 9: インスタンスなしクライアント上での db2diag ログ・ファイルのアーカイブ

IBM Data Server Driver Package および IBM Data Server for ODBC and CLI を使って使用できる `db2diag -archive` (または `-A`) オプションを使用して、インスタンスなしクライアント上で診断ログ・ファイルをアーカイブできます。例えば、

```
$ db2diag -A
db2diag: Moving "/home/usr1/clidriver/db2dump/db2diag.log"
to "/home/usr1/clidriver/db2dump/db2diag.log_2010-09-14-01.16.26"
```

`-archive` または `-A` 以外のオプションを指定すると、エラー・メッセージが返されます。例えば、

```
$ db2diag -x
db2diag: Unrecognized option: -x

$ db2diag -pid 1234
db2diag: Unrecognized option: -pid
```

db2greg を使用したグローバル・レジストリーの表示および変更 (UNIX)

グローバル・レジストリーは、UNIX および Linux プラットフォームで **db2greg** コマンドを使用して表示できます。

DB2 バージョン 9.7 以降では、DB2 グローバル・プロファイル・レジストリーは、テキスト・ファイル <DB2DIR>/default.env に記録されません。現在の DB2 インストールに関する DB2 グローバル・プロファイル設定を登録するために、現在、グローバル・レジストリー・ファイル `global.reg` が使用されています。

グローバル・レジストリーは、UNIX および Linux プラットフォームにのみ存在します。

- root インストールの場合、グローバル・レジストリー・ファイルは `/var/db2/global.reg` (HP-UX では `/var/opt/db2/global.reg`) にあります。
- 非 root インストールの場合、グローバル・レジストリー・ファイルは、`$HOME/sql1lib/global.reg` にあります (`$HOME` は非 root ユーザーのホーム・ディレクトリーです)。

グローバル・レジストリーは、以下の 3 つの異なるレコード・タイプから成っています。

- "Service": Service レコードには、製品レベルの情報が含まれています。例えば、バージョンやインストール・パスなどです。
- "Instance": Instance レコードには、インスタンス・レベルの情報が含まれています。例えば、インスタンス名、インスタンス・パス、バージョン、`start-at-boot` フラグなどです。
- "Variable": Variable レコードには、変数レベルの情報が含まれています。例えば、変数名、変数値などです。
- コメント。

グローバル・レジストリーは **db2greg** ツールを使用して表示できます。このツールは `sql1lib/bin` にあります。また、`install` ディレクトリーの `bin` の下にもあります (root としてログインしたときに使用するため)。

グローバル・レジストリーは、**db2greg** ツールを使用して編集できます。root インストールでのグローバル・レジストリーの編集には、root 権限が必要です。

指示されているような特殊な場合、または IBM ソフトウェア・サポートから要請された場合のみ、**db2greg** を使用してください。このコマンドを不適切に使用すると、グローバル・レジストリーが損傷する可能性があります。

製品のバージョンとサービス・レベルの識別

db2level コマンドを使用すれば、ご使用の DB2 インスタンスのバージョンとサービス・レベル (ビルド・レベルおよびフィックスパック番号) を判別できます。

DB2 インスタンスが最新のサービス・レベルであるかどうかを判別するには、db2level の出力結果と、DB2 サポートの Web サイト (www.ibm.com/support/docview.wss?rs=71&uid=swg27007053) のフィックスパック・ダウンロード・ページに表示される情報を比べてください。

db2level コマンドを Windows システムで実行した場合の典型的な結果は次のようなものです。

```
DB21085I Instance "DB2" uses "32" bits and DB2 code release "SQL09010" with
level identifier "01010107".
Informational tokens are "DB2 v9.1.0.189", "n060119", "", and Fix Pack "0".
Product is installed at "c:\SQLLIB" with DB2 Copy Name "db2build".
```

4 つの情報トークンを組み合わせれば、ご使用の DB2 インスタンスの正確なサービス・レベルを固有に識別できます。IBM ソフトウェア・サポートに連絡して支援を受けるには、この情報が必要になります。

JDBC または SQLJ アプリケーションで SQLJ および JDBC 用の IBM DB2 ドライバーを使用している場合には、以下のように db2jcc ユーティリティを実行することによってドライバーのレベルを判別できます。

```
db2jcc -version
```

```
IBM DB2 JDBC Driver Architecture 2.3.63
```

db2look を使用したデータベースの模造

あるデータベースと構造が類似した別のデータベースを作成できると便利な場合がよくあります。例えば、新しいアプリケーションやリカバリー・プランを実動システムでテストするより、ほぼ同じ構造とデータを持つテスト・システムを作成して、代わりにそのテスト・システムでテストする方が理にかなっています。

こうすることにより、実動システムの方は、テストによってパフォーマンス低下の影響を受けたり、エラーがあるアプリケーションによって誤ってデータを破壊されたりする心配がなくなります。また、問題（無効な結果やパフォーマンスの問題など）を調査する際には、実動システムとまったく同じテスト・システムを使った方が問題をデバッグしやすい場合もあります。

db2look ツールを使用すると、あるデータベースのデータベース・オブジェクトを別のデータベースに複製するのに必要な DDL ステートメントを抽出することができます。このツールはまた、あるデータベースから別のデータベースに統計情報を複製するのに必要な SQL ステートメントや、データベース構成、データベース・マネージャー構成、およびレジストリー変数の複製に必要なステートメントを生成することもできます。これは重要なことです。なぜなら、これにより、新しいデータベースに元のデータベースとまったく同じデータ集合が含まれていなくても、2 つのシステムで同じアクセス・プランが選択されるようにすることができるからです。**db2look** コマンドを発行する必要があるのは、DB2 サーバーのバージョン 9.5 以降のレベルが稼働しているデータベースだけです。

db2look ツールの詳細については、「DB2 コマンド・リファレンス」を参照してください。オプションのリストについては、何もパラメーターを指定せずにこのツールを実行すると表示されます。**-h** オプションを使用すると、より詳しい使い方が表示されます。

db2look によるデータベース内の表の模造

データベース内の表の DDL を抽出するには、**-e** オプションを使用します。例として、SAMPLE2 という SAMPLE データベースのコピーを作成してみます。最初のデータベース内のすべてのオブジェクトが新しいデータベースに作成されるようにします。

```
C:¥>db2 create database sample2
DB20000I The CREATE DATABASE command completed successfully.
C:¥>db2look -d sample -e > sample.ddl
-- USER is:
-- Creating DDL for table(s)
-- Binding package automatically ...
-- Bind is successful
-- Binding package automatically ...
-- Bind is successful
```

注: ユーザー定義のスペース、データベース・パーティション・グループ、およびバッファ・プールの DDL も作成する場合には、前出のコマンドの **-e** の後に **-l** フラグを追加します。デフォルトのデータベース・パーティション・グループ、バッファ・プール、および表スペースは抽出されません。なぜなら、それらはすでにデフォルトで、すべてのデータベースに存在しているからです。これらを模造する場合には、それらを手動で変更する必要があります。

sample.ddl ファイルをテキスト・エディターで開きます。このファイルの DDL を新しいデータベースに対して実行するには、**CONNECT TO SAMPLE** ステートメントを **CONNECT TO SAMPLE2** に変更する必要があります。 **-l** オプションを使用した場合には、表スペース・コマンドに関連したパスも、適切なパスを指すように変更する必要があるかもしれません。この作業の間に、ファイルの内容の残りの部分を見てください。サンプル・データベース内のすべてのユーザー表について、**CREATE TABLE**、**ALTER TABLE**、および **CREATE INDEX** ステートメントがあるはずです。

```
...
-----
-- DDL Statements for table "DB2"."ORG"
-----

CREATE TABLE "DB2"."ORG" (
  "DEPTNUMB" SMALLINT NOT NULL ,
  "DEPTNAME" VARCHAR(14) ,
  "MANAGER" SMALLINT ,
  "DIVISION" VARCHAR(10) ,
  "LOCATION" VARCHAR(13) )
IN "USERSPACE1" ;
...
```

接続ステートメントを変更したら、以下のように、そのステートメントを実行します。

```
C:¥>db2 -tvf sample.ddl > sample2.out
```

出力ファイル `sample2.out` で、すべてが正しく実行されていることを確認します。エラーが発生した場合は、エラー・メッセージで問題を確認できます。それらの問題を修正したら、もう一度ステートメントを実行します。

出力を見るとわかるように、すべてのユーザー表の DDL がエクスポートされています。これはデフォルトの動作ですが、操作に含める表を細かく指定するためのオプションもあります。例えば、STAFF 表と ORG 表のみを含めるには、以下のように **-t** オプションを使用します。

```
C:¥>db2look -d sample -e -t staff org > staff_org.ddl
```

スキーマ DB2 を持つ表だけを含めるには、以下のように **-z** オプションを使用します。

```
C:¥>db2look -d sample -e -z db2 > db2.ddl
```

表の統計の模造

テスト・データベースの目的がパフォーマンスのテストやパフォーマンスの問題のデバッグにある場合は、両方のデータベースで同一のアクセス・プランが生成されるようにすることが重要です。最適化プログラムは、統計、構成パラメーター、レジストリー変数、および環境変数に基づいてアクセス・プランを生成します。これらが 2 つのシステムで同一であれば、同じアクセス・プランが生成されると考えられます。

両方のデータベースにまったく同じデータをロードし、同じオプションを指定して `runstats` を実行すれば、統計は同じになるはずですが、しかし、データベースに格納されているデータが異なっていたり、データの一部しかテスト・データベースで使われていなかったりすると、まったく異なる統計になります。このような場合は、**db2look** を使用すると、実動データベースから統計情報を収集して、それをテスト・データベースに追加することができます。そのためには、更新可能なカタログ表の `SYSSTAT` セットに対する `UPDATE` ステートメントと、すべての表に対する `RUNSTATS` コマンドを作成します。

統計ステートメントを作成するためのオプションは **-m** です。SAMPLE/SAMPLE2 の例に戻って、SAMPLE からの統計を収集し、それらを SAMPLE2 に追加します。

```
C:¥>db2look -d sample -m > stats.dml
-- USER is:
-- Running db2look in mimic mode
```

前と同様に、出力ファイルを編集して、`CONNECT TO SAMPLE` ステートメントを `CONNECT TO SAMPLE2` に変更する必要があります。ここでまたファイルの残りの部分を見て、`RUNSTATS` ステートメントや `UPDATE` ステートメントのいくつかに何が含まれているのかを確認します。

```
...
-- Mimic table ORG
RUNSTATS ON TABLE "DB2"."ORG" ;
```

```
UPDATE SYSSTAT.INDEXES
SET NLEAF=-1,
    NLEVELS=-1,
    FIRSTKEYCARD=-1,
    FIRST2KEYCARD=-1,
    FIRST3KEYCARD=-1,
    FIRST4KEYCARD=-1,
    FULLKEYCARD=-1,
    CLUSTERFACTOR=-1,
    CLUSTERRATIO=-1,
    SEQUENTIAL_PAGES=-1,
```

```

PAGE_FETCH_PAIRS='',
DENSITY=-1,
AVERAGE_SEQUENCE_GAP=-1,
AVERAGE_SEQUENCE_FETCH_GAP=-1,
AVERAGE_SEQUENCE_PAGES=-1,
AVERAGE_SEQUENCE_FETCH_PAGES=-1,
AVERAGE_RANDOM_PAGES=-1,
AVERAGE_RANDOM_FETCH_PAGES=-1,
NUMRIDS=-1,
NUMRIDS_DELETED=-1,
NUM_EMPTY_LEAFS=-1
WHERE TABNAME = 'ORG' AND TABSCHEMA = 'DB2  ';
...

```

DDL を抽出する **-e** オプションの場合と同様、**-t** および **-z** オプションを使用して、表集合を指定できます。

構成パラメーターと環境変数の抽出

最適化プログラムは、統計、構成パラメーター、レジストリー変数、および環境変数に基づいてプランを選択します。統計の場合と同様に、必要な構成更新ステートメントや構成設定ステートメントも、**db2look** を使って生成できます。そのためには、**-f** オプションを使用します。例えば、

```

c:¥>db2look -d sample -f>config.txt
-- USER is: DB2INST1
-- Binding package automatically ...
-- Bind is successful
-- Binding package automatically ...
-- Bind is successful

```

config.txt には、以下の例のような出力が含まれます。

```

-- This CLP file was created using DB2LOOK Version 9.1
-- Timestamp: 2/16/2006 7:15:17 PM
-- Database Name: SAMPLE
-- Database Manager Version: DB2/NT Version 9.1.0
-- Database Codepage: 1252
-- Database Collating Sequence is: UNIQUE

CONNECT TO SAMPLE;

-----
-- Database and Database Manager configuration parameters
-----

UPDATE DBM CFG USING cpuspeed 2.991513e-007;
UPDATE DBM CFG USING intra_parallel NO;
UPDATE DBM CFG USING comm_bandwidth 100.000000;
UPDATE DBM CFG USING federated NO;

...

-----
-- Environment Variables settings
-----

COMMIT WORK;

CONNECT RESET;

```

注: DB2 コンパイラーに影響を与えるパラメーターおよび変数だけが組み込まれます。コンパイラーに影響を与えるレジストリー変数がデフォルト値に設定されている場合、それは "Environment Variables settings" の下に表示されません。

システムにインストールされている DB2 製品のリスト表示 (Linux および UNIX)

サポートされている Linux および UNIX オペレーティング・システムでは、**db21s** コマンドを実行すると、システムにインストールされている DB2 データベース製品とフィーチャー (DB2 バージョン 10.1 HTML 文書を含む) がリスト表示されます。

始める前に

db21s コマンドへのシンボリック・リンクを `/usr/local/bin` ディレクトリー内で使用できるようにするために、少なくとも 1 つの DB2 バージョン 9 (またはそれ以降) データベース製品が `root` ユーザーによって既にインストールされている必要があります。

このタスクについて

DB2 データベース製品では複数のコピーをシステムにインストールしたり、DB2 データベース製品およびフィーチャーのインストール先パスを自由に選択したりするためには、何がどこにインストールされているかを把握しておくためのツールが必要となります。サポートされている Linux および UNIX オペレーティング・システムでは、**db21s** コマンドを実行すると、システムにインストールされている DB2 製品とフィーチャー (DB2 HTML 文書を含む) がリスト表示されます。

db21s コマンドは、インストール・メディアとシステム上の DB2 インストール・コピーの両方にあります。**db21s** コマンドはどちらの場所からも実行できます。**db21s** コマンドは、IBM Data Server Driver Package を除くすべての製品のインストール・メディアから実行できます。

db21s コマンドを使用して、以下の内容をリストに表示できます。

- DB2 データベース製品がインストールされているシステム内の場所、および DB2 データベース製品レベル
- 特定のインストール・パスにあるすべての、または特定の DB2 データベース製品およびフィーチャー

制約事項

db21s コマンドがリスト表示する出力は、使用する ID によって異なります。

- **db21s** コマンドを `root` 権限で実行する場合、`root` DB2 インストールだけが照会されます。
- **db21s** コマンドを非 `root` ID で実行する場合、`root` DB2 インストール、および一致する非 `root` ID が所有する非 `root` インストールが照会されます。その他の非 `root` ID が所有する DB2 インストールは照会されません。

DB2 データベース製品を照会する方法は、**db21s** コマンドしかありません。Linux または UNIX オペレーティング・システム固有のユーティリティー (**pkginfo**、

`rpm`、`SMIT`、または `swlist` など) を使用して DB2 データベース製品を照会することはできません。DB2 インストール環境との照会およびインターフェースに使用する既存のスクリプトで、固有のインストール・ユーティリティーを含むものは、変更する必要があります。

Windowsオペレーティング・システムで `db21s` コマンドを使用することはできません。

手順

- DB2 データベース製品がインストールされているシステム内のパス、および DB2 データベース製品レベルをリスト表示するには、次を実行します。

```
db21s
```

コマンドを実行すると、システムにインストールされている DB2 データベース製品ごとに次の情報がリスト表示されます。

- インストール・パス
 - レベル
 - フィックスパック
 - 特別なインストール番号。この列は、IBM DB2 サポートが使用します。
 - インストール日付。この列には、DB2 データベース製品の最終変更日時が表示されます。
 - インストーラー UID。この列には、DB2 データベース製品をインストールした UID が表示されます。
- 特定のインストール・パス内にある DB2 データベース製品またはフィーチャーに関する情報をリストするには、`q` パラメーターを指定する必要があります。

```
db21s -q -p -b baseInstallDirectory
```

説明:

- `q` は、製品またはフィーチャーを照会することを指定します。このパラメーターは必須です。
- `p` は、リスト表示にフィーチャーのリストではなく製品を表示することを指定します。
- `b` は、製品またはフィーチャーのインストール・ディレクトリーを指定します。インストール・ディレクトリーからコマンドを実行しない場合は、このパラメーターは必須です。

タスクの結果

指定するパラメーターに応じて、コマンドは以下の情報をリストします。

- インストール・パス。これは一度だけ指定され、フィーチャーごとにはリストされません。
- 以下の情報が表示されます。
 - インストール済みのフィーチャーの応答ファイル ID、または `p` オプションが指定されている場合はインストール済みの製品の応答ファイル ID。例えば、`ENTERPRISE_SERVER_EDITION`。
 - フィーチャー名、または `p` オプションが指定されている場合は製品名。

- 製品のバージョン、リリース、修正レベル、フィックスパック・レベル (VRMF)。例えば、10.1.0.0
- フィックスパック (該当する場合)。例えば、フィックスパック 1 がインストールされている場合は、表示される値は 1 になります。フィックスパック 1a などの暫定フィックスパックの場合も同様です。
- いずれかの製品の VRMF 情報が一致しない場合は、出力リストの末尾に警告メッセージが表示されます。このメッセージは、フィックスパックを適用するよう指示します。

db2pd コマンドを使用したモニターおよびトラブルシューティング

db2pd コマンドは、トラブルシューティングのために使用します。このツールによって、DB2 メモリー・セットからの情報を素早く即時に返すことが可能です。

概説

このツールは、ラッチを獲得したりエンジン・リソースを使用したりせずに情報を収集します。このため、**db2pd** が情報を収集している間に、取得対象の情報内容が変わる可能性があります。つまり、データは完全には正確でないかもしれません (この可能性を考慮に入れる必要があります)。変更メモリー・ポインターが見つかった場合、**db2pd** が終了するのを防ぐために、シグナル・ハンドラーが使われます。その結果、例えば「データ構造が変更されたためコマンドが強制終了されました (Changing data structure forced command termination)」というメッセージが出力に含まれる可能性があります。この点を考慮に入れて使用すれば、このツールはトラブルシューティングに役立ちます。ラッチなしで情報を収集することには 2 つの利点があります (より高速な検索、エンジン・リソースとの競合がない)。

特定の SQLCODE、ZRC コードまたは ECF コードの発生時にデータベース管理システムに関する情報をキャプチャーする場合は **db2pdcfg -catch** コマンドを使用します。エラーをキャッチすると、db2cos (コールアウト・スクリプト) が起動します。任意の **db2pd** コマンド、オペレーティング・システム・コマンド、または問題解決のために必要なその他のコマンドを実行するために、db2cos スクリプトを動的に変更することができます。テンプレート db2cos スクリプト・ファイルは、UNIX および Linux では、sqllib/bin にあります。Windows オペレーティング・システムの場合、db2cos は \$DB2PATH¥bin ディレクトリーにあります。

新規ノードを追加している際は、**db2pd -addnode** コマンド (詳細情報を参照するには oldviewapps および detail パラメーターを指定します) を使用して、ノードを追加しているデータベース・パーティション・サーバーでの操作の進行状況をモニターできます。

現在アクティブである、または何らかの理由で非アクティブになったイベント・モニターのリストが必要な場合は、**db2pd -gfw** コマンドを実行します。このコマンドでは、高速ライター EDU ごとに、イベント・モニターがデータを書き込むターゲットに関する統計および情報も返されます。

例

以下のリストは、トラブルシューティングを円滑にする **db2pd** コマンドの使用例を集めたものです。

- 例 1: ロック待機の診断
- 例 2: 待機中のすべてのロックをキャプチャーするための **-wlocks** パラメーターの使用
- 例 3: ロック所有者およびロック待機者に関する詳細な実行時情報をキャプチャーするための **-apinfo** パラメーターの使用
- 例 4: ロッキング問題を検討するときのコールアウト・スクリプトの使用
- 例 5: アプリケーションと動的 SQL ステートメントのマッピング
- 例 6: メモリー使用量のモニター
- 例 7: どのアプリケーションが表スペースを消費しているかの確認
- 例 8: リカバリーのモニター
- 例 9: トランザクションによって使用されているリソースの量の確認
- 例 10: ログの使用状況のモニター
- 例 11: SYSPLEX リストの表示
- 例 12: スタック・トレースの生成
- 例 13: データベース・パーティションのメモリー統計の表示
- 例 14: 索引再編成の進行状況のモニター
- 例 15: プロセッサ時間の消費別の上位 EDU の表示と EDU スタック情報の表示
- 例 16: エージェント・イベント・メトリックの表示

例 1: ロック待機の診断

db2pd -db *databasename* -locks -transactions -applications -dynamic を実行すると、結果は以下のようになります。

Locks:

Address	TranHdl	Lockname	Type	Mode	Sts	Owner	Dur	HldCnt	Att	ReleaseFlg
0x07800000202E5238	3	00020002000000040000000052	Row	..X	G	3	1	0	0x0000	0x40000000
0x07800000202E4668	2	00020002000000040000000052	Row	..X	W*	2	1	0	0x0000	0x40000000

-db データベース名オプションを使用して指定したデータベースの場合、最初の結果はそのデータベースのロックを示します。この結果は、TranHdl 2 が、TranHdl 3 によって保持されているロックを待機していることがわかります。

Transactions:

Address	AppHandl	[nod-index]	TranHdl	Locks	State	Tflag	Tflag2	FirstIsn	LastIsn	FirstIso	LastIso	LogSpace	SpaceReserved...
0x0780000020251880	11	[000-00011]	2	4	READ	0x00000000	0x00000000	0x000000000000	0x000000000000	0x000000000000	0x000000000000	0	...
0x0780000020252900	12	[000-00012]	3	4	WRITE	0x00000000	0x00000000	0x00000002AC04	0x00000002AC04	0x000000FA000C	0x000000FA000C	113	154

TranHdl 2 が AppHandl 11 に関連付けられていて、TranHdl 3 が AppHandl 12 に関連付けられていることがわかります。

Applications:

Address	AppHandl	[nod-index]	NumAgents	CoorPid	Status	C-AnchID	C-StmtUID	L-AnchID	L-StmtUID	AppId
0x0780000006879E0	12	[000-00012]	1	1073336	UOW-Waiting	0	0	17	1	*LOCAL.burford.060303225602
0x078000000685E80	11	[000-00011]	1	1040570	UOW-Executing	17	1	94	1	*LOCAL.burford.060303225601

AppHandl 12 は、動的ステートメント 17, 1 を最後に実行したのがわかります。AppHandl 11 は、動的ステートメント 17, 1 を現在実行中で、最後に実行ステートメント 94, 1 を実行しました。

Dynamic SQL Statements:

Address	AnchID	StmtUID	NumEnv	NumVar	NumRef	NumExe	Text
0x07800000209FD800	17	1	1	1	2	2	update pdtest set c1 = 5
0x07800000209FCCC0	94	1	1	1	2	2	set lock mode to wait 1

テキスト列は、ロック・タイムアウトに関連している SQL ステートメントを示しています。

例 2: 待機中のすべてのロックをキャプチャーするための **-wlocks** パラメーターの使用

db2pd -wlocks -db pdtest を実行すると、以下のような結果が生成されます。これらの結果は、最初のアプリケーション (AppHandl 47) が表に対して挿入を実行し、2 番目のアプリケーション (AppHandl 46) その表に対して選択を実行していることを示しています。

```
venus@boson:/home/venus =>db2pd -wlocks -db pdtest

Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:22

Locks being waited on :
AppHandl [nod-index] TranHdl Lockname Type Mode Conv Sts CoordEDU AppName AuthID AppID
47 [000-00047] 8 00020004000000000840000652 Row ..X G 5160 db2bp VENUS *LOCAL.venus.071207213730
46 [000-00046] 2 00020004000000000840000652 Row .NS W 5913 db2bp VENUS *LOCAL.venus.071207213658
```

例 3: ロック所有者およびロック待機者に関する詳細な実行時情報をキャプチャーするための **-apinfo** パラメーターの使用

例 2 と同じ条件では、以下のような出力例が生成されました。

```
venus@boson:/home/venus =>db2pd -apinfo 47 -db pdtest

Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:30

Application :
Address : 0x0780000001676480
AppHandl [nod-index] : 47 [000-00047]
Application PID : 876558
Application Node Name : boson
IP Address: n/a
Connection Start Time : (1197063450)Fri Dec 7 16:37:30 2007
Client User ID : venus
System Auth ID : VENUS
Coordinator EDU ID : 5160
Coordinator Partition : 0
Number of Agents : 1
Locks timeout value : 4294967294 seconds
Locks Escalation : No
Workload ID : 1
Workload Occurrence ID : 2
Trusted Context : n/a
Connection Trust Type : non trusted
Role Inherited : n/a
Application Status : UOW-Waiting
Application Name : db2bp
Application ID : *LOCAL.venus.071207213730

ClientUserID : n/a
ClientWrkstnName : n/a
ClientApplName : n/a
ClientAcctng : n/a

List of inactive statements of current UOW :
UOW-ID : 2
Activity ID : 1
Package Schema : NULLID
Package Name : SQLC2G13
Package Version :
Section Number : 203
SQL Type : Dynamic
Isolation : CS
```

```
Statement Type : DML, Insert/Update/Delete
Statement :      insert into pdtest values 99
```

```
venus@boson:/home/venus =>db2pd -apinfo 46 -db pdtest
```

```
Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:39
```

```
Application :
Address :      0x0780000000D77A60
AppHandl [nod-index] : 46      [000-00046]
Application PID : 881102
Application Node Name : boson
IP Address:    n/a
Connection Start Time : (1197063418)Fri Dec 7 16:36:58 2007
Client User ID : venus
System Auth ID : VENUS
Coordinator EDU ID : 5913
Coordinator Partition : 0
Number of Agents : 1
Locks timeout value : 4294967294 seconds
Locks Escalation : No
Workload ID : 1
Workload Occurrence ID : 1
Trusted Context : n/a
Connection Trust Type : non trusted
Role Inherited : n/a
Application Status : Lock-wait
Application Name : db2bp
Application ID : *LOCAL.venus.071207213658

ClientUserID : n/a
ClientWrkstnName : n/a
ClientApplName : n/a
ClientAcctng : n/a
```

```
List of active statements :
*UOW-ID :      3
Activity ID : 1
Package Schema : NULLID
Package Name : SQLC2G13
Package Version :
Section Number : 201
SQL Type :     Dynamic
Isolation :    CS
Statement Type : DML, Select (blockable)
Statement :     select * from pdtest
```

例 4: ロッキング問題を検討するときのコールアウト・スクリプトの使用

コールアウト・スクリプトを使用するには、db2cos 出力ファイルを見つけます。このファイルの場所は、データベース・マネージャ構成パラメーター **diagpath** によって制御されます。出力ファイルの内容は、db2cos スクリプト・ファイルに入力するコマンドによって異なります。以下は、db2cos スクリプト・ファイルに **db2pd -db sample -locks** コマンドが含まれている場合に提供される出力の例です。

```
Lock Timeout Caught
Thu Feb 17 01:40:04 EST 2006
Instance DB2
Database: SAMPLE
Partition Number: 0
PID: 940
TID: 2136
Function: sqlplnfd
Component: lock manager
Probe: 999
```


Timestamp: 2006-02-17-01.40.04.106000

AppID: *LOCAL.DB2...

AppHdl:

...

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:06:53

Locks:

Address	TranHdl	Lockname	Type	Mode	Sts	Owner	Dur	HldCnt	Att	Rlse
0x402C6B30	3	00020003000000040000000052	Row	..X	W*	3	1	0	0	0x40

この出力で、W* はタイムアウトになったロックを示します。この場合、ロック待機が発生しています。ロック・タイムアウトは、ロックがより高いモードに変換されるときにも発生します。このことは、出力内の C* で示されています。

db2cos ファイルにある他の **db2pd** コマンドによって提供される出力を参照しながら、この出力結果をトランザクション、アプリケーション、エージェント、SQL ステートメントにマップすることが可能です。出力の特定部分を絞り込んだり、他のコマンドを使用することによって、必要な情報を収集できます。例えば、**db2pd -locks wait** パラメーターを使用して、待機状況であるロックだけを印刷できます。**-app** および **-agent** パラメーターを使用することもできます。

例 5: アプリケーションと動的 SQL ステートメントのマップ

コマンド **db2pd -applications -dynamic** は、動的 SQL ステートメントの現行および最新アンカー ID と、ステートメント・ユニーク ID を報告します。これにより、アプリケーションから動的 SQL ステートメントへの直接的なマッピングが可能になります。

Applications:

Address	AppHdl	[nod-index]	NumAgents	CoorPid	Status
0x000000022006D2120	780	[000-00780]	1	10615	UOW-Executing

C-AnchID	C-StmtUID	L-AnchID	L-StmtUID	Appid
163	1	110	1	*LOCAL.burford.050202200412

Dynamic SQL Statements:

Address	AnchID	StmtUID	NumEnv	NumVar	NumRef	NumExe	Text
0x0000000220A02760	163	1	2	2	2	1	CREATE VIEW MYVIEW
0x0000000220A0B460	110	1	2	2	2	1	CREATE VIEW YOURVIEW

例 6: メモリー使用量のモニター

以下の出力例のようなメモリー使用量を知りたい場合は **db2pd -memblock** コマンドが役立ちます。

All memory blocks in DBMS set.

Address	PoolID	PoolName	BlockAge	Size(Bytes)	I	LOC	File
0x078000000740068	62	resynch	2	112	1	1746	1583816485
0x078000000725688	62	resynch	1	108864	1	127	1599127346
0x0780000001F4348	57	ostrack	6	5160048	1	3047	698130716
0x0780000001B5608	57	ostrack	5	240048	1	3034	698130716
0x0780000001A0068	57	ostrack	1	80	1	2970	698130716
0x0780000001A00E8	57	ostrack	2	240	1	2983	698130716
0x0780000001A0208	57	ostrack	3	80	1	2999	698130716
0x0780000001A0288	57	ostrack	4	80	1	3009	698130716
0x078000000700068	70	apmh	1	360	1	1024	3878879032
0x0780000007001E8	70	apmh	2	48	1	914	1937674139
0x078000000700248	70	apmh	3	32	1	1000	1937674139
...							

この後に、ソートされた「プールごとの」出力が続きます。

```

Memory blocks sorted by size for ostrack pool:
PoolID   PoolName   TotalSize(Bytes)   TotalCount LOC   File
57       ostrack   5160048             1          3047 698130716
57       ostrack   240048              1          3034 698130716
57       ostrack   240                 1          2983 698130716
57       ostrack   80                  1          2999 698130716
57       ostrack   80                  1          2970 698130716
57       ostrack   80                  1          3009 698130716
Total size for ostrack pool: 5400576 bytes

```

```

Memory blocks sorted by size for apmh pool:
PoolID   PoolName   TotalSize(Bytes)   TotalCount LOC   File
70       apmh      40200              2          121 2986298236
70       apmh      10016              1          308 1586829889
70       apmh      6096               2          4014 1312473490
70       apmh      2516               1          294 1586829889
70       apmh      496                1          2192 1953793439
70       apmh      360                1          1024 3878879032
70       apmh      176                1          1608 1953793439
70       apmh      152                1          2623 1583816485
70       apmh      48                 1          914 1937674139
70       apmh      32                 1          1000 1937674139
Total size for apmh pool: 60092 bytes
...

```

出力の最終セクションでは、全メモリー・セットについて、メモリー・コンシューマーをソートします。

```

All memory consumers in DBMS memory set:
PoolID   PoolName   TotalSize(Bytes)   %Bytes TotalCount %Count LOC   File
57       ostrack   5160048             71.90 1          0.07 3047 698130716
50       sqlch    778496              10.85 1          0.07 202 2576467555
50       sqlch    271784              3.79 1          0.07 260 2576467555
57       ostrack   240048              3.34 1          0.07 3034 698130716
50       sqlch    144464              2.01 1          0.07 217 2576467555
62       resynch  108864              1.52 1          0.07 127 1599127346
72       eduah    108048              1.51 1          0.07 174 4210081592
69       krcbh    73640               1.03 5          0.36 547 4210081592
50       sqlch    43752               0.61 1          0.07 274 2576467555
70       apmh     40200               0.56 2          0.14 121 2986298236
69       krcbh    32992               0.46 1          0.07 838 698130716
50       sqlch    31000               0.43 31         2.20 633 3966224537
50       sqlch    25456               0.35 31         2.20 930 3966224537
52       kerh     15376               0.21 1          0.07 157 1193352763
50       sqlch    14697               0.20 1          0.07 345 2576467555
...

```

UNIX および Linux オペレーティング・システムでは、専用メモリーのメモリー・ブロックを報告することもできます。例えば、**db2pd -memb pid=159770** を実行すると、以下のような結果が生成されます。

```

All memory blocks in Private set.
Address      PoolID   PoolName   BlockAge   Size(Bytes) I LOC   File
0x0000000110469068 88      private   1          2488      1 172 4283993058
0x0000000110469A48 88      private   2          1608      1 172 4283993058
0x000000011046A0A8 88      private   3          4928      1 172 4283993058
0x000000011046B408 88      private   4          7336      1 172 4283993058
0x000000011046D0C8 88      private   5          32        1 172 4283993058
0x000000011046D108 88      private   6          6728      1 172 4283993058
0x000000011046EB68 88      private   7          168       1 172 4283993058
0x000000011046EC28 88      private   8          24        1 172 4283993058
0x000000011046EC68 88      private   9          408       1 172 4283993058
0x000000011046EE28 88      private  10         1072      1 172 4283993058
0x000000011046F288 88      private  11         3464      1 172 4283993058
0x0000000110470028 88      private  12          80       1 172 4283993058
0x00000001104700A8 88      private  13          480      1 1534 862348285
0x00000001104702A8 88      private  14          480      1 1939 862348285
0x0000000110499FA8 88      private  80        65551     1 1779 4231792244
Total set size: 94847 bytes

```

Memory blocks sorted by size:

PoolID	PoolName	TotalSize(Bytes)	TotalCount	LOC	File
88	private	65551	1	1779	4231792244
88	private	28336	12	172	4283993058
88	private	480	1	1939	862348285
88	private	480	1	1534	862348285

Total set size: 94847 bytes

例 7: どのアプリケーションが表スペースを消費しているかの確認

db2pd -tcbstats コマンドを使用すれば、表に対する挿入の数を識別することができます。以下の例は、TEMP1 という名前のユーザー定義グローバル一時表のサンプル情報を示しています。

```
TCB Table Information:
Address      TbspaceID TableID PartID MasterTbs MasterTab TableName SchemaNm ObjClass DataSize LfSize LobSize XMLSize
0x0780000020B62AB0 3      2      n/a      3      2      TEMP1     SESSION  Temp    966      0      0      0

TCB Table Stats:
Address      TableName Scans  UDI  PgReorgs NoChgUpdts Reads  Fscripupdates Inserts  Updates  Deletes  OvFlReads OvFlCrtes
0x0780000020B62AB0 TEMP1      0      0      0      0      0      0      43968   0      0      0      0
```

その後、**db2pd -tablespaces** コマンドを使用することにより、表スペース 3 の情報を入手できます。以下はその出力例です。

```
Tablespace 3 Configuration:
Address      Type Content PageSz ExtentSz Auto Prefetch BufID BufIDDisk FSC NumCntrs MaxStripe LastConsecPg Name
0x0780000020B1B5A0 DMS  UsrTmp 4096 32 Yes 32 1 1 On 1 0 31 TEMPSPACE2

Tablespace 3 Statistics:
Address      TotalPgs UsablePgs UsedPgs PndFreePgs FreePgs HWM State MinRecTime NQuiescers
0x0780000020B1B5A0 5000 4960 1088 0 3872 1088 0x00000000 0 0

Tablespace 3 Autoresize Statistics:
Address      AS AR InitSize IncSize IIP MaxSize LastResize LRF
0x0780000020B1B5A0 No No 0 0 No 0 None No

Containers:
Address      ContainNum Type TotalPgs UseablePgs StripeSet Container
0x0780000020B1DCC0 0 File 5000 4960 0 /home/db2inst1/tempspace2a
```

FreePgs 列は、スペースが埋まってきていることを示しています。フリー・ページの値が減るほど、使用可能なスペースは少なくなります。**FreePgs** の値と **UsedPgs** の値の和が **UsablePgs** の値と等しくなっていることにも注目してください。

それが分かれば、**db2pd -db sample -dyn** を実行することにより、表 TEMP1 を使用している動的 SQL ステートメントを識別できます。

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:13:06

Dynamic Cache:
Current Memory Used      1022197
Total Heap Size          1271398
Cache Overflow Flag      0
Number of References     237
Number of Statement Inserts 32
Number of Statement Deletes 13
Number of Variation Inserts 21
Number of Statements     19

Dynamic SQL Statements:
Address      AnchID StmtUID NumEnv NumVar NumRef NumExe Text
0x00000000220A08C40 78 1 2 2 3 2 declare global temporary table temp1 (c1 char(6)) not logged
0x00000000220A08D960 253 1 1 1 24 24 insert into session.temp1 values('TEST')
```

最後に、**db2pd -db sample -app** を実行することによって上記の出力の情報をアプリケーション出力と付き合わせるにより、アプリケーションを識別することができます。

```
Applications:
Address      AppHandl [nod-index] NumAgents CoorPid Status
0x00000000200661840 501 [000-00501] 1 11246 UOW-Waiting

C-AnchID C-StmtUID L-AnchID L-StmtUID Appid
0 0 253 1 *LOCAL.db2inst1.050202160426
```

動的 SQL ステートメントを識別したアンカー ID (AnchID) 値を使用して、関連付けられているアプリケーションを識別することができます。この結果は、最後のア

アンカー ID (L-AnchID) 値がアンカー ID (AnchID) 値と同じであることを示しています。 **db2pd** の 1 回の実行の結果が次の **db2pd** の実行で使用されます。

db2pd -agent からの出力は、アプリケーションによって読み込まれた行数 (Rowsread 列) と書き込まれた行数 (Rowswrtn 列) を示しています。これらの値によって、以下の出力例のように、アプリケーションが何を完了し、何をまだ完了していないかがわかります。

```
Address          AppHandl [nod-index] AgentPid Priority Type DBName
0x00000000200698080 501      [000-00501] 11246    0       Coord SAMPLE

State           ClientPid Userid   ClientNm Rowsread  Rowswrtn  LkTmOt
Inst-Active    26377    db2inst1 db2bp   22       9588     NotSet
```

db2pd -agent コマンドを実行した結果として得られる AppHandl と AgentPid の値を、**db2pd -app** コマンドを実行した結果として得られる AppHandl と CoordPid の対応値にマップできます。

内部一時表が表スペースを埋めている可能性があれば、これらのステップは若干異なります。ただし、この場合も **db2pd -tcbstats** を使用して、多数の挿入を持つ表を識別します。以下は、暗黙的一時表のサンプル情報です。

```
TCB Table Information:
Address      TbspaceID TableID PartID MasterTbs MasterTab TableName          SchemaNm ObjClass  DataSize ...
0x0780000020CC0D30 1      2      n/a   1      2      TEMP (00001,00002) <30> <JMC Temp 2470 ...
0x0780000020CC14B0 1      3      n/a   1      3      TEMP (00001,00003) <31> <JMC Temp 2367 ...
0x0780000020CC21B0 1      4      n/a   1      4      TEMP (00001,00004) <30> <JMC Temp 1872 ...

TCB Table Stats:
Address      TableName          Scans  UDI  PgReorgs  NoChgUpdpts Reads  FscrUpdates Inserts ...
0x0780000020CC0D30 TEMP (00001,00002) 0      0    0          0          0      43219 ...
0x0780000020CC14B0 TEMP (00001,00003) 0      0    0          0          0      42485 ...
0x0780000020CC21B0 TEMP (00001,00004) 0      0    0          0          0      0 ...
```

この例では、命名規則 TEMP (TbspaceID, TableID) を持つ表に多数の挿入があります。これらは暗黙的一時表です。SchemaNm 列の値には、AppHandl の値が SchemaNm の値と連結するという命名規則があります。これにより、操作を実行しているアプリケーションを識別できます。

その後、その情報を **db2pd -tablespaces** からの出力にマップし、表スペース 1 で使用されるスペースを確認できます。以下の出力内の表スペース統計で、UsedPgs の値と UsablePgs の値の間に注目にしてください。

```
Tablespace Configuration:
Address      Id  Type Content PageSz ExtentSz Auto Prefetch BufID BufIDDisk FSC NumCntrs MaxStripe LastConseqPg Name
0x07800000203FB5A0 1  SMS SysTmp 4096 32 Yes 320 1 1 On 10 0 31 TEMPSPACE1

Tablespace Statistics:
Address      Id  TotalPgs UsablePgs UsedPgs PndFreePgs FreePgs HWM State MinRecTime NQuiescers
0x07800000203FB5A0 1 6516 6516 6516 0 0 0 0x00000000 0 0

Tablespace Autoresize Statistics:
Address      Id  AS AR InitSize IncSize IIP MaxSize LastResize LRF
0x07800000203FB5A0 1 No No 0 0 No 0 None No

Containers:
...
```

その後、コマンド **db2pd -app** を使用して、アプリケーション・ハンドル 30 および 31 を識別することができます (これらは **-tcbstats** 出力で確認されたものです)。

```
Applications:
Address      AppHandl [nod-index] NumAgents CoordPid Status C-AnchID C-StmtUID L-AnchID L-StmtUID Appid
0x07800000006FB880 31 [000-00031] 1 4784182 UOW-Waiting 0 0 107 1 *LOCAL.db2inst1.051215214142
0x07800000006F9CE0 30 [000-00030] 1 8966270 UOW-Executing 107 1 107 1 *LOCAL.db2inst1.051215214013
```

最後に、上記の出力の情報を、**db2pd -dyn** コマンドを実行することによって得られた動的 SQL 出力と付き合わせます。

```
Dynamic SQL Statements:
Address          AnchID StmtUID NumEnv NumVar NumRef NumExe Text
0x0780000020B296C0 107 1 1 1 43 43 select c1, c2 from test group by c1,c2
```

例 8: リカバリーのモニター

コマンド **db2pd -recovery** を実行した場合、出力は、以下の出力例で示されているように、リカバリーの進行状況を確認するために使用できるいくつかのカウンターを示します。「Current Log (現在のログ)」および「Current LSO (現在の LSO)」の値はログの位置を示します。「CompletedWork」は、完了済みのバイト数です。

```
Recovery:
Recovery Status      0x00000401
Current Log          S0000005.LOG
Current LSN          000001F07BC
Current LSO          000002551BEA
Job Type              ROLLFORWARD RECOVERY
Job ID                7
Job Start Time       (1107380474) Wed Feb  2 16:41:14 2005
Job Description       Database Rollforward Recovery
Invoker Type          User
Total Phases         2
Current Phase        1

Progress:
Address          PhaseNum Description StartTime          CompletedWork TotalWork
0x0000000200667160 1 Forward Wed Feb  2 16:41:14 2005 2268098 bytes Unknown
0x0000000200667258 2 Backward NotStarted 0 bytes Unknown
```

例 9: トランザクションによって使用されているリソースの量の確認

コマンド **db2pd -transactions** を実行した場合、以下の出力例のように、ロックの数、最初のログ・シーケンス番号 (LSN)、最後の LSN、最初の LSO、最後の LSO、使用されているログ・スペース、および予約済みスペースが出力に示されます。この情報はトランザクションの動作を理解するうえで役立ちます。

```
Transactions:
Address          AppHandl [nod-index] TranHdl Locks State Tflag
0x000000022026D980 797 [000-00797] 2 108 WRITE 0x00000000
0x000000022026E600 806 [000-00806] 3 157 WRITE 0x00000000
0x000000022026F280 807 [000-00807] 4 90 WRITE 0x00000000

Tflag2          Firstlsn Lastlsn Firstlso Lastlso
0x00000000 0x0000001A4212 0x0000001C2022 0x000001072262 0x0000010B2C8C
0x00000000 0x000000107320 0x0000001S3462 0x000001057574 0x0000010B3340
0x00000000 0x0000001BC00C 0x0000001X2F03 0x00000107CF0C 0x0000010B2FDE
LogSpace SpaceReserved TID AxRegCnt GXID
4518 95450 0x000000000451 1 0
6576 139670 0x0000000003E0 1 0
3762 79266 0x000000000472 1 0
```

例 10: ログの使用状況のモニター

db2pd -logs コマンドは、データベースのログ使用状況をモニターするのに役立ちます。「Pages Written (書き込み済みページ数)」値を使用することにより、以下の出力例で示されているように、ログの使用量が増加しているかどうかを判別できます。

```
Logs:
Current Log Number      2
Pages Written           846
Method 1 Archive Status Success
Method 1 Next Log to Archive 2
Method 1 First Failure n/a
```

```

Method 2 Archive Status      Success
Method 2 Next Log to Archive 2
Method 2 First Failure      n/a

Address      StartLSN      StartLSO      State      Size  Pages  Filename
0x000000023001BF58 0x00000022F032 0x000001B58000 0x00000000 1000 1000  S0000002.LOG
0x000000023001BE98 0x000000000000 0x000001F40000 0x00000000 1000 1000  S0000003.LOG
0x0000000230008F58 0x000000000000 0x000002328000 0x00000000 1000 1000  S0000004.LOG

```

この出力を使用することにより、2 種類の問題を識別できます。

- 最近のログ・アーカイブが失敗した場合、「Archive Status (アーカイブ状況)」は値「Failure (失敗)」に設定されています。アーカイブ失敗が継続しているためにログがアーカイブされない場合、「Archive Status (アーカイブ状況)」は値「First Failure (最初に失敗)」に設定されます。
- ログのアーカイブ処理が非常に遅い場合には、「Next Log to Archive (次のアーカイブ対象ログ)」の値が「Current Log Number (現在のログ番号)」の値より小さくなっています。アーカイブ処理が非常に遅い場合には、アクティブ・ログ用のスペースがなくなり、データベース内のデータがまったく変更されなくなる可能性があります。

注: S0000003.LOG および S0000004.LOG にはログ・レコードがまだ含まれていないため、StartLSN は 0x0 です。

例 11: SYSPLEX リストの表示

以下の出力例を示す **db2pd -sysplex** コマンドを使用しない場合、SYSPLEX リストを報告する他の唯一の方法は、DB2 トレースを介する方法です。

```

Sysplex List:
Alias:          HOST
Location Name: HOST1
Count:         1

IP Address      Port      Priority  Connections Status  PRDID
1.2.34.56      400      1        0          0

```

例 12: スタック・トレースの生成

Windows オペレーティング・システムの場合は **db2pd -stack all** コマンド (UNIX オペレーティング・システムの場合は **-stack** コマンド) を使用すれば、現在のデータベース・パーティションにあるすべてのプロセスのスタック・トレースを生成できます。プロセスやスレッドがループ状態または停止状態にあると疑われる場合には、このコマンドを反復して使用できます。

db2pd -stack eduid コマンドを実行して、以下の例のように特定のエンジン・ディスプレイパッチ可能単位 (EDU) の現在の呼び出しスタックを取得できます。

```

Attempting to dump stack trace for eduid 137.
See current DIAGPATH for trapfile.

```

DB2 プロセスに対するすべての呼び出しスタックを確認するには、**db2pd -stack all** コマンドを使用します。例えば、Windows オペレーティング・システムであれば、以下のようにします。

```

Attempting to dump all stack traces for instance.
See current DIAGPATH for trapfiles.

```

複数の物理ノードのあるパーティション・データベース環境では、コマンド **db2_all "; db2pd -stack all"** を使用することにより、すべてのパーティションか

ら情報を取得できます。しかし、同じマシン上の複数の論理パーティションだけから成る環境では、**db2pd -alldbp -stacks** を使用した方が速く動作します。

dumpdir パラメーターを指定して **db2pdb -stacks** コマンドの出力を特定のディレクトリー・パスにリダイレクトしたり、**timeout** パラメーターを指定して特定の期間の出力だけをリダイレクトしたりすることもできます。例えば、すべてのプロセスに関するスタック・トレースの出力を /home/waleed/mydir に 30 秒間リダイレクトするには、以下のコマンドを発行してください。

```
db2pd -alldbp -stack all dumpdir=/home/waleed/mydir timeout=30
```

例 13: データベース・パーティションのメモリー統計の表示

db2pd -dbptnmem コマンドは、DB2 サーバーが現在使用しているメモリーの量を表示します。また、サーバーのどの領域がそのメモリーを使用しているかの概要を表示します。

AIX マシン上での **db2pd -dbptnmem** の実行による出力例を以下の例に示します。

Database Partition Memory Controller Statistics

```
Controller Automatic: Y
Memory Limit:      122931408 KB
Current usage:     651008 KB
HWM usage:        651008 KB
Cached memory:    231296 KB
```

これらのデータ・フィールドおよび列に関する説明を以下に示します。

Controller Automatic (自動コントローラー)

メモリー・コントローラーの設定を示します。**instance_memory** 構成パラメーターが **AUTOMATIC** に設定されている場合、値 **Y** を示します。これは、データベース・マネージャーが自動的にメモリー使用量の上限を決定することを意味します。

Memory Limit (メモリーの限度)

インスタンスのメモリー限度が強制される場合、**instance_memory** 構成パラメーターの値は、消費できる DB2 サーバー・メモリーの上限です。

Current usage (現在の使用量)

サーバーが現在使用しているメモリーの量。

HWM usage (HWM 使用量)

データベース・パーティションがアクティブ化されたとき (**db2start** コマンドの実行時) 以来消費された、最高水準点 (HWM)、つまりピーク時のメモリー使用量。

Cached memory (キャッシュ済みメモリー)

現在の使用量のうちどの程度の量が、現在使用中でないものの、パフォーマンス上の理由で今後のメモリー要求のためにキャッシュに入れられているか。

AIX オペレーティング・システム上での **db2pd -dbptnmem** の実行による出力例の続きを以下に示します。

```
Individual Memory Consumers:
Name          Mem Used (KB)  HWM Used (KB)  Cached (KB)
=====
```

APPL-DBONE	160000	160000	159616
DBMS-name	38528	38528	3776
FMP_RESOURCES	22528	22528	0
PRIVATE	13120	13120	740
FCM_RESOURCES	10048	10048	0
LCL-p606416	128	128	0
DB-DBONE	406656	406656	67200

DB2 サーバー内のすべての登録済みのメモリーの「コンシューマー」が、使用しているメモリーの合計量とともにリストされます。列の説明は以下のとおりです。

Name (名前)

メモリーの「コンシューマー」の簡潔な識別名。以下のものがあります。

APPL-dbname

データベース *dbname* について消費されるアプリケーション・メモリー。

DBMS-name

グローバル・データベース・マネージャーのメモリー所要量。

FMP_RESOURCES

db2fmps との通信に必要なメモリー。

PRIVATE

各種の専用メモリー所要量。

FCM_RESOURCES

高速コミュニケーション・マネージャーのリソース。

LCL-pid

ローカル・アプリケーションとの通信に使用されるメモリー・セグメント。

DB-dbname

データベース *dbname* について消費されるデータベース・メモリー。

Mem Used (KB) (使用されているメモリー (KB))

コンシューマーに現在割り当てられているメモリーの量。

HWM Used (KB) (使用された HWM (KB))

コンシューマーが使用した最高水準点 (HWM)、あるいはピーク時のメモリー。

Cached (KB) (キャッシュ済み (KB))

「Mem Used (KB) (使用されているメモリー (KB))」のうち、現在使用中ではないものの、今後のメモリー割り振りのためにすぐに使用可能なメモリーの量。

例 14: 索引再編成の進行状況のモニター

DB2 バージョン 9.8 フィックスパック 3 以降、索引再編成の進行状況レポートには、以下の特性があります。

- **db2pd -reorgs index** コマンドを使用すると、パーティション索引の索引 REORG 進行状況がレポートされます (フィックスパック 1 では、非パーティション索引に対するサポートのみが導入されました)。

- **db2pd -reorgs index** コマンドを使用すると、パーティション・レベルでの索引 REORG 操作 (つまり、単一パーティションの再編成) のモニターがサポートされます。
- 非パーティション索引とパーティション索引の REORG 進行状況は別々の出力でレポートされます。1 つの出力では非パーティション索引の REORG 進行状況が表示され、それに続く出力ではそれぞれの表パーティションのパーティション索引に関する REORG 進行状況が示されます。各出力でレポートされるのは、1 つのパーティションだけの索引 REORG 統計です。
- 非パーティション索引が最初に処理され、その後、順次パーティション索引が処理されます。
- **db2pd -reorgs index** コマンドを使用すると、パーティション索引の出力には、以下の追加情報フィールドが表示されます。
 - MaxPartition - 処理されている表のパーティション総数。パーティション・レベルの REORG 操作の場合、MaxPartition の値は常に 1 です。一度に 1 つのパーティションだけが再編成されるからです。
 - PartitionID - 処理中のパーティションのデータ・パーティション ID。

以下の例は、**db2pd -reorgs index** コマンドを使用した出力例を示し、2 つのパーティションがある範囲パーティション表の索引 REORG 進行状況がレポートされています。

注: 最初の出力では、非パーティション索引の索引 REORG 状況がレポートされています。続く出力では、各パーティションにあるパーティション索引の索引 REORG 状況がレポートされています。

```

Index Reorg Stats:
Retrieval Time: 02/08/2010 23:04:21
TbpaceID: -6      TableID: -32768
Schema: ZORAN    TableName: BIGRPT
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:03:55   End Time: 02/08/2010 23:04:04
Total Duration: 00:00:08
Prev Index Duration: -
Cur Index Start: -
Cur Index: 0          Max Index: 2          Index ID: 0
Cur Phase: 0          ( - )      Max Phase: 0
Cur Count: 0          Max Count: 0
Total Row Count: 750000

Retrieval Time: 02/08/2010 23:04:21
TbpaceID: 2      TableID: 5
Schema: ZORAN    TableName: BIGRPT
PartitionID: 0   MaxPartition: 2
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:04:04   End Time: 02/08/2010 23:04:08
Total Duration: 00:00:04
Prev Index Duration: -
Cur Index Start: -
Cur Index: 0          Max Index: 2          Index ID: 0
Cur Phase: 0          ( - )      Max Phase: 0
Cur Count: 0          Max Count: 0
Total Row Count: 375000

Retrieval Time: 02/08/2010 23:04:21
TbpaceID: 2      TableID: 6
Schema: ZORAN    TableName: BIGRPT

```

```

PartitionID: 1      MaxPartition: 2
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:04:08   End Time: 02/08/2010 23:04:12
Total Duration: 00:00:04
Prev Index Duration: -
Cur Index Start: -
Cur Index: 0      Max Index: 2      Index ID: 0
Cur Phase: 0      ( - )      Max Phase: 0
Cur Count: 0      Max Count: 0
Total Row Count: 375000

```

例 15: プロセッサ時間の消費別の上位 EDU の表示と EDU スタック情報の表示

-**edus** パラメーター・オプションを指定して **db2pd** コマンドを発行すると、すべてのエンジン・ディスパッチ可能単位 (EDU) が出力中にリストされます。EDU の出力を、インスタンス・レベルやメンバー・レベルなど、細分度のレベルを指定して戻すことができます。Linux および UNIX オペレーティング・システムに限り、**interval** パラメーター・サブオプションを指定して、すべての EDU のスナップショットを指定した間隔だけ空けて 2 つ取ることもできます。**interval** パラメーターを指定すると、出力中に 2 つの列が追加され、この間隔の前後のプロセッサ・ユーザー時間の差分 (USR DELTA 列) と、プロセッサ・システム時間の差分 (SYS DELTA 列) が示されます。

以下の例では、5 秒間のプロセッサ・ユーザー時間の差分とプロセッサ・システム時間の差分が示されます。

```
$ db2pd -edus interval=5
```

```
Database Partition 0 -- Active -- Up 0 days 00:53:29 -- Date 06/04/2010 03:34:59
```

```
List of all EDUs for database partition 0
```

```
db2sysc PID: 1249522
```

```
db2wdog PID: 2068678
```

EDU ID	TID	Kernel TID	EDU Name	USR	SYS	USR DELTA	SYS DELTA
6957	6957	13889683	db2agntdp (SAMPLE) 0	58.238506	0.820466	1.160726	0.014721
6700	6700	11542589	db2agent (SAMPLE) 0	52.856696	0.754420	1.114821	0.015007
5675	5675	4559055	db2agntdp (SAMPLE) 0	60.386779	0.854234	0.609233	0.014304
3088	3088	13951225	db2agntdp (SAMPLE) 0	80.073489	2.249843	0.499766	0.006247
3615	3615	2887875	db2loggw (SAMPLE) 0	0.939891	0.410493	0.011694	0.004204
4900	4900	6344925	db2pfchr (SAMPLE) 0	1.748413	0.014378	0.014343	0.000103
7986	7986	13701145	db2agntdp (SAMPLE) 0	1.410225	0.025900	0.003636	0.000074
2571	2571	8503329	db2ipccm 0	0.251349	0.083787	0.002551	0.000857
7729	7729	14168193	db2agntdp (SAMPLE) 0	1.717323	0.029477	0.000998	0.000038
7472	7472	11853991	db2agnta (SAMPLE) 0	1.860115	0.032926	0.000860	0.000012
3358	3358	2347127	db2loggr (SAMPLE) 0	0.151042	0.184726	0.000387	0.000458
515	515	13820091	db2aiothr 0	0.405538	0.312007	0.000189	0.000178
7215	7215	2539753	db2agntdp (SAMPLE) 0	1.165350	0.019466	0.000291	0.000008
6185	6185	2322517	db2wlm (SAMPLE) 0	0.061674	0.034093	0.000169	0.000100
6442	6442	2756793	db2evmli (DB2DETAILDEADLOCK) 0	0.072142	0.052436	0.000092	0.000063
4129	4129	15900799	db2glock (SAMPLE) 0	0.013239	0.000741	0.000064	0.000001
2	2	11739383	db2alarm 0	0.036904	0.028367	0.000009	0.000009
4386	4386	13361367	db2dlock (SAMPLE) 0	0.015653	0.001281	0.000014	0.000003
1029	1029	15040579	db2fcm (SAMPLE) 0	0.041929	0.016598	0.000010	0.000004
5414	5414	14471309	db2pfchr (SAMPLE) 0	0.000093	0.000002	0.000000	0.000000
258	258	13656311	db2sysc 0	8.369967	0.263539	0.000000	0.000000
5157	5157	7934145	db2pfchr (SAMPLE) 0	0.027598	0.000177	0.000000	0.000000
1543	1543	2670647	db2fcmr 0	0.004191	0.000079	0.000000	0.000000
1286	1286	8417339	db2extev 0	0.000312	0.000043	0.000000	0.000000
2314	2314	14360813	db2licc 0	0.000371	0.000051	0.000000	0.000000
5928	5928	3137537	db2taskd (SAMPLE) 0	0.004903	0.000572	0.000000	0.000000
3872	3872	2310357	db21fr (SAMPLE) 0	0.000126	0.000007	0.000000	0.000000
4643	4643	11694287	db2pclnr (SAMPLE) 0	0.000094	0.000002	0.000000	0.000000
1800	1800	5800175	db2extev 0	0.001212	0.002137	0.000000	0.000000

```

772      772      7925817      db2thcln 0      0.000429      0.000072      0.000000      0.000000
2057    2057    6868993      db2pdbc 0      0.002423      0.001603      0.000000      0.000000
2828    2828    10866809     db2resync 0     0.016764      0.003098      0.000000      0.000000

```

プロセッサ時間を最も消費している EDU に関する情報のみに限定して、戻される出力の量を少なくするには、さらに **top** パラメーター・オプションを組み込むことができます。以下の例では、5 秒の間隔での上位 5 つの EDU のみが戻されます。スタック情報も戻され、DUMPDIR で指定されたディレクトリー・パス (デフォルトは **diagpath**) 内に別個に保管されることになります。

```
$ db2pd -edus interval=5 top=5 stacks
```

```
Database Partition 0 -- Active -- Up 0 days 00:54:00 -- Date 06/04/2010 03:35:30
```

```
List of all EDUs for database partition 0
```

```
db2sysc PID: 1249522
```

```
db2wdog PID: 2068678
```

EDU ID	TID	Kernel TID	EDU Name	USR	SYS	USR DELTA	SYS DELTA
3358	3358	2347127	db2loggr (SAMPLE) 0	0.154906	0.189223	0.001087	0.001363
3615	3615	2887875	db2loggw (SAMPLE) 0	0.962744	0.419617	0.001779	0.000481
515	515	13820091	db2aiothr 0	0.408039	0.314045	0.000658	0.000543
258	258	13656311	db2sysc 0	8.371388	0.264812	0.000653	0.000474
6700	6700	11542589	db2agent (SAMPLE) 0	54.814420	0.783323	0.000455	0.000310

```
$ ls -ltr
```

```
total 552
```

```

drwxrwxr-t 2 vbmithun build      256 05-31 09:59 events/
drwxrwxr-t 2 vbmithun build      256 06-04 03:17 stmmlog/
-rw-r--r-- 1 vbmithun build    46413 06-04 03:35 1249522.3358.000.stack.txt
-rw-r--r-- 1 vbmithun build    22819 06-04 03:35 1249522.3615.000.stack.txt
-rw-r--r-- 1 vbmithun build    20387 06-04 03:35 1249522.515.000.stack.txt
-rw-r--r-- 1 vbmithun build    50426 06-04 03:35 1249522.258.000.stack.txt
-rw-r--r-- 1 vbmithun build    314596 06-04 03:35 1249522.6700.000.stack.txt
-rw-r--r-- 1 vbmithun build     94913 06-04 03:35 1249522.000.processObj.txt

```

例 16: エージェント・イベント・メトリックの表示

db2pd コマンドは、エージェントのイベント・メトリックを返す機能をサポートします。特定の期間にエージェントが状態を変更したかどうかを判断する必要がある場合は、**event** オプションを **-agents** パラメーターと共に使用します。返される **AGENT_STATE_LAST_UPDATE_TIME(Tick Value)** 列は、エージェントによって処理されているイベントが最後に変更された時刻を示します。

AGENT_STATE_LAST_UPDATE_TIME(Tick Value) について以前取得した値と見比べて、エージェントが新しいタスクに移動したか、または長時間にわたって同じタスクを継続して処理しているか判別できます。

```
db2pd -agents event
```

```
Database Partition 0 -- Active -- Up 0 days 03:18:52 -- Date 06/27/2011 11:47:10
```

```
Agents:
```

```
Current agents:      12
```

```
Idle agents:        0
```

```
Active coord agents: 10
```

```
Active agents total: 10
```

```
Pooled coord agents: 2
```

```
Pooled agents total: 2
```

AGENT_STATE_LAST_UPDATE_TIME(Tick Value)	EVENT_STATE	EVENT_TYPE	EVENT_OBJECT	EVENT_OBJECT_NAME
2011-06-27-14.44.38.859785 (5622972377924968075)	IDLE	WAIT	REQUEST	n/a

db2support コマンドを使用した環境情報の収集

DB2 の問題に関する情報を集めるとき、実行すべき最も重要な DB2 ユーティリティーは **db2support** です。 **db2support** コマンドは、DB2 とシステムのあらゆる診断情報を自動的に収集します。さらに、オプションとして、問題の状況について問い合わせる「質問と回答」形式の対話も可能です。

このタスクについて

db2support ユーティリティーを使用すると **GET DATABASE CONFIGURATION FOR database-name** または **LIST TABLESPACES SHOW DETAIL** のようなコマンドを手動で入力する必要がないため、ユーザー・エラーを未然に防ぐことができます。さらに、どの特定のコマンドを実行するか、どのファイルを集めるかを指示する必要がないため、データの収集をより短時間で行うことができます。

手順

- コマンド **db2support -h** を実行して、コマンド・オプションの詳細なリストを表示します。
- 適切な **db2support** コマンドを使用してデータを収集します。

エラーなしで必要なすべての情報を収集するためには、SYSADM 権限を持つユーザー (インスタンス所有者など) として **db2support** ユーティリティーを実行します。SYSADM 権限を持たずにこのコマンドを実行した場合、**QUERY CLIENT** や **LIST ACTIVE DATABASES** といったコマンドが実行される結果として、SQL エラー (例えば SQL1092N) が発生するかもしれません。

IBM ソフトウェア・サポートのための情報収集の目的で **db2support** コマンドを使用する場合には、システムで問題が発生している最中に **db2support** コマンドを実行します。そのようにして、ツールはタイムリーな情報 (オペレーティング・システムのパフォーマンスの詳細など) を収集します。問題発生時にユーティリティーを実行できない場合でも、First Occurrence Data Capture (FODC) 診断ファイルが自動的に生成されるので、問題が停止した後に **db2support** コマンドを実行できます。

FODC パッケージの保管先のディレクトリー・パスが、デフォルトの診断パスではなく、FODCPATH 設定で指定されているパスでもない場合は、**db2support** コマンドに **-fodcpath** パラメーターを指定して FODC パスを指示し、**db2support.zip** ファイルに FODC パッケージを組み込めるようにします。

-fodcpath パラメーターを指定して FODC パッケージへのパスを組み込む必要がある場合を除き、通常は次の基本的な呼び出しを使用するだけで、問題をデバッグするために必要な情報のほとんどが収集されます。

```
db2support <output path> -d <database name>
```

db2support ツールは、デフォルトで DB2 pureScale コンポーネントに固有の診断データのほとんどを収集します。 **-purecluster**、**-purescale**、**-cm**、**-cfs**、または **-udapl** パラメーターを指定すると、**db2support** コマンドはスペースの使用量が多い、または収集に時間のかかる追加の診断データを収集しますが、DB2 pureScale 環境での問題の原因判別にかかる時間を短縮できます。

出力は適切に編成されて ZIP アーカイブ `db2support.zip` に圧縮されるため、任意のシステムに転送して簡単に解凍することができます。

タスクの結果

`db2support` がどのような情報をキャプチャーするかは、コマンドを呼び出す方法、データベース・マネージャーが開始済みかどうか、およびデータベースに接続できるかどうかによって異なります。

すべての状況において、`db2support` ユーティリティは以下の情報を収集します。

- `db2diag` ログ・ファイル
- すべてのトラップ・ファイル
- ロック・リスト・ファイル
- ダンプ・ファイル
- さまざまなシステム関連ファイル
- さまざまなシステム・コマンドからの出力
- `db2cli.ini`

さらに、状況に応じて、`db2support` ユーティリティは以下の情報も収集することがあります。

- アクティブ・ログ・ファイル
- バッファ・プールと表スペース (SQLSPCS.1、SQLSPCS.2) の制御ファイル (`-d` オプションを使用した場合)
- `db2dump` ディレクトリーの内容
- より詳しいシステム情報 (`-s` オプションを使用した場合)
- データベース構成の設定値 (`-d` オプションを使用した場合)
- データベース・マネージャーの構成設定ファイル
- First Occurrence Data Capture (FODC) 情報 (`-fodc` オプションと `-fodcpath` オプションを使用した場合)
- ログ・ファイルのヘッダー・ファイル (`-d` オプションを使用した場合)
- リカバリー履歴ファイル (`-d` オプションを使用した場合)
- `SYSIBM.SYSTABLES` システム・カタログ表、`SYSIBM.SYSINDEXES` システム・カタログ表、および `SYSIBM.SYSDATAPARTITIONS` システム・カタログ表のフォーマット済みデータ (オブティマイザー・モード以外で `-d` オプションと `db2support` コマンドを使用)

`db2support` ユーティリティは、DB2 pureScale 環境で次の情報を収集します。

- クラスタ・マネージャー、クラスタ・ファイル・システム、uDAPL など、DB2 pureScale コンポーネントの診断データ
- クラスタ・マネージャーの追加の診断データ (`-cm` オプションを使用)
- クラスタ・ファイル・システムの追加の診断データ (`-cfs` オプションを使用)
- uDAPL の追加の診断データ (`-udapl` オプションを使用)
- 追加の DB2 診断データ (`-purecluster` または `-purescale` オプションを使用)

HTML 形式のレポート `db2support.html` には、常に以下の情報が含まれます。

- 問題記録 (PMR) 番号 (-n を指定した場合)
- オペレーティング・システムおよびレベル (例えば AIX 5.1)
- DB2 リリース情報
- 32 ビット環境かそれとも 64 ビット環境かの標識
- DB2 インストール・パス情報
- db2nodes.cfg の内容
- プロセッサの数、ディスクの数、メモリー容量
- インスタンスにあるデータベースのリスト
- レジストリー情報と環境 (PATH および LIBPATH を含む)
- 現在のファイル・システムのディスク・フリー・スペースおよび UNIX の i ノード
- Java SDK のレベル
- Java JCC バージョン
- Java JCC 構成
- データベース・マネージャー構成
- sqllib ディレクトリーの **ls -lR** 出力 (または Windows でこれに相当するもの)
- **LIST NODE DIRECTORY** コマンドの結果
- **LIST ADMIN NODE DIRECTORY** コマンドの結果
- **LIST DCS DIRECTORY** コマンドの結果
- **LIST DCS APPLICATIONS EXTENDED** コマンドの結果
- **db2prereqcheck** コマンドの結果
- すべてのインストール済みソフトウェアのリスト
- DB2 ライセンス情報
- DB2 準拠レポート
- 監査構成情報
- CLI 構成情報
- 問題判別の設定値
- **db2trc** コマンドの状況
- ログ・ディレクトリーのリスト表示

DB2 データベース・マネージャーが開始済みの場合には、以下の追加情報が db2support.html ファイルに含まれます。

- クライアント接続の状態
- データベース構成とデータベース・マネージャー構成 (データベース構成を含めるには **-d** オプションが必要です)
- アプリケーションのスナップショット
- メモリー・プールの情報 (サイズと使用量)。 **-d** オプションを使用すると、詳細なデータが収集されます。
- **LIST ACTIVE DATABASES** コマンドの結果
- **LIST DCS APPLICATIONS** コマンドの結果

データベースへの接続が正常に確立されたとき、db2support.html ファイルに次の情報が含まれます。

- ユーザー表の数
- データベース・データのおおよそのサイズ
- データベースのスナップショット
- アプリケーションのスナップショット
- バッファ・プール情報
- **LIST APPLICATIONS** コマンドの結果
- **LIST COMMAND OPTIONS** コマンドの結果
- **LIST DATABASE DIRECTORY** コマンドの結果
- **LIST INDOUBT TRANSACTIONS** コマンドの結果
- **LIST DATABASE PARTITION GROUPS** コマンドの結果
- **LIST DBPARTITIONNUMS** コマンドの結果
- **LIST ODBC DATA SOURCES** コマンドの結果
- **LIST PACKAGES/TABLES** コマンドの結果
- **LIST TABLESPACE CONTAINERS** コマンドの結果
- **LIST TABLESPACES** コマンドの結果
- **LIST DRDA IN DOUBT TRANSACTIONS** コマンドの結果
- DB2 ワークロード・マネージャー情報
- データベース・リカバリー履歴ファイルのリスト
- オプティマイザーのデータベース構成
- データベース構成
- ノード・グループ情報
- ストレージ・グループ情報
- ストリング ID の数
- 表のリスト

db2support.html ファイルが **db2support** パッケージの最上位に含まれており、**db2support** コマンドによって収集される診断データを素早く検索できます。この HTML ファイルには、db2support.html ファイルで収集されたデータのリンクが含まれます。リンクの参照先となる対応するフラット・ファイルは、db2support パッケージのサブディレクトリーにあります。 db2support.map という名前の、マップ・ファイルのプレーン・テキスト・バージョンのファイルも、**db2support** パッケージに含まれています。

db2support.zip ファイルの内容の例

-unzip パラメーターを指定した **db2support** コマンドを使用して、db2support.zip ファイルの内容をローカルに抽出できます。オプションで内容の抽出先のディレクトリー・パスを指定できます。**-unzip** オプションを使用して、アーカイブされた診断データの内容を抽出することもできます。その際、追加のソフトウェアは必要ありません。実際の内容を抽出せず、db2support.zip ファイルに組み込まれているファイルを確認するだけの場合は、代わりに **-unzip list** パラメーターと **db2support** コマンドを併用できます。

db2support.zip ファイルの内容の例の場合、以下のコマンドが実行されました。

```
db2support . -d sample -c -f -st "select * from staff"
```

db2support.zip ファイルが解凍される際、以下のファイルとディレクトリーが収集されました。

- DB2CONFIG/ - 構成情報 (例えば、データベース、データベース・マネージャ、BP、CLI、および Java Developer Kit)
- DB2DUMP/ - db2diag ログ・ファイルのここ 3 日間の内容
- DB2MISC/ - sql1lib ディレクトリーのリスト
- DB2SNAP/ - DB2 コマンド (例えば、**db2set**、**LIST TABLES**、**LIST INDOUBT TRANSACTIONS**、**LIST APPLICATIONS** など) の出力
- PURESCALE/ - クラスタ・マネージャ、クラスタ・ファイル・システム、uDAPL などの、DB2 pureScale コンポーネントの診断情報
- db2supp_opt.zip - オプティマイザ問題の診断情報
- db2supp_system.zip - オペレーティング・システム情報
- db2support.html - HTML フォーマットでリストされた db2support.zip ファイルの各サブディレクトリーで収集されたフラット・ファイルへのマップ、および HTML セクションにフォーマットされた診断情報
- db2support.log - **db2support** 収集の診断ログ情報
- db2support_options.in - **db2support** 収集を開始するために使用されるコマンド行のオプション
- db2support.map - プレーン・テキスト・フォーマットでリストされた db2support.zip ファイルの各サブディレクトリーで収集されたフラット・ファイルへのマップ

オプティマイザに関する情報は、db2supp_opt.zip ファイル内にあります。このファイルの解凍は、以下のディレクトリーにあります。

- OPTIMIZER/ - オプティマイザ問題の診断情報
- OPTIMIZER/optimizer.log - ファイルにすべてのアクティビティのログが含まれます
- OPTIMIZER/CATALOGS - 以下のサブディレクトリー内の LOB を含むすべてのカタログ (カタログ表内の LOB 列が空でない場合のみ生成される)
 - FUNCTIONS
 - INDEXES
 - NODEGROUPS
 - ROUTINES
 - SEQUENCES
 - TABLES
 - VIEWS
- OPTIMIZER/DB2DUMP - db2serv 出力 (serv.* および serv2.* 出力ファイル)

システム情報は、db2supp_system.zip ファイル内にあります。このファイルの解凍は、以下のファイルおよびディレクトリーにあります。

- DB2CONFIG/ - db2cli.ini (~/sql1lib/cfg のファイル)

- DB2MISC/ - DB2SYSTEM ファイル (バイナリー) など
- OSCONFIG/ - さまざまなオペレーティング・システム情報ファイル (例えば、netstat、services、vfs、ulimit、および hosts)
- OSSNAP/ - オペレーティング・システム・スナップショット (例えば、iostat、netstat、uptime、vmstat、および ps_elf)
- SQLDBDIR/ - 重要なバッファ・プール・メタファイル (~/sqllib/sqlbdir)
- SQLGWDIR/ - DCS ディレクトリー (~/sqllib/sqlgwdir のファイル)
- SQLNODIR/ - Node ディレクトリー (~/sqllib/sqlnodir のファイル)
- SPMLOG/ - ~/sqllib/spmlog のファイル
- report.log - すべての収集アクティビティのログ

DB2 コピーの検証

db2val コマンドにより、DB2 コピーが適正に機能していることを確認できます。

このタスクについて

db2val ツールは、インストール・ファイル、インスタンス、データベース作成、データベースへの接続、およびパーティション・データベース環境の状態を検証することで、DB2 コピーのコア機能を検査します。この検証は、DB2 コピーを Linux および UNIX オペレーティング・システム上に tar.gz ファイルを使用して手動でデプロイした場合に役立ちます。**db2val** コマンドにより、すべての構成が正しく完了したことを素早く確認すること、および DB2 コピーが予期するとおりのものであることを確認できます。インスタンスおよびデータベースを指定すること、またはすべてのインスタンスに対して **db2val** を実行することができます。**db2val** コマンドは、*DB2-install-path*¥bin および sqllib/bin ディレクトリーにあります。

例

例えば、DB2 コピーのすべてのインスタンスを妥当性検査するには、次のコマンドを実行します。

```
db2val -a
```

db2val コマンドに関する完全な詳細および追加の例については、『db2val - DB2 コピー検証ツール・コマンド』のトピックを参照してください。

基本的なトレース診断

DB2 で問題が繰り返し起こる場合、トレースによって、追加情報を収集できる場合があります。通常の場合では、IBM ソフトウェア・サポートに依頼された場合のみ、トレースを使用してください。トレースを取る処理には、トレース機能の設定、問題の再現、およびデータの収集が必要になります。

トレースによって集められる情報量は、急速に増加します。トレースを取る場合はエラーの状態のみを収集し、できる限りその他のアクティビティを避けてください。トレースを取るときは、問題の再現に、最小のシナリオを使用してください。

トレースを収集すると、DB2 インスタンスのパフォーマンスに悪影響が及ぶことがあります。性能低下の程度は、問題のタイプ、およびトレース情報の収集に使用されるリソースの数に応じて異なります。

トレースが求められるときには、IBM ソフトウェア・サポートでは以下の情報が用意されています。

- 簡単で段階的な手順
- 各トレースを実行する場所についての説明
- 何をトレースすべきかについての説明
- トレースを求める理由についての説明
- バックアウト手順 (すべてのトレースを無効にする方法など)

どのトレースを取得するかに関しては、IBM ソフトウェア・サポートの指示に従う必要がありますが、いつ特定のトレースを取得するよう求められるかについては、以下に示すようないくつかの一般ガイドラインがあります。

- インストール中に問題が発生し、デフォルトのインストール・ログがその問題の原因を調べるのに十分ではない場合には、インストールのトレースが適切です。
- CLI アプリケーションで問題が生じ、アプリケーションの外部でその問題を再作成できない場合は、CLI トレースが適切です。
- JDBC アプリケーションで問題が生じ、アプリケーションの外部でその問題を再作成できない場合には、JDBC トレースが適切です。
- 問題が DRDA レイヤーで通信中の情報と直接関連しているなら、DRDA トレースが適切です。
- トレースが実行できるその他のすべての状態では、DB2 トレースが最も適切でしょう。

トレース情報は、エラーの診断に常に役立つとはかぎりません。例えば、以下の状態では、エラー状態を収集しない場合があります。

- 指定したトレース・バッファ・サイズは、トレース・イベントの完全セットを保持するのに十分な大きさがなく、トレースがファイルへの書き込みを停止したとき、またはトレースがラップされた時に、情報が失われました。
- トレース・シナリオが、エラー状態を再現しませんでした。
- エラー状態は再現されましたが、問題が発生した場所などの前提事項が間違っていました。例えば、サーバーで実際のエラーが発生している間に、トレースはクライアント・ワークステーションで収集されました。

DB2 トレース

db2trc を使用した DB2 トレースの取得

db2trc コマンドは、DB2 で提供されるトレース機能を制御します。このトレース機能は、操作に関する情報を記録し、その情報を読み取り可能な形式にフォーマットします。

なお、トレースの実行中は追加のプロセッサ使用量が発生するという点に注意してください。このため、トレース機能を使用可能にすると、システムのパフォーマンスに影響が及ぶ可能性があります。

一般に DB2 トレースは、IBM ソフトウェア・サポートおよび開発チームがトラブルシューティングのために使用します。調査中の問題に関するより詳しい情報を得るためにトレースを実行することはできますが、DB2 のソース・コードに関する知識がなければ、その用途はごく限られたものになります。

しかし、それらを取得するよう要求される場合に備えて、トレースをオンにしたりトレース・ファイルをダンプしたりするための正しい方法を理解しておくことは大切です。

注: **db2trc** を使用するには、SYSADM、SYSCTRL または SYSMMAINT 権限のいずれかが必要です。

使用可能なオプションの概要を把握するには、何もパラメーターを指定せずに **db2trc** コマンドを実行します。

```
C:¥>db2trc
Usage: db2trc (chg|clr|dmp|flw|fmt|inf|off|on) options
```

特定の **db2trc** コマンド・パラメーターについての詳細情報を取得するには、**-u** オプションを使用します。例えば、トレースをオンにする方法の詳細を調べるには、次のコマンドを実行します。

```
db2trc on -u
```

これにより、DB2 トレースをオンにする際に指定可能なすべての追加オプション ("facilities" というラベルが付いている) に関する情報が提供されます。

トレースをオンにするときに、最も重要なオプションは **-L** です。このオプションは、トレースした情報を格納するのに使用するメモリー・バッファのサイズを指定します。バッファのサイズはバイト単位またはメガバイト単位で指定できます。(メガバイトを指定するには、値の後ろに "M" または "m" のいずれかを付加します。) トレース・バッファのサイズはメガバイト単位で 2 のべき乗にする必要があります。この要件を満たしていないサイズを指定すると、バッファのサイズは自動的にメガバイト単位で 2 のべき乗となるうち最も近接した容量に端数切り捨てられます。

バッファが小さすぎると、情報が失われる可能性があります。デフォルトでは、バッファが満杯になると最も新しいトレース情報のみが保持されます。バッファが大きすぎると、ファイルを IBM ソフトウェア・サポート・チームに送信するのが難しくなる可能性があります。

比較的短い操作 (データベース接続など) をトレースする場合は、通常約 8 MB 程度で十分です。

```
C:¥> db2trc on -l 8M
トレースはオンになります。
```

しかし、より大きな操作をトレースする場合や、多数の処理が同時に進行するような場合は、より大きなトレース・バッファが必要になります。

トレースは、ほとんどのプラットフォームでいつでもオンにでき、その動作も前述のとおりです。ただし、以下のような注意すべき状況もあります。

1. 複数データベース・パーティション・システムでは、それぞれの物理データベース・パーティション (論理データベース・パーティションではない) に対して、トレースを実行する必要があります。
2. HP-UX、Linux、Solaris のプラットフォームでは、インスタンスの開始後にトレースをオフにすると、トレースを次回開始するときには、サイズの指定に関わりなく、ごく小さなバッファが使用されます。例えば、昨日 **db2trc on -l 8m** でトレースをオンにし、1 つのトレースを収集してから、トレースをオフにしたとします (**db2trc off**)。今日は、インスタンスを終了して再始動する作業を行わず、メモリー・バッファを 32 MB に設定してトレースを実行するとしましょう (**db2trc on -l 32m**)。この場合、トレースはごく小さなバッファを使用します。これらのプラットフォームでトレースを効果的に実行するには、トレースをオンにしてから必要なバッファ・サイズでインスタンスを開始し、それから必要に応じてバッファを「クリア」してください。

収集されるデータやフォーマットされるデータの量を少なくするために、**db2trc** コマンドは複数のマスク・オプションをサポートしています。収集されるデータの量を減らすと、トレースの収集が継続しているために発生する追加のプロセッサ使用量を減らすことができますし、収集されるデータの選択範囲を狭めることができるため、有効です。収集されるデータの選択範囲を狭めると、問題診断の時間短縮にも役立ちます。

通常は、IBM サポートの指導に従い **-m** マスク・オプションを使用します。しかし、**-p** マスク・オプションを使用して、特定のプロセス ID (およびオプションでスレッド ID) のトレースのみ収集することもできます。例えば、プロセス 77 とスレッド 1、2、3、および 4 のトレースと、プロセス 88 とスレッド 5、6、7、および 8 のトレースを使用可能にするには、以下の構文を使用します。

```
db2trc on -p 77.1.2.3.4,88.5.6.7.8
```

trcon スクリプトと troff スクリプトを使用してトレースの収集を制御する

2 つのスクリプトを使用して、手動で発行する複数のコマンドを単一のスクリプト呼び出しに置き換えることにより、トレースの収集を単純化できます。

db2trcon スクリプトは、トレースをオンにし、複数のオプションをサポートします。このスクリプトを使用して、指定された期間だけ **db2trc** をオンにし、エンジン・ディスパッチ可能単位 (EDU) を消費する上位のプロセッサ時間のみトレースを収集するよう指定し、自動的にダンプ・ファイル、フロー・ファイル、およびフォーマット・ファイルを生成することができます。例えば、EDU を消費する上位 5 つのプロセッサ時間のトレースを 45 秒間オンにし、スクリプトの実行を開始した時刻から 15 秒間隔でサンプルを取るには、以下のコマンドを発行します。

```
db2trcon
-duration 45 -top 5 -interval 15 -flw -fmt
```

指定された期間の経過後に **db2trc** がオフになると、**db2trcon** は自動的にダンプ・ファイル、フロー・ファイル、およびフォーマット・ファイルを生成します。

db2trc off は単一のコマンドを使用して、トレースをオフにし、自動的にダンプ・ファイル、フロー・ファイル、およびフォーマット・ファイルを生成できます。例えば、**-force** を指定して **db2trc** をオフにし、フロー・ファイル、フォーマット・ファイル、およびダンプ・ファイルを生成するには、以下のコマンドを発行します。

```
db2trcoff -flw  
-fmt -force
```

db2trcon スクリプトを使用してトレースをオンにし、期間を指定した場合は、改めて db2troff コマンドを発行する必要がないことに注意してください。

DB2 トレース・ファイルのダンプ

ON オプションを使用してトレース機能を使用可能にした後、その後のインスタンスの作業はすべてトレースされます。

トレースが実行されている間は、clr オプションを使用してトレース・バッファークリアできます。トレース・バッファーにある既存の情報はすべて削除されず。

```
C:¥>db2trc clr  
Trace has been cleared
```

トレースの対象となる操作が終了したら、次のように、dmp オプションとトレース・ファイルの名前を指定して、メモリー・バッファークをディスクにダンプします。例えば、

```
C:¥>db2trc dmp trace.dmp  
Trace has been dumped to file
```

トレース・バッファークをディスクにダンプした後も、トレース機能の実行は継続されます。トレースをオフにするには、OFF オプションを使用します。

```
C:¥>db2trc off  
Trace is turned off
```

DB2 トレース・ファイルのフォーマット

db2trc dmp というコマンドによって作成されるダンプ・ファイルは、バイナリー形式であり、読み取り可能ではありません。トレース・ファイルが読み取り可能かどうかを検証するには、バイナリー・トレース・ファイルをフォーマットして、フロー制御を表示し、フォーマットされた出力を NULL デバイスに送信します。

以下の例は、このタスクを実行するコマンドを表示します。

```
db2trc flw example.trc nul
```

example.trc は、dmp オプションを使用して作成されたバイナリー・ファイルです。

このコマンドの出力は、ファイルの読み取りに問題があったかどうか、およびトレースがラップされたかどうかを、明示的に知らせます。

この時点で、ダンプ・ファイルを IBM ソフトウェア・サポートに送信することができます。サポート・チームによって、ご使用の DB2 サービス・レベルに基づいてそのファイルがフォーマットされます。しかし、DB2 サポートに送信する前に、そのダンプ・ファイルを ASCII フォーマットにフォーマットするよう求められる場合があります。これは、flw オプションおよび fmt オプションを使用することで実行されます。バイナリー・ダンプ・ファイルと、作成する ASCII ファイルの名前を次のように指定する必要があります。

```

C:¥>db2trc flw trace.dmp trace.flw
C:¥Temp>db2trc flw trace.dmp trace.flw
Total number of trace records      : 18854
Trace truncated                    : NO
Trace wrapped                      : NO
Number of trace records formatted  : 1513 (pid: 2196 tid 2148 node: -1)
Number of trace records formatted  : 100 (pid: 1568 tid 1304 node: 0)
...

C:¥>db2trc fmt trace.dmp trace.fmt
C:¥Temp>db2trc fmt trace.dmp trace.fmt
Trace truncated                    : NO
Trace wrapped                      : NO
Total number of trace records      : 18854
Number of trace records formatted  : 18854

```

この出力で "Trace wrapped" が "YES" と表示されていれば、トレース・バッファが小さすぎて、トレース期間の間に収集された情報を全部は格納できなかったこととなります。場合によっては、折り返されたトレースでも特に問題はありません。最も新しい情報（これは、**-i** オプションが指定されていない限り、保持されているデフォルトの情報を表す）を対象としている場合は、トレース・ファイルに残っている情報だけで十分です。しかし、トレース期間の最初に起こったことを対象としている場合や、その期間に起こったことすべてを対象としている場合は、トレース・バッファのサイズを大きくして操作をやり直すこととなります。

バイナリー・ファイルを実際に読める形のテキスト・ファイルにフォーマットするときには使用できるオプションがあります。例えば、`db2trc fmt -xml trace.dmp trace.fmt` を使用すれば、バイナリー・データを変換し、その結果を XML 構文解析可能フォーマットに出力できます。追加のオプションについては、トレース・コマンド (**db2trc**) の詳細な説明を参照してください。

注意しなければならないことがもう 1 つあります。Linux と UNIX のオペレーティング・システムでは、重大エラーによってインスタンスがシャットダウンされると、DB2 によってトレース・バッファが自動的にディスクにダンプされます。このため、インスタンスの異常終了時にトレースを使用可能にすると、診断ディレクトリに `db2trdmp.###` (### はデータベース・パーティション番号) という名前のファイルが作成されます。このような動作は、Windows のプラットフォームでは発生しません。その場合には、トレースを手動でダンプする必要があります。

要約すると、**db2trc** コマンドの一般的な順序の例は以下のようになります。

```

db2trc on -l 8M
db2trc clr
<Execute problem recreation commands>
db2trc dump db2trc.dmp
db2trc off
db2trc flw db2trc.dmp <filename>.flw
db2trc fmt db2trc.dmp <filename>.fmt
db2trc fmt -c db2trc.dmp <filename>.fmtc

```

DRDA トレース・ファイル

DRDA トレースを分析する前に、DRDA はデータ構造および通信構造を定義するためのオープン・スタンダードであることを理解しておく必要があります。DRDA は、例えば伝送用データの編成方法、その情報の通信方法などに関するいくつかの規則から成ります。

これらの規則は、以下の解説書で定義されています。

- DRDA V3 第 1 巻: Distributed Relational Database Architecture™
- DRDA V3 第 2 巻: Formatted Data Object Content Architecture
- DRDA V3 第 3 巻: Distributed Data Management Architecture

これらの資料の PDF 版は www.opengroup.org から入手できます。

db2drdat ユーティリティーは DRDA アプリケーション・リクエスター (AR) と DB2 DRDA アプリケーション・サーバー (AS) の間でやり取りされるデータを記録します (例えば、DB2 Connect とホストまたは Power Systems™ サーバーのデータベース・サーバーとの間)。

トレース・ユーティリティー

db2drdat ユーティリティーを利用して、DB2 Connect サーバー (IBM Data Server Clientの代理) と IBM メインフレーム・データベース・サーバーとの間で交換されたデータを記録することができます。

データベース管理者として (またはアプリケーション開発者として)、このデータ・フローがどのように働くかを理解することは有用です。この知識は、特定の問題の起点を判別するのに役立つからです。例えば、CONNECT TO データベース・ステートメントを IBM メインフレーム・データベース・サーバーに対して発行したが、コマンドが失敗して、失敗の戻りコードを受け取ったとします。そのとき、どのような情報が IBM メインフレーム・データベース・サーバー管理システムに送られたかを正確に理解していれば、たとえ戻りコードの情報が一般的なものであったとしても、失敗の原因を判別することができます。ユーザー自身による単純なエラーが、多くの失敗の原因となっています。

db2drdat からの出力は、DB2 Connect ワークステーションと IBM メインフレーム・データベース・サーバー管理システムとの間で交換されたデータ・ストリームをリストします。IBM メインフレーム・データベース・サーバーへ送られたデータには SEND BUFFER とラベル付けされ、IBM メインフレーム・データベース・サーバーから受け取られたデータは RECEIVE BUFFER とラベル付けされます。

受信バッファが SQLCA 情報を含んでいる場合、その後、このデータの書式化された解釈が続き、SQLCA とラベル付けされます。SQLCA の SQLCODE フィールドは、IBM メインフレーム・データベース・サーバーが戻したとおりのマップされていない値です。送信バッファと受信バッファは、ファイル内で最も古いものから順に、最新のものへと配置されます。それぞれのバッファには、以下のものが入ります。

- プロセス ID
- SEND BUFFER、RECEIVE BUFFER、または SQLCA ラベル。バッファ内の 1 番目の DDM コマンドまたはオブジェクトは、DSS TYPE とラベル付けされています。

送信バッファと受信バッファ内の残りのデータは、以下のものを構成する 5 つの列に分けられます。

- バイト・カウント。

- 第 2 列および第 3 列は、2 つのシステム間で交換される DRDA データ・ストリームを ASCII または EBCDIC で表します。
- 第 2 列および第 3 列の ASCII 表示。
- 第 2 列および第 3 列の EBCDIC 表示。

トレース出力

db2drdat ユーティリティーは、*tracefile* に以下の情報を書き込みます。

- -r
 - DRDA 応答/オブジェクトのタイプ
 - 受信バッファ
- -s
 - DRDA 要求のタイプ
 - 送信バッファ
- -c
 - SQLCA
- TCP/IP エラー情報
 - 受信関数の戻りコード
 - 重大度
 - 使用したプロトコル
 - 使用した API
 - 関数
 - エラー番号

注:

1. 終了コードのゼロ値は、そのコマンドが正常に完了したことを示し、ゼロ以外の値は、そのコマンドが正常に完了しなかったことを示します。
2. 戻されるフィールドは、使用した API によって変わります。
3. 戻されるフィールドは、同じ API の場合でさえ、DB2 Connect が実行しているプラットフォームによって変わります。
4. **db2drdat** コマンドが、すでに存在しているファイルへ出力を送信した場合、ファイル上の権限により消去を禁止しているのではない限り、以前のファイルは消去されてしまいます。

トレース出力ファイルの分析

以下の情報が **db2drdat** トレースに取り込まれます。

- クライアント・アプリケーションのプロセス ID (PID)
- データベース接続サービス (DCS) ディレクトリーでカタログされた RDB_NAME
- DB2 Connect CCSID (コード化文字セット ID)
- IBM メインフレーム・データベース・サーバー CCSID
- DB2 Connect システムの通信相手の IBM メインフレーム・データベース・サーバー管理システム。

1 番目のバッファには、IBM メインフレーム・データベース・サーバー管理システムに送信される交換サーバー属性 (EXCSAT) およびアクセス RDB (ACCRDB) コマンドが入っています。そして、それらのコマンドを CONNECT TO データベース・コマンドの結果として送信します。2 番目のバッファには、DB2 Connectが IBM メインフレーム・データベース・サーバー管理システムから受け取る応答が入ります。このバッファには、交換サーバー属性応答データ (EXCSATRD) およびアクセス RDB 応答メッセージ (ACCRDBRM) が入っています。

EXCSAT

EXCSAT コマンドには、サーバー名 (SRVNAM) オブジェクトにより指定されたクライアントのワークステーション名が入っています。そのオブジェクトのコード点は X'116D' であり、DDM 仕様に従っています。**EXCSAT** コマンドは、1 番目のバッファにあります。**EXCSAT** コマンドでは、値 X'9481A292' (CCSID 500 によりコード化) は、X'116D' を除去すると、*mask* に変換されます。

EXCSAT コマンドには、EXTNAM (外部名) オブジェクトも含まれます。このオブジェクトは、しばしば、IBM メインフレーム・データベース管理システムについての診断情報に入れられます。それは、20 バイトのアプリケーション ID、続いて 8 バイトのプロセス ID (または 4 バイトのプロセス ID と 4 バイトのスレッド ID) から成ります。それは、コード点 X'115E' で表され、この例ではその値は db2bp で、ブランクが埋め込まれ、000C50CC へと続きます。Linux または UNIX IBM Data Server Clientについては、この値は **ps** コマンドを使用して関連させることができ、このコマンドは活動状態のプロセスについてのプロセス状況情報を標準出力に戻します。

ACCRDB

ACCRDB コマンドは、RDBNAM オブジェクトにある RDB_NAME を含んでいます。そのコード点は X'2110' です。**ACCRDB** コマンドは、1 番目のバッファの中で **EXCSAT** コマンドの後に続きます。**ACCRDB** コマンドでは、値 X'E2E3D3C5C3F1' は、X'2110' を除去すると、STLEC1 に変換されます。これは、DCS ディレクトリーにあるターゲット・データベース名フィールドに対応しています。

アカウント・ストリングのコード点は X'2104' です。

DB2 Connect ワークステーション用に構成されたコード・セットは、**ACCRDB** コマンドの中でコード点が X'119C' である CCSID オブジェクトの CCSIDSBC (1 バイト文字の CCSID) の位置により示されます。この例では、CCSIDSBC は X'0333' です。これは 819 になります。

コード点が X'119D' である追加オブジェクト CCSIDDBC (2 バイト文字の CCSID) と、コード点が X'119E' である追加オブジェクト CCSIDMBC (混合バイト文字の CCSID) も、**ACCRDB** コマンドに存在します。この例では、CCSIDDBC は X'04B0' (1200)、CCSIDMBC は X'0333' (819) です。

EXCSATRD および ACCRDBRM

また CCSID 値は、IBM メインフレーム・データベース・サーバーから、2 番目のバッファ内にあるアクセス RDB 応答メッセージ (ACCRDBRM) にも戻されます。このバッファには、EXCSATRD とそれに続く ACCRDBRM が入っています。サンプルの出力ファイルには、IBM メインフレーム・データベース・サーバー・システム用の 2 つの CCSID の値が

含まれます。値は 1208 (1 バイト文字および混合バイト文字の両方の場合) と 1200 (2 バイト文字の場合) になります。

IBM メインフレーム・データベース・サーバーから戻るコード・ページを DB2 Connect が認識しない場合は、SQLCODE -332 がソースおよびターゲット・コード・ページと共にユーザーに戻されます。DB2 Connect から送信されたコード・セットを IBM メインフレーム・データベース・サーバーが認識しない場合、VALNSPRM (サポートされていないパラメーター値、DDM コード・ポイント X'1252') を戻し、ユーザー用に SQLCODE -332 に変換されます。

ACCRDBRM には、パラメーター PRDID (製品固有 ID、コード点は X'112E') も含まれています。値は、X'C4E2D5F0F8F0F1F5' (EBCDIC では DSN08015) です。標準では、DSN は DB2 for z/OS® です。バージョン番号も示されます。ARI は DB2 Server for VSE & VM、SQL は DB2データベースまたは DB2 Connect、QSQ は IBM DB2 for IBM iです。

トレース出力ファイル・サンプル

以下の図は、出力例を示しており、DB2 Connect ワークステーションとホストまたは System i® データベース・サーバーとの間で交換されるいくつかの DRDA データ・ストリームを例示しています。ユーザーの観点からは、コマンド行プロセッサ (CLP) を使用して CONNECT TO データベース・コマンドを実行しています。

583 ページの図 35 は TCP/IP 接続で DB2 Connect Enterprise Edition バージョン 9.1 および DB2 for z/OS バージョン 8 を使用します。

```
1 data DB2 UDB DRDA Communication Manager sqljcsend fnc (3.3.54.5.0.100)
pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 0 probe 100
bytes 16
```

```
Data1 (PD_TYPE_UINT,8) unsigned integer:
233
```

```
2 data DB2 UDB DRDA Communication Manager sqljcsend fnc (3.3.54.5.0.1177)
pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 19532 probe 1177
bytes 250
```

```
SEND BUFFER(AR):
```

	EXCSAT RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	00C3D041000100BD 1041007F115E8482	...A.....A...^..	.C}.....";db
0010	F282974040404040 4040404040404040	...@@@@@@@@@@@@@	2bp
0020	4040F0F0F0C3F5F0 C3C3F0F0F0000000	@@.....	000C50CC000...
0030	0000000000000000 0000000000000000
0040	0000000000000000 000000000060F0F0`-00
0050	F0F1A2A495404040 4040404040404040@@@@@@@@@@@	01sun
0060	4040404040404040 4040404040404040	@@@@@@@@@@@@@@@@@	
0070	C4C5C3E5F8404040 F0A2A49540404040@@@...@@@	DECV8 0sun
0080	4040404040404040 4000181404140300	@@@@@@@@@.....
0090	0724070008147400 05240F0008144000	.\$...t..\$...@.
00A0	08000E1147D8C4C2 F261C1C9E7F6F400G....a.....QDB2/AIX64.
00B0	08116D9481A29200 0C115AE2D8D3F0F9	..m.....Z.....	.._mask...]SQL09
00C0	F0F0F0	...	000

	ACCSEC RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	0026D00100020020 106D000611A20003	.&.....m.....	..}.....s..
0010	00162110E2E3D3C5 C3F1404040404040	..!.....@@@@@@@STLEC1
0020	40404040404040	@@@@@@@	

```
3 data DB2 UDB DRDA Communication Manager sqljcreceive fnc (3.3.54.3.0.100)
pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110546200 probe 100
bytes 12
```

```
Data1 (PD_TYPE_UINT,4) unsigned integer:
105
```

```
4 data DB2 UDB DRDA Communication Manager sqljcreceive fnc (3.3.54.3.0.1178)
pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110549755 probe 1178
bytes 122
```

```
RECEIVE BUFFER(AR):
```

	EXCSATRD OBJDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	0059D04300010053 1443000F115EE5F8	.Y.C...S.C...^..	..}.....;V8
0010	F1C14BE2E3D3C5C3 F100181404140300	..K.....	1A.STLEC1.....
0020	0724070007147400 05240F0007144000	.\$...t..\$...@.
0030	0700081147D8C4C2 F20014116DE2E3D3G.....m.....QDB2..._STL
0040	C5C3F14040404040 4040404040000C11	...@@@@@@@@@@@...	EC1 ...
0050	5AC4E2D5F0F8F0F1 F5	Z.....]DSN08015

	ACCSECRD OBJDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	0010D0030002000A 14AC000611A20003}.....s..

```
5 data DB2 UDB DRDA Communication Manager sqljcsend fnc (3.3.54.5.0.100)
pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110656806 probe 100
bytes 16
```

```
Data1 (PD_TYPE_UINT,8) unsigned integer:
233
```

図 35. トレース出力の例 (TCP/IP 接続)

6 data DB2 UDB DRDA Communication Manager sqljcsend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110659711 probe 1177
 bytes 250

SEND BUFFER(AR):

	SECCHK RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	003CD04100010036	106E000611A20003	.<.A...6.n.....
0010	00162110E2E3D3C5	C3F1404040404040	..!.....@@@...
0020	40404040404000C	11A1D9858799F485	@...@...Regr4e
0030	A599000A11A09585	A6A39695vr....newton

	ACCRDB RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	00ADD001000200A7	20010006210F2407!.\$..}
0010	00172135C7F9F1C1	F0C4F3C14BD7C1F8	..!5.....K...G91A0D3A.PA8
0020	F806030221064600	162110E2E3D3C5C3	...!.F...!.....8.....STLEC
0030	F140404040404040	4040404040000C11	.@...@...@...@...1
0040	2EE2D8D3F0F9F0F0	F000D002FD8E3C4/...SQL0900....QTD
0050	E2D8D3C1E2C30016	00350006119C03335.....3 SQLASC.....
0060	0006119D04B00006	119E0333003C21043.

7 data DB2 UDB DRDA Communication Manager sqljcreceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259908001 probe 100
 bytes 12

Data1 (PD_TYPE_UINT,4) unsigned integer:
 176

8 data DB2 UDB DRDA Communication Manager sqljcreceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259911584 probe 1178
 bytes 193

RECEIVE BUFFER(AR):

	SECCHKRM RPYDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	0015D0420001000F	1219000611490000	...B.....I..}
0010	000511A400	u.

	ACCRDBRM RPYDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	009BD00200020095	2201000611490000"....I..}
0010	000D002FD8E3C4E2	D8D3F3F7F0000C11	.../.....QTDSQL370...
0020	2EC4E2D5F0F8F0F1	F5001600350006115...DSN08015.....
0030	9C04B80006119E04	B80006119D04B000
0040	0C11A0D5C5E6E3D6	D540400006212524@...!%\$...NEWTON
0050	34001E244E000624	4C00010014244D00	4..\$N..\$L...\$M.+...<.....(.
0060	06244FFFFF000A11	E8091E768301BE00	.\$0.....v.....!.....Y....c...
0070	2221030000000005	68B3B8C7F9F1C1F0	"!.....h.....G91A0
0080	C4F3C1D7C1F8F840	4040400603022106@...!.. D3APA88
0090	46000A11E8091E76	831389	F.....v....Y....c.i

9 data DB2 UDB DRDA Communication Manager sqljcsend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364420503 probe 100
 bytes 16

Data1 (PD_TYPE_UINT,8) unsigned integer:
 10

図 36. トレース出力の例 (TCP/IP 接続) (続き)

10 data DB2 UDB DRDA Communication Manager sqljcsend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364440751 probe 1177
 bytes 27

SEND BUFFER(AR):

	RDBCMM RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	000AD00100010004 200E}.....

11 data DB2 UDB DRDA Communication Manager sqljcreceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475009631 probe 100
 bytes 12

Data1 (PD_TYPE_UINT,4) unsigned integer:
 54

12 data DB2 UDB DRDA Communication Manager sqljcreceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475014579 probe 1178
 bytes 71

RECEIVE BUFFER(AR):

	ENDUOWRM RPYDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	002BD05200010025 220C000611490004	..+R...%"....I..	..}.....
0010	00162110E2E3D3C5 C3F1404040404040	..!.....@#####STLEC1
0020	4040404040400005 211501	@#####!...

	SQLCARD OBJDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	000BD00300010005 2408FF\$..	..}.....

13 data DB2 UDB DRDA Communication Manager sqljcsend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721710319 probe 100
 bytes 16

Data1 (PD_TYPE_UINT,8) unsigned integer:
 126

14 data DB2 UDB DRDA Communication Manager sqljcsend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721727276 probe 1177
 bytes 143

SEND BUFFER(AR):

	EXCSQLIMM RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	0053D0510001004D 200A00442113E2E3	.S.Q...M ..D!...	..}....(.....ST
0010	D3C5C3F140404040 4040404040404040#####	LEC1
0020	D5E4D3D3C9C44040 4040404040404040#####	NULLID
0030	4040E2D8D3C3F2C6 F0C1404040404040	@.....#####	SQLCF0A
0040	4040404041414141 41484C5600CB0005	@#####AHLV....<.....
0050	2105F1	!..	..1

	SQLSTT OBJDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	002BD00300010025 2414000000001B64	..+.....%\$.....d	..}.....
0010	656C657465206672 6F6D206464637375	elete from ddcsu	..%.....?_.....
0020	73312E6D79746162 6C65FF	sl.mytable.	..._`./.%..

15 data DB2 UDB DRDA Communication Manager sqljcreceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832901261 probe 100
 bytes 12

Data1 (PD_TYPE_UINT,4) unsigned integer:
 102

図 37. トレース出力の例 (TCP/IP 接続) (続き)

16 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832906528 probe 1178
 bytes 119

RECEIVE BUFFER(AR):

	SQLCARD OBJDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	0066D00300010060 240800FFFFFF3434	.f.....`\$.44	..}....-.....
0010	3237303444534E58 4F544C2000FFFFFFE	2704DSNXOTL+!.<.....
0020	0C00000000000000 00FFFFFFF000000
0030	00000000000572020 2057202020202020W W
0040	001053544C454331 2020202020202020	..STLEC1<.....
0050	2020000F44444353 5553312E4D595441	..DDCSUS1.MYTA(...
0060	424C450000FF	BLE...	..<....

17 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833156953 probe 100
 bytes 16

Data1 (PD_TYPE_UINT,8) unsigned integer:
 10

18 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833159843 probe 1177
 bytes 27

SEND BUFFER(AR):

	RDBRLLBCK RQSDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	000AD00100010004 200F}.....

19 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943302832 probe 100
 bytes 12

Data1 (PD_TYPE_UINT,4) unsigned integer:
 54

20 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943306288 probe 1178
 bytes 71

RECEIVE BUFFER(AR):

	ENDUOWRM RPYDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	002BD05200010025 220C000611490004	..+R...%"....I..	..}.....
0010	00162110E2E3D3C5 C3F1404040404040	..!.....@@@...STLEC1
0020	4040404040400005 211502	@@@@@...!

	SQLCARD OBJDSS	(ASCII)	(EBCDIC)
	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
0000	000BD00300010005 2408FF\$..	..}.....

図 38. トレース出力の例 (TCP/IP 接続) (続き)

DRDA トレースの後続のバッファ情報

それ以降の送信バッファと受信バッファを分析して、追加情報を得ることができます。次の要求はコミットを含んでいます。 **commit** コマンドは、IBM メインフレーム・データベース・サーバー管理システムに現在の作業単位をコミットするよ

う命令します。4番目のバッファは、IBM メインフレーム・データベース・サーバーデータベース管理システムから、コミットまたはロールバックの結果として受け取られます。そこには最終作業単位の応答メッセージ (ENDUOWRM) が含まれ、それは現行の作業単位が終了したことを示します。

この例のトレース・エントリー 12 は、DDM コード点 X'2408' とそれに続く X'FF' が示しているとおおり、NULL の SQLCA を含んでいます。NULL の SQLCA (X'2408FF') は、成功 (SQLCODE 0) を示しています。

583 ページの図 35 は、トレース・エントリー 16 にエラー SQLCA を含んだ受信バッファの例を示しています。

JDBC トレース

DB2 Universal JDBC ドライバーを使用するアプリケーションのトレースの取得

このタスクでは、DB2 Universal JDBC ドライバーを使用するアプリケーションのトレースを取得する方法について説明します。

手順

DB2 Universal JDBC ドライバーを使用している SQLJ または JDBC アプリケーションがある場合、以下のいくつかの方法で JDBC トレースを使用できます。

- DataSource インターフェースを使用してデータ・ソースに接続する場合、DataSource.setTraceLevel() および DataSource.setTraceFile() メソッドを使用してトレースを使用可能にします。
- DriverManager インターフェースを使用してデータ・ソースに接続する場合、トレースを使用可能にする最も簡単な方法は、接続を取得する前に DriverManager に logWriter を設定することです。

例えば、

```
DriverManager.setLogWriter(new PrintWriter(new FileOutputStream("trace.txt")));
```

- DriverManager インターフェースを使用している場合、別の方法として、ドライバーのロード時に、traceFile と traceLevel プロパティを URL の一部として指定できます。

例えば、

```
String databaseURL =  
"jdbc:db2://hal:50000/sample:traceFile=c:/temp/trace.txt;" ;
```

CLI トレース・ファイル

CLI ドライバーにアクセスするアプリケーションは、CLI トレース・ユーティリティーを利用できます。このユーティリティーは、CLI ドライバーによって行われるすべての関数呼び出しをトレース・ファイルに記録するので、問題判別に役立ちます。

このタイプのトレースは、以下の場所で問題が発生する場合に有効です。

- CLI アプリケーション

- ODBC アプリケーション (ODBC アプリケーションが CLI インターフェースを使用して DB2 にアクセスした後)
- CLI ストアード・プロシージャ

ODBC アプリケーションを診断する際、ODBC トレースまたは CLI トレースを使用して問題を判別するのが最も簡単という場合は少なくありません。ODBC ドライバー・マネージャーを使用している場合、ODBC トレースを取るための機能が提供されている可能性があります。ODBC のトレースを使用可能にする方法を判別するには、ご使用のドライバー・マネージャーの資料を参照してください。CLI トレースは DB2 ソフトウェアに固有のもので、汎用の ODBC トレースよりも詳細な情報を含んでいる場合が少なくありません。通常、これらの両方のトレースは非常に似ています。両方ともアプリケーションからの CLI 呼び出しの項目と出口点をリストしており、これらの呼び出しに対するパラメーターと戻りコードも含まれています。

注: CLI トレースは、CLI ドライバーの内部的な働きに関する微小な情報を提供します。

このトピックでは、以下の内容について説明します。

- CLI トレース
- CLI トレース・オプションと db2cli.ini ファイル
- CLI ドライバー・トレースと ODBC ドライバー・マネージャー・トレース
- CLI ドライバーと DB2 トレース
- CLI トレースと CLI ストアード・プロシージャ

CLI for Linux, UNIX, and Windows には、包括的なトレース・ユーティリティが用意されています。デフォルトでは、トレース・ユーティリティは使用不可であるため、追加のコンピューター・リソースを使用しません。これが使用可能になると、アプリケーションが CLI ドライバーにアクセスするときに、トレース・ユーティリティが必ず 1 つ以上のテキスト・トレース・ファイルを生成します。これらのトレース・ファイルは次の情報を提供します。

- CLI 関数がアプリケーションによって呼び出された順序
- CLI 関数との間でやり取りされた入出力パラメーターの内容
- CLI 関数によって生成された戻りコード、エラー・メッセージ、および警告メッセージ

CLI トレース・ファイルを分析すると、アプリケーション開発者にとっていくつかの面で役立ちます。まず、プログラム・ロジックおよびパラメーター初期化に関する捕らえにくいエラーが、しばしばトレースで明確になります。2 番目に、CLI トレースから、アプリケーションやそのアクセス先のデータベースをより良くチューニングする方法が分かる場合があります。例えば、CLI トレースで特定の列セットで何度も照会されている表があることが示されている場合、アプリケーションのパフォーマンス向上のために、その表の列の 1 つに対応する索引を作成することもできます。最後に、CLI トレース・ファイルの分析は、ベンダーから入手したアプリケーションやインターフェースがどのように動作しているかをアプリケーション開発者が理解するのに役立ちます。

CLI トレース

CLI トレースを使用可能にするには、db2cli.ini 構成ファイルを使用して、**SQL_ATTR_TRACE** 環境属性をプログラムで設定する方法と、DB2 トレース・ユーティリティ (**db2trc** コマンド) を使用する方法があります。

通常、db2cli.ini ファイルの構成キーワードの値は、CLIドライバを使用するアプリケーションが初期化された場合にのみ読み取られます。db2cli.ini ファイルの場所は、次の要因に基づき変わることがあります。

- DB2 製品がインストールされているオペレーティング・システム。
- 使用しているデータ・ソース名 (DSN) のタイプ。
- インストールする DB2 製品のタイプ。
- **DB2CLIINIPATH** レジストリー変数を設定するかどうか。
- DB2 製品のインストール中に「DB2 共通アプリケーション・データ・パス」または **DB2_COMMON_APP_DATA_TOP_PATH** 応答ファイルのキーワードを設定したかどうか (Windows オペレーティング・システムでのみ利用可能)。

コマンド行プロセッサから db2cli.ini ファイルの構成キーワードの現在の設定を表示するには、次のコマンドを発行します。

```
db2 GET CLI CFG FOR SECTION COMMON
```

CLI トレース機能を構成するために db2cli.ini ファイルを変更するには、次の 2 つの方法があります。

- テキスト・エディターを使用して db2cli.ini ファイルを編集します。
- コマンド行プロセッサから **UPDATE CLI CFG** コマンドを発行します (コマンド行プロセッサが利用可能である場合)。例えば、コマンド行プロセッサから次のコマンドを発行すると、db2cli.ini ファイルが更新され、CLIトレースが使用可能になります。

```
DB2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 1
DB2 UPDATE CLI CFG FOR SECTION COMMON USING TracePathName c:\TMP\trace
DB2 UPDATE CLI CFG FOR SECTION COMMON USING TraceComm 1
DB2 UPDATE CLI CFG FOR SECTION COMMON USING TraceFlush 1
DB2 UPDATE CLI CFG FOR SECTION COMMON USING TraceTimestamp 1
```

通常、CLI トレース構成キーワードは、アプリケーションが初期化された場合にのみ db2cli.ini ファイルから読み取られます。ただし、特別な db2cli.ini ファイル・オプションである **TraceRefreshInterval** を使用して、特定の CLI トレース・キーワードの値が db2cli.ini ファイルから再読み取りされる間隔を指定することができます。

注:

1. TracePathName キーワードが使用される場合、指定されるパスは、CLI のトレースを開始する前から存在していなければならない、しかもグローバルな読み取り権限と書き込み権限がなければなりません。
2. IBM Data Server Driver Package と IBM Data Server Driver for ODBC and CLI のインストール済み環境には、DB2 コマンド行プロセッサは組み込まれていません。トレース構成キーワードの設定を変更するときには、db2cli.ini ファイルを手動で変更できます。

また、`SQLSetEnvAttr()` API を使用してアプリケーションから `SQL_ATTR_TRACE` 環境属性を設定することによって、CLI トレースを構成することもできます。この属性は、`db2cli.ini` ファイルの設定をオーバーライドします。

アプリケーションを再起動せずに CLI トレースを使用可能にするには、次のように `db2trc` コマンドに `-CLI` パラメーターを指定して使用します。

```
db2trc on -f <filename> -cli
```

DB2 トレース・ユーティリティーは、`<filename>` パラメーターで指定されているファイル名を使用してバイナリー・ダンプ・ファイルを作成します。このファイルフォーマットを設定して CLI トレースを生成できます。例えば、`-CLI` パラメーターを使用して DB2 トレースを使用可能にした後で、次のように `db2trc` コマンドに `off` パラメーターを指定して発行し、トレースをオフにします。

```
db2trc off
```

`db2trc` がオフになると、`<filename>` で指定された名前を持つバイナリー・ファイルが、トレース・コマンドが発行されたローカル・パスに生成されます。`db2trc` コマンドに `fmt` パラメーターと `-cli` パラメーターを指定して使用することにより、バイナリー・ファイルを CLI トレースにフォーマット設定します。

```
db2trc fmt -cli <filename> <cli-filename>
```

`<filename>` は、`db2trc` コマンドから生成されたバイナリー・ファイルの名前で、`<cli-filename>` は CLI トレース出力の名前です。

重要: CLI トレースは、必要なければ使用不可にします。不必要なトレースを行うと、アプリケーションのパフォーマンスが低下すると共に、不要なトレース・ファイルが生成される場合があります。CLI トレース・ファイルは削除されず、新しいトレース情報が新しいファイルに作成されるか、または既存のトレース・ファイルに追加されます。

CLI トレース・オプションと `db2cli.ini` ファイル

CLI ドライバーを使用するアプリケーションが実行を開始すると、ドライバーは `db2cli.ini` ファイルの `[COMMON]` セクションで CLI トレース・キーワードを確認します。

```
[COMMON]
Trace=1
TracePathName=c:\TMP\trace
TraceFlush=1
```

以下の CLI トレース・キーワードを定義できます。

- `com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008821.dita`
- `com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008822.dita`
- `com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0011527.dita`
- `com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008823.dita`
- `com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008824.dita`
- `com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0011528.dita`
- `com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008825.dita`
- `com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008826.dita`

- com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008827.dita
- com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008839.dita
- com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008828.dita
- com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008829.dita
- com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008840.dita
- com.ibm.db2.luw.admin.trb.doc/com.ibm.db2.luw.apdv.cli.doc/doc/r0008831.dita

使用上の注意:

1. CLI トレース・キーワードは db2cli.ini ファイルの **[COMMON]** セクションに存在するので、それらの値は CLI ドライバーを介したすべてのデータベース接続に適用されます。
2. CLI トレース・キーワードには大/小文字の区別がありません。ただし、パスおよびファイル名のキーワード値は、UNIX オペレーティング・システムなど、一部のオペレーティング・システムで大/小文字を区別する場合があります。
3. db2cli.ini ファイル内の CLI トレース・キーワードまたは関連する値のいずれかが無効である場合、CLI トレース・ユーティリティはその値を無視します。有効な CLI キーワードが無効な値を使用して指定されている場合、そのトレース・キーワードのデフォルト値が代わりに使用されます。
4. **TraceRefreshInterval** キーワードを設定する場合を除き、CLI トレース・キーワードは db2cli.ini ファイルから、アプリケーションの初期化時に 1 回だけ読み取られます。**TraceRefreshInterval** キーワードを設定した場合、**Trace** キーワードと **TracePIDList** キーワードが指定された間隔で db2cli.ini ファイルから再読み取りされ、適切であれば現在実行されているアプリケーションに適用されます。

CLI ドライバー・トレースと ODBC ドライバー・マネージャー・トレース

ODBC Driver Manager トレースと CLI ドライバー・トレースの違いを理解することは重要です。ODBC Driver Manager トレースは、ODBC アプリケーションが ODBC Driver Manager に対して行う ODBC 関数呼び出しを示します。ODBC Driver Manager は DB2 製品の一部ではなく、独立系ソフトウェア・ベンダーから個別に取得されます。対照的に、CLI ドライバー・トレースは、ODBC Driver Manager がアプリケーションに代わって CLI ドライバーに対して行う関数呼び出しを示します。

ODBC Driver Manager は、一部の関数呼び出しをアプリケーションから CLI ドライバーに直接転送することがあります。しかし、ODBC Driver Manager は、ドライバーへの一部の関数呼び出しの転送を遅らせたり回避したりすることもあります。さらに ODBC Driver Manager は、呼び出しを CLI ドライバーに転送する前に、アプリケーション関数引数を変更したり、アプリケーション関数を他の関数にマップしたりすることもあります。ODBC Driver Manager は、次のタスクを実行するために介入します。

- ODBC 3.0 で非推奨になった ODBC 2.0 の関数が、新しい関数にマップされる。
- ODBC 3.0 で非推奨になった ODBC 2.0 の関数引数が、同等の ODBC 3.0 引数にマップされる。

- Microsoft カーソル・ライブラリーが、SQLExtendedFetch() 関数などの各関数の呼び出しを、SQLFetch() 関数などのサポート対象となる各関数の複数の呼び出しにマップして、同じ結果を実現する。
- ODBC Driver Manager 接続プールが、通常は SQLDisconnect() 関数呼び出しの要求を延期する。接続が再利用される場合には要求をすべて回避する。

これらおよび他の理由により、ODBC Driver Manager トレースを、CLI ドライバー・トレースを補完するものとして活用できます。

ODBC Driver Manager トレースの取り込みと解釈については、独立系ソフトウェア・ベンダーによって提供される ODBC Driver Manager の資料を参照してください。Windows オペレーティング・システムでは、「Microsoft ODBC 3.0 Software Development Kit and Programmer's Reference」を参照してください。これはオンライン (<http://www.msdn.microsoft.com/>) で利用できます。

CLI ドライバーと DB2 トレース

CLI ドライバーは、処理を行うために多くの内部 DB2 関数を使用します。これらの内部 DB2 関数呼び出しは、DB2 トレースに記録されます。DB2 トレースは、IBM サービスによる問題判別と解決を支援するために用意されています。

CLI トレースと CLI ストアード・プロシージャ

すべてのワークステーション・プラットフォームで、CLI トレース・ユーティリティを使用して、CLI ストアード・プロシージャの実行をトレースできます。

これ以前のセクションに記載されている CLI トレース情報および指示のほとんどは一般化された内容であり、アプリケーションとストアード・プロシージャの両方に等しく当てはまります。しかし、通常、アプリケーションは、データベース・サーバーのクライアントとしてデータベース・サーバーとは別個のマシンで実行されますが、ストアード・プロシージャはデータベース・サーバーで実行されるという点で異なります。このため、CLI ストアード・プロシージャの実行をトレースする場合は、次の追加の手順を行う必要があります。

- トレース・キーワード・オプションを、DB2 サーバーの db2cli.ini ファイルに指定します。
- **db2start** コマンドを発行してデータベース・マネージャーを開始する前に、すべてのキーワードを適切に設定していることを確認します。
- CLI キーワードである **TRACEREFRESHINTERVAL** および **QUERYTIMEINTERVAL** の使用は、CLI API 呼び出しを使用するストアード・プロシージャまたはルーチンの内部で使用される場合は無視されます。

CLI トレースの取得

CLI トレースをオンにするには、CLI 構成キーワードのセットを使用可能にする必要があります。

始める前に

注: db2cli.ini ファイルに追加できるキーワードは数多くあります。こうしたキーワードは、アプリケーション動作に影響を及ぼす可能性があります。こうしたキーワードは、アプリケーション問題を解決することもあれば、その原因となる場合も

あります。CLI 資料では取り上げられていないキーワードもあります。そのようなキーワードは IBM ソフトウェア・サポートでのみ使用可能です。文書化されていないキーワードが db2cli.ini ファイルにある場合、おそらくそれは IBM ソフトウェア・サポート・チームの推奨によるものです。

このタスクについて

トレース機能を使ってアプリケーションの問題を診断する場合には、それによりアプリケーション・パフォーマンスに影響があり、テスト・アプリケーションだけではなくすべてのアプリケーションに影響を及ぼすことを覚えておいてください。このため、問題の識別後には忘れずにこの機能をオフにすることが大切です。

手順

CLI トレースを取得するには、以下を行います。

1. トレース・ファイルのパスを作成します。

すべてのユーザーが書き込み可能なパスを作成するのは重要なことです。例えば、Windows オペレーティング・システムでは次のようにします。

```
mkdir c:%temp%trace
```

Linux および UNIX オペレーティング・システムの場合:

```
mkdir /tmp/trace  
chmod 777 /tmp/trace
```

2. CLI 構成キーワードを更新します。

これは、db2cli.ini ファイルを手動で編集するか、あるいは **UPDATE CLI CFG** コマンドを使用することによって実行できます。

- db2cli.ini ファイルを手動で編集するには、以下を行います。
 - a. プレーン・テキスト・エディターで、db2cli.ini ファイルをオープンします。db2cli.ini ファイルの場所は、Microsoft ODBC Driver Manager が使用されているかどうか、使用されるデータ・ソース名 (DSN) のタイプ、インストールされているクライアントまたはドライバのタイプ、およびレジストリー変数 **DB2CLIINIPATH** が設定されているかどうかによって変わることがあります。詳しくは、「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『db2cli.ini 初期設定ファイル』のトピックを参照してください。
 - b. このファイルに以下のセクションを追加します (COMMON セクションが既に存在する場合には、変数の追加だけを行います)。

```
[COMMON]  
Trace=1  
TracePathName=path  
TraceComm=1  
TraceFlush=1  
TraceTimeStamp=1
```

ここで、*path* は、例えば Windows では C:%temp%trace、Linux および UNIX では /tmp/trace などとなります。

- c. ファイルの最後に空白行を少なくとも 1 行設けて、ファイルを保管します。(これで一部の構文解析エラーを回避できます。)

- CLI 構成キーワードを更新するために **UPDATE CLI CFG** コマンドを使用するには、以下のコマンドを実行してください。

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 1
db2 UPDATE CLI CFG FOR SECTION COMMON USING TracePathName path
db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceComm 1
db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceFlush 1
db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceTimeStamp 3
```

ここで、*path* は、例えば Windows では C:%temp%trace、Linux および UNIX では /tmp/trace などとなります。

3. db2cli.ini の構成を確認します。

次のコマンドを実行して、適切なキーワードが設定されて、取得されていることを検査します。

```
db2 GET CLI CFG FOR SECTION COMMON
```

4. アプリケーションを再開します。

db2cli.ini ファイルが読み取られるのは、アプリケーションの開始時に限られるので、変更を有効にするにはアプリケーションを再始動する必要があります。

CLI ストアード・プロシージャをトレースすると、DB2 インスタンスが再始動します。

5. エラーをキャプチャーします。

アプリケーションを実行し、エラーが生成されたら終了します。問題の再現に関連したアプリケーションだけをトレース時に実行するというように、この状態を絞り込むことができれば、トレース分析はより明確なものになります。

6. CLI トレースを使用不可にします。

db2cli.ini の [COMMON] セクションで **Trace** キーワードを手動でゼロの値に設定するか、以下のコマンドを実行します。

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 0
```

7. (オプション) 実行中およびトレース中の可能性があるアプリケーションがあれば、それを再始動してください。

タスクの結果

CLI トレース・ファイルは、**TracePathName** キーワードで指定されたパスに書き込まれます。ファイル名は、*ppidttid.cli* の形式になります。*pid* は、オペレーティング・システムが割り当てたプロセス ID、および *tid* は、アプリケーション処理によって生成されるスレッドごとの数値カウンター (0 から始まる) です。例えば、*p1234t1.cli* のようになります。IBM ソフトウェア・サポートを利用して問題を診断する場合、トレース・パスに生成されたファイルをすべて提供する必要があります。

CLI トレース・ファイル内の入出力パラメーターの解釈

通常の間数の場合と同様、DB2 コール・レベル・インターフェース (CLI) 関数には入出力パラメーターがあります。これらの入出力パラメーターが CLI トレースに現れることがあります。これにより、各アプリケーションが特定の CLI API を呼び出

す方法が詳細にわかります。CLI トレースに現れた CLI 関数の入出力パラメーターは、資料の CLI リファレンスの節にある CLI 関数の定義と対照できます。

以下は、CLI トレース・ファイルの断片です。

```
SQLConnect( hDbc=0:1, szDSN="sample", cbDSN=-3, szUID="",
            cbUID=-3, szAuthStr="", cbAuthStr=-3 )
----> Time elapsed - +6.960000E-004 seconds

SQLRETURN  SQLConnect      (
SQLHDBC      ConnectionHandle, /* hdbc */
SQLCHAR      *FAR ServerName, /* szDSN */
SQLSMALLINT  NameLength1,    /* cbDSN */
SQLCHAR      *FAR UserName,  /* szUID */
SQLSMALLINT  NameLength2,    /* cbUID */
SQLCHAR      *FAR Authentication, /* szAuthStr */
SQLSMALLINT  NameLength3);   /* cbAuthStr */
```

CLI 関数の最初の呼び出しには、入力パラメーターとそれに割り当てられた値が示されます (該当する場合)。

CLI 関数が戻されると、結果として生じた出力パラメーターが示されます。以下に例を示します。

```
SQLAllocStmt( phStmt=1:1 )
<--- SQL_SUCCESS Time elapsed - +4.444000E-003 seconds
```

このケースでは、CLI 関数 `SQLAllocStmt()` は、出力パラメーター `phStmt` と値 `"1:1"` (接続ハンドル 1、ステートメント・ハンドル 1) を戻しています。

CLI トレース内の動的 SQL の分析

`SQLPrepare()` および `SQLBindParameter()` でパラメーター・マーカを宣言および使用することにより、CLI トレースは動的 SQL がどのように実行されるかも示します。これにより、どの SQL ステートメントが実行されるかを実行時に判別できます。

-以下のトレース・エントリーは、SQL ステートメントの準備を示しています (疑問符 (?) または名前が続くコロン (:name) はパラメーター・マーカを示します)。

```
SQLPrepare( hStmt=1:1, pszSqlStr=
"select * from employee where empno = ?",
cbSqlStr=-3 )
----> Time elapsed - +1.648000E-003 seconds
( StmtOut="select * from employee where empno = ?" )
SQLPrepare( )
<--- SQL_SUCCESS Time elapsed - +5.929000E-003 seconds
```

以下のトレース・エントリーは、パラメーター・マーカを最大長 7 の CHAR としてバインディングする様子を示しています。

```
SQLBindParameter( hStmt=1:1, iPar=1, fParamType=SQL_PARAM_INPUT,
fCType=SQL_C_CHAR, fSQLType=SQL_CHAR, cbColDef=7, ibScale=0,
rgbValue=&00854f28, cbValueMax=7, pcbValue=&00858534 )
----> Time elapsed - +1.348000E-003 seconds
SQLBindParameter( )
<--- SQL_SUCCESS Time elapsed - +7.607000E-003 seconds
```

次いで動的 SQL ステートメントが実行されます。 `rgbValue="000010"` は、実行時にアプリケーションによってパラメーター・マーカから置き換えられた値を示しています。

```

SQLExecute( hStmt=1:1 )
----> Time elapsed - +1.317000E-003 seconds
( iPar=1, fCType=SQL_C_CHAR, rgbValue="000010" - X"30303030303130",
pcbValue=6, piIndicatorPtr=6 )
sqlccsend( ulBytes - 384 )
sqlccsend( Handle - 14437216 )
sqlccsend( ) - rc - 0, time elapsed - +1.915000E-003
sqlccrecv( )
sqlccrecv( ulBytes - 1053 ) - rc - 0, time elapsed - +8.808000E-003
SQLExecute( )
<--- SQL_SUCCESS Time elapsed - +2.213300E-002 seconds

```

CLI トレース内の時間情報の解釈

CLI トレースから時間情報を収集する方法がいくつかあります。デフォルトでは、CLI トレースは、特定のスレッドで最後の CLI API 呼び出しを実行してからアプリケーションで費やされた時間を収集します。

これには DB2 で費やされた時間だけでなく、クライアントとサーバーの間のネットワーク時間も含まれます。例えば、

```

SQLAllocStmt( hDbc=0:1, phStmt=&0012ee48 )
----> Time elapsed - +3.964187E+000 seconds

```

(この時間値は、最後の CLI API が呼び出されてからアプリケーションで費やされた時間を示しています)

```

SQLAllocStmt( phStmt=1:1 )
<--- SQL_SUCCESS Time elapsed - +4.444000E-003 seconds

```

(関数は完了しているので、この時間値は、ネットワーク時間を含め、DB2 で費やされた時間を示します)

時間情報を収集する別の方法は、CLI キーワード `TraceTimeStamp` を使用することです。このキーワードを使用すると、CLI API 呼び出しが呼び出されたり、結果が戻されたりするたびに、タイム・スタンプが生成されます。キーワードの表示オプションは 4 つあります。タイム・スタンプ情報なし、プロセッサ・ティックおよび ISO タイム・スタンプ、プロセッサ・ティック、ISO タイム・スタンプの 4 つです。

これは、「CLI0125E - 関数シーケンス・エラー」など、時間に関連した問題を扱う際にたいへん役立つ可能性があります。また、マルチスレッド・アプリケーションを扱う際に、どのイベントが最初に発生したかを判別するためにも役立つことがあります。

CLI トレース内の不明な値の解釈

CLI 関数が、CLI トレース内の入力パラメーターの値として "Unknown value" (不明な値) を戻すことがあります。

これが生じ得るのは、CLI ドライバーがその入力パラメーターに特有の値を予期しているのに、アプリケーションが別の値を提供している場合です。例えば、CLI 関数の旧式の定義に従っている場合や、推奨されない CLI 関数を使用している場合に、これが生じ得ます。

CLI 関数呼び出しが "Option value changed" または "Keyset Parser Return Code" を返すこともあります。これはキー・セット・カーソルがメッセージを表示することの結果として生じます。例えば、カーソルが何らかの理由で静的カーソルに格下げされる場合などです。

```
SQLExecDirect( hStmt=1:1, pszSqlStr="select * from org", cbSqlStr=-3 )
  ---> Time elapsed - +5.000000E-002 seconds
( StmtOut="select * from org" )
( COMMIT=0 )
( StmtOut=" SELECT A.TABSCHEMA, ..... )
( StmtOut=" SELECT A.TABSCHEMA, ..... )
( Keyset Parser Return Code=1100 )
```

```
SQLExecDirect( )
  <--- SQL_SUCCESS_WITH_INFO Time elapsed - +1.06E+001 seconds
```

この CLI トレースでは、キー・セット・パーサーが戻りコード 1100 を示しています。これは、表のユニーク索引または主キーがなく、したがってキー・セット・カーソルを作成できないことを示しています。これらの戻りコードは外部化されていないので、戻りコードの意味についてさらに情報が必要な場合、現時点では IBM ソフトウェア・サポートに連絡を取る必要があります。

SQLError または SQLDiagRec を呼び出すと、カーソル・タイプが変更されたことが示されます。アプリケーションは、どの属性が変更されたかを判別するために、カーソル・タイプと並行性を照会する必要があります。

マルチスレッド CLI トレース出力の解釈

CLI トレースは、マルチスレッド・アプリケーションをトレースできます。マルチスレッド・アプリケーションをトレースする最善の方法は、CLI キーワード TracePathName を使用することです。これにより、p<pid>t<tid>.cli という名前のトレース・ファイルが作成されます (<tid> は、アプリケーションの実際のスレッド ID)。

実際のスレッド ID を知る必要がある場合は、この情報を CLI トレースのヘッダーから調べることができます。

```
[ Process: 3500, Thread: 728 ]
[ Date & Time:          02/17/2006 04:28:02.238015 ]
[ Product:              QDB2/NT DB2 v9.1.0.190 ]
...
```

CLI キーワード TraceFileName を使用すれば、マルチスレッド・アプリケーションを 1 つのファイルにトレースできます。この方法を使うとユーザーの選んだ 1 つのファイルが生成されますが、読みにくくなるかもしれません。というのは、あるスレッドの API が別のスレッドの別の API と同時に実行されることがあり、トレースを検査するときに混同してしまう恐れがあるためです。

通常は TraceTimeStamp をオンにするようお勧めします。そうすれば、特定の API の実行された時刻を見て、イベントの本当の順序を判別することができます。これは、1 つのスレッドが別のスレッドの中で問題を引き起こすといった問題を調査する際にたいへん役立ちます (例えば、CLI0125E - 関数シーケンス・エラーなど)。

db2cli.ini ファイルを使用して作成された CLI トレース・ファイルの解析

CLI ドライバーにアクセスするアプリケーションは、CLI トレース・ユーティリティーを利用できます。このユーティリティーは、CLI ドライバーによって行われるすべての関数呼び出しをトレース・ファイルに記録するので、問題判別に役立ちます。

このトピックでは、トレース・ユーティリティーにより生成されるトレース・ファイルにアクセスして解釈する方法について説明します。

- CLI トレース・ファイルのロケーション
- CLI トレース・ファイルの解釈

CLI トレース・ファイルのロケーション

TraceFileName キーワードが db2cli.ini ファイルで完全修飾ファイル名を指定するために使用されている場合、CLI トレース・ファイルは指定されたロケーションに生成されます。CLI トレース・ファイル名として相対ファイル名が指定されている場合、そのファイルのロケーションは、オペレーティング・システムが認識しているアプリケーションの現行パスに応じて異なります。

注: アプリケーションを実行するユーザーに、指定のパスのトレース・ファイルに書き込む十分な権限がない場合、ファイルは生成されず、警告やエラーも出されません。

CLI トレース・ユーティリティーは、**TracePathName** キーワードが設定されている場合、アプリケーションのプロセス ID とスレッド・シーケンス番号を自動的に使用してトレース・ファイルの名前を付けます。例えば、3 つのスレッドがあるアプリケーションの CLI トレースによって、p1680t3124.cli、p1680t3125.cli、および p1680t3126.cli という CLI トレース・ファイルが生成される可能性があります。

CLI トレース出力ファイルが作成されていないように思える場合、次のようにします。

- トレース構成キーワードが db2cli.ini ファイルで正しく設定されていることを確認します。コマンド行プロセッサから db2 GET CLI CFG FOR SECTION COMMON コマンドを発行すれば、簡単にこのことを実行できます。
- db2cli.ini ファイルを更新した後、アプリケーションが再開されたことを確認します。CLI トレース・ユーティリティーは、アプリケーションの開始中に初期化されます。CLI トレース・ユーティリティーは、**TraceRefreshInterval** キーワードがアプリケーションの開始前に適切に指定されている場合に限り、実行時に再構成できます。

注: 実行時に再構成できる CLI キーワードは、**Trace** と **TracePIDList** だけです。他の CLI キーワード (**TraceRefreshInterval** を含む) を変更しても、アプリケーションを再始動しなければ反映されません。

- **TraceRefreshInterval** キーワードが、アプリケーションの開始前に指定されていた場合で、しかも **Trace** キーワードが 0 に初期設定されていた場合は、CLI トレース機能が **Trace** キーワードの値を再読み取りするために十分な時間が経過していることを確認します。

- **TracePathName** キーワードを使用してトレース・ディレクトリーを指定する場合には、アプリケーションの開始前に指定ディレクトリーが存在していなければなりません。
- アプリケーションに、指定されたトレース・ファイルまたはトレース・ディレクトリーに対する書き込みアクセス権があることを確認します。
- **DB2CLIINIPATH** 環境変数を調べます。これが設定されている場合、CLI トレース機能は、db2cli.ini ファイルがこの変数で指定されたロケーションにあると想定します。
- DB2 製品のインストール中に「DB2 共通アプリケーション・データ・パス」または **DB2_COMMON_APP_DATA_TOP_PATH** 応答ファイル・キーワードが指定された場合、**DB2_COMMON_APP_DATA_PATH** 環境変数で指定されたパスを確認します (Windows オペレーティング・システムでのみ利用可能)。
- アプリケーションが CLI ドライバーとのインターフェースに ODBC を使用している場合、SQLConnect()、SQLDriverConnect()、または SQLBrowseConnect() のいずれかの関数が正常に呼び出されていることを確認します。データベース接続が正常に行われるまで、エントリーは CLI トレース・ファイルに書き込まれません。

CLI トレース・ファイルの解釈

CLI トレースの先頭には常に、トレースを生成したアプリケーションのプロセス ID とスレッド ID、トレースの開始時刻、および製品固有の情報 (ローカル DB2 のビルド・レベルや CLI ドライバーのバージョンなど) を示すヘッダーがあります。例えば、

```

1 [ Process: 1356, Thread: 5656 ]
2 [ Date & Time:           08/05/2011 15:35:00.326001 ]
3 [ Product:              QDB2/NT DB2 v9.7.400.501 ]
4 [ Level Identifier:     08050107 ]
5 [ CLI Driver Version:   09.02.0000 ]
6 [ Informational Tokens: "DB2 v9.7.400.501","s110330","IP23237","Fixpack 4" ]
7 [ Install Path:        C:%PROGRAMFILES%IBM\SQLLIB ]
8 [ db2cli.ini Location2: C:%Documents and Settings%All Users%Application Data%IBM\DB2\COPY1\cfg%
db2cli.ini ]
9 [ CLI Driver Type:     IBM Data Server Runtime Client ]

```

注: このセクションで使用するトレースの例では、トレースの左側に行番号を追加しています。これらの行番号は説明の助けとして追加したもので、実際の CLI トレースにはありません。

トレース・ヘッダーのすぐ次には、通常、環境と接続ハンドルの割り振りと初期化に関連したいくつかのトレース項目があります。例えば、

```

10 SQLAllocEnv( phEnv=&bffff684 )
11    —> Time elapsed - +9.200000E-004 seconds

12 SQLAllocEnv( phEnv=0:1 )
13    <— SQL_SUCCESS   Time elapsed - +7.500000E-004 seconds

14 SQLAllocConnect( hEnv=0:1, phDbc=&bffff680 )
15    —> Time elapsed - +2.334000E-003 seconds

16 SQLAllocConnect( phDbc=0:1 )
17    <— SQL_SUCCESS   Time elapsed - +5.280000E-004 seconds

18 SQLSetConnectOption( hDbc=0:1, fOption=SQL_ATTR_AUTOCOMMIT, vParam=0 )
19    —> Time elapsed - +2.301000E-003 seconds

20 SQLSetConnectOption( )

```

```

21    <— SQL_SUCCESS   Time elapsed - +3.150000E-004 seconds
22  SQLConnect( hDbc=0:1, szDSN="SAMPLE", cbDSN=-3, szUID="", cbUID=-3,
                szAuthStr="", cbAuthStr=-3 )
23    —> Time elapsed - +7.000000E-005 seconds
24  ( DBMS NAME="DB2/LINUX", Version="07.01.0000", Fixpack="0x22010105" )

25  SQLConnect( )
26    <— SQL_SUCCESS   Time elapsed - +5.209880E-001 seconds
27  ( DSN=""SAMPLE" )

28  ( UID="" )

29  ( PWD=""* )

```

トレース例に、CLI 関数呼び出しごとに 2 つの項目があることに注目してください (例えば、22 行から 24 行と 25 行から 29 行に SQLConnect() 関数呼び出しに対する項目があります)。これは CLI トレースの場合、常に当てはまります。最初の項目は関数呼び出しに渡された入力パラメーター値を示し、2 番目の項目は関数の出力パラメーター値とアプリケーションへの戻りコードを示します。

トレース例では、SQLAllocEnv() 関数は 12 行目で環境ハンドル (phEnv=0:1) を正常に割り振ったことが示されています。次いでそのハンドルは、SQLAllocConnect() 関数に渡され、16 行目で正常にデータベース接続ハンドル (phDbc=0:1) を割り振っています。次に 18 行目で、phDbc=0:1 接続の SQL_ATTR_AUTOCOMMIT 属性を SQL_AUTOCOMMIT_OFF (vParam=0) に設定するために SQLSetConnectOption() 関数が使用されています。最後に 22 行目で SQLConnect() が呼び出され、ターゲット・データベース (SAMPLE) に接続しています。

24 行目の SQLConnect() 関数の入力トレース項目に含まれているのは、ターゲット・データベース・サーバーのビルドおよびフィックスパックのレベルです。このトレース項目に表れている他の情報には、入力接続ストリング・キーワードおよびクライアントとサーバーのコード・ページが含まれます。例えば、次の情報も SQLConnect() トレース項目にあったとします。

```

( Application Codepage=819, Database Codepage=819,
  Char Send/Recv Codepage=819, Graphic Send/Recv Codepage=819,
  Application Char Codepage=819, Application Graphic Codepage=819 )

```

これは、アプリケーションとデータベース・サーバーが同じコード・ページ (819) を使用していることを意味しています。

SQLConnect() 関数の戻りトレース項目には、重要な接続情報 (トレース例では 27 から 29 行目) も含まれています。戻り項目に表示される可能性がある追加情報には、接続に適用されるすべての **PATCH1** キーワードまたは **PATCH2** キーワードの値が含まれます。例えば、PATCH2=27,28 が db2cli.ini ファイルの COMMON セクションに指定されている場合、次の行も SQLConnect() 戻り項目に示されます。

```

( PATCH2="27,28" )

```

環境と接続に関連したトレース項目の次は、ステートメント関連のトレース項目です。例えば、

```

30  SQLAllocStmt( hDbc=0:1, phStmt=&bffff684 )
31    —> Time elapsed - +1.868000E-003 seconds

32  SQLAllocStmt( phStmt=1:1 )

```

```

33    <— SQL_SUCCESS   Time elapsed - +6.890000E-004 seconds
34  SQLExecDirect( hStmt=1:1, pszSqlStr="CREATE TABLE GREETING (MSG
                               VARCHAR(10))", cbSqlStr=-3 )
35    —> Time elapsed - +2.863000E-003 seconds
36  ( StmtOut="CREATE TABLE GREETING (MSG VARCHAR(10))" )

37  SQLExecDirect( )
38    <— SQL_SUCCESS   Time elapsed - +2.387800E-002 seconds

```

トレース例では、32 行目で、データベース接続ハンドル (phDbc=0:1) が使用されてステートメント・ハンドル (phStmt=1:1) が割り振られています。その後 34 行目で、準備されていない SQL ステートメントがそのステートメント・ハンドルで実行されています。TraceComm=1 キーワードが db2cli.ini ファイルで設定されている場合、SQLExecDirect() 関数呼び出しのトレース項目に次のような追加のクライアント/サーバー通信情報が見られます。

```

SQLExecDirect( hStmt=1:1, pszSqlStr="CREATE TABLE GREETING (MSG
                               VARCHAR(10))", cbSqlStr=-3 )
    —> Time elapsed - +2.876000E-003 seconds
( StmtOut="CREATE TABLE GREETING (MSG VARCHAR(10))" )

    sqlccsend( ulBytes - 232 )
    sqlccsend( Handle - 1084869448 )
    sqlccsend( ) - rc - 0, time elapsed - +1.150000E-004
    sqlccrecv( )
    sqlccrecv( ulBytes - 163 ) - rc - 0, time elapsed - +2.243800E-002

SQLExecDirect( )
    <— SQL_SUCCESS   Time elapsed - +2.384900E-002 seconds

```

このトレース項目にある sqlccsend() 関数呼び出しおよび sqlccrecv() 関数呼び出しの追加情報に注目してください。sqlccsend() 呼び出し情報には、クライアントからサーバーに送られたデータ量、伝送にかかった時間、およびその伝送が成功した (0 = SQL_SUCCESS) ことが示されています。次いで sqlccrecv() 呼び出し情報には、クライアントがサーバーからの応答を待った時間と応答に含まれていたデータ量が示されています。

しばしば、CLI トレースには複数のステートメント・ハンドルがあります。ステートメント・ハンドル ID に十分注意することにより、トレースに示されている他のすべてのステートメント・ハンドルの中から、特定のステートメント・ハンドルの実行パスを容易にたどることができます。

CLI トレースに表示されるステートメントの実行パスは、通常、トレース例より複雑です。例えば、

```

39  SQLAllocStmt( hDbc=0:1, phStmt=&bffff684 )
40    —> Time elapsed - +1.532000E-003 seconds

41  SQLAllocStmt( phStmt=1:2 )
42    <— SQL_SUCCESS   Time elapsed - +6.820000E-004 seconds

43  SQLPrepare( hStmt=1:2, pszSqlStr="INSERT INTO GREETING VALUES ( ? )",
               cbSqlStr=-3 )
44    —> Time elapsed - +2.733000E-003 seconds
45  ( StmtOut="INSERT INTO GREETING VALUES ( ? )" )

46  SQLPrepare( )
47    <— SQL_SUCCESS   Time elapsed - +9.150000E-004 seconds

48  SQLBindParameter( hStmt=1:2, iPar=1, fParamType=SQL_PARAM_INPUT,

```

```

                                fCType=SQL_C_CHAR, fSQLType=SQL_CHAR, cbColDef=14,
                                ibScale=0, rgbValue=&080eca70, cbValueMax=15,
                                pcbValue=&080eca4c )
49      —> Time elapsed - +4.091000E-003 seconds

50  SQLBindParameter( )
51      <— SQL_SUCCESS   Time elapsed - +6.780000E-004 seconds

52  SQLExecute( hStmt=1:2 )
53      —> Time elapsed - +1.337000E-003 seconds
54  ( iPar=1, fCType=SQL_C_CHAR, rgbValue="Hello World!!!", pcbValue=14,
      piIndicatorPtr=14 )

55  SQLExecute( )
56      <— SQL_ERROR    Time elapsed - +5.951000E-003 seconds

```

トレース例の 41 行目で、データベース接続ハンドル (phDbc=0:1) が使用されて 2 番目のステートメント・ハンドル (phStmt=1:2) が割り振られています。その後 43 行目で、1 つのパラメーター・マーカのある SQL ステートメントがそのステートメント・ハンドルで準備されています。次に 48 行目で、該当する SQL タイプ (SQL_CHAR) の入力パラメーター (iPar=1) がそのパラメーター・マーカにバインドされています。最後に、52 行目でステートメントが実行されています。54 行目のトレースで入力パラメーターの内容と長さ (rgbValue="Hello World!!!", pcbValue=14) の両方が表示されていることに注目してください。

52 行目で SQLExecute() 関数が失敗しています。アプリケーションが、SQLError() のような診断 CLI 関数を呼び出して失敗の原因を診断する場合、その原因がトレースに表示されます。例えば、

```

57  SQLError( hEnv=0:1, hDbc=0:1, hStmt=1:2, pszSqlState=&bffff680,
              pfNativeError=&bffffee78, pszErrorMsg=&bffff280,
              cbErrorMsgMax=1024, pcbErrorMsg=&bffffee76 )
58      —> Time elapsed - +1.512000E-003 seconds

59  SQLError( pszSqlState="22001", pfNativeError=-302, pszErrorMsg="[IBM][CLI
              Driver][DB2/LINUX] SQL0302N The value of a host variable in the EXECUTE
              or OPEN statement is too large for its corresponding use.
              SQLSTATE=22001", pcbErrorMsg=157 )
60      <— SQL_SUCCESS   Time elapsed - +8.060000E-004 seconds

```

59 行目で返されるエラー・メッセージには、生成された DB2 のネイティブ・エラー・コード (SQL0302N)、そのコードに対応する sqlstate (SQLSTATE=22001)、およびエラーの要旨が含まれています。この例では、エラーの原因は明らかです。52 行目で、34 行目で VARCHAR(10) として定義された列にアプリケーションが 14 文字の文字列を挿入しようとしています。

アプリケーションが SQLError() のような診断関数を呼び出して CLI 関数の警告やエラーの戻りコードに回答しなかったとしても、警告メッセージまたはエラー・メッセージは CLI トレースに書き込まれます。ただし、トレースでのそのメッセージの位置は、実際にエラーが発生した場所とは離れている可能性があります。さらに、エラー・メッセージまたは警告メッセージがアプリケーションにより検索されなかったことがトレースに示されます。例えば、例のエラー・メッセージは検索されなかった場合、次のように、さらに後の関連のないと思われる CLI 関数呼び出しになるまで出力されない可能性があります。

```

SQLDisconnect( hDbc=0:1 )
      —> Time elapsed - +1.501000E-003 seconds
sqlccsend( ulBytes - 72 )

```

```

    sqlccsend( Handle - 1084869448 )
    sqlccsend( ) - rc - 0, time elapsed - +1.080000E-004
    sqlccrecv( )
    sqlccrecv( ulBytes - 27 ) - rc - 0, time elapsed - +1.717950E-001
( Unretrieved error message="SQL0302N The value of a host variable in the
EXECUTE or OPEN statement is too large for its corresponding use.
SQLSTATE=22001" )

SQLDisconnect( )
<— SQL_SUCCESS Time elapsed - +1.734130E-001 seconds

CLI トレースの最後の部分は、アプリケーションがトレースの前の部分で割り振ったデータベース接続や環境ハンドルを解放する操作を示すはずでず。例えば、

58 SQLTransact( hEnv=0:1, hDbc=0:1, fType=SQL_ROLLBACK )
59   → Time elapsed - +6.085000E-003 seconds
60 ( ROLLBACK=0 )

61 SQLTransact( )
   <— SQL_SUCCESS Time elapsed - +2.220750E-001 seconds

62 SQLDisconnect( hDbc=0:1 )
63   → Time elapsed - +1.511000E-003 seconds

64 SQLDisconnect( )
65   <— SQL_SUCCESS Time elapsed - +1.531340E-001 seconds

66 SQLFreeConnect( hDbc=0:1 )
67   → Time elapsed - +2.389000E-003 seconds

68 SQLFreeConnect( )
69   <— SQL_SUCCESS Time elapsed - +3.140000E-004 seconds

70 SQLFreeEnv( hEnv=0:1 )
71   → Time elapsed - +1.129000E-003 seconds

72 SQLFreeEnv( )
73   <— SQL_SUCCESS Time elapsed - +2.870000E-004 seconds

```

プラットフォーム固有のツール

診断ツール (Windows)

Windows システムの 3 つの便利な診断ツールについて説明します。

Windows オペレーティング・システムでは、以下の診断ツールが使用可能です。

イベント・ビューアー、パフォーマンス・モニターおよび他の管理ツール

「管理ツール」フォルダーには、イベント・ログへのアクセスおよびパフォーマンス情報へのアクセスを含む、さまざまな診断情報が提供されています。

タスク マネージャ

「タスク マネージャ」には、Windows サーバーで実行されているすべてのプロセスと、メモリー使用の詳細が表示されます。どの DB2 プロセスが実行されているかを確認し、パフォーマンス上の問題を診断するには、このツールを使用します。このツールを使用すると、メモリー使用量、メモリーの限界、使用されているスワッパー・スペース、およびプロセスのメモリー漏えいを判別できます。

「タスク マネージャ」をオープンするには、Ctrl + Alt + Delete を押し、選択可能なオプションから「タスク マネージャ」をクリックします。

Watson™ 博士

一般保護障害 (GPF) が発生した場合は、ワトソン博士ユーティリティーが呼び出されます。これは、問題の診断に役立つ可能性のあるデータをログに記録し、この情報をファイルに保管します。このユーティリティーを開始するには、コマンド行に `drwatson` と入力します。

診断ツール (Linux および UNIX)

このセクションでは、Linux および UNIX プラットフォームにおける、トラブルシューティングおよびパフォーマンス・モニターのために必要ないくつかのコマンドについて説明しています。

これらのコマンドの詳細は、コマンド行でコマンドの前に、「man」と入力すると表示されます。システムで、発生している問題の原因を識別するのに役立つデータを収集および処理するには、これらのコマンドを使用します。いったん収集したデータは、問題に詳しい担当者が調べることもできますし、要請に応じて IBM ソフトウェア・サポートに送ることもできます。

トラブルシューティング・コマンド (AIX)

以下の AIX システム・コマンドが、DB2 のトラブルシューティングに役立ちます。

errpt `errpt` コマンドは、ハードウェア・エラーおよびネットワーク障害などのシステム・エラーを報告します。

- 各エラーごとに 1 行の概要を表示するには、**errpt** を使用します。
- 各エラーごとに 1 ページの詳細ビューを表示するには、**errpt -a** を使用します。
- エラー番号が "1581762B" のエラーには、**errpt -a -j 1581762B** を使用します。
- 過去にページング・スペースを使い尽くしたかどうかを確認するには、**errpt | grep SYSVMM** を使用します。
- トークンリング・カードまたはディスクに問題があるかどうかを確認するには、"disk" および "tr0" 句を `errpt` の出力で確認します。

lsps `lsps -a` コマンドは、ページング・スペースの使用状況をモニターおよび表示します。

lsattr このコマンドは、さまざまなオペレーティング・システムのパラメーターを表示します。例えば、以下のコマンドを使用してデータベース・パーティション上の実際のメモリーの量を確認します。

```
lsattr -l sys0 -E
```

xmperf

Motif を使用する AIX システムでは、このコマンドはシステム関連のパフォーマンス・データを収集および表示する、グラフィカル・モニターを開始します。このモニターでは、各データベース・パーティションごとの 3 デイメンションのダイアグラムが単一のウィンドウに表示されるので、高水準

なモニターに役立ちます。しかし、アクティビティーが低い場合、このモニターからの出力は限定された値になります。

spmon 並列システム・サポート・プログラム (PSSP) の一部としてシステム・パーティションを使用するときには、SP スイッチがすべてのワークステーションで実行されていることを確認しなければならない場合があります。すべてのデータベース・パーティションの状況を表示するには、コントロール・ワークステーションから、以下のコマンドのいずれかを使用します。

- **spmon -d** ASCII 出力用
- **spmon -g** グラフィカル・ユーザー・インターフェース用

別の方法では、スイッチがダウンしているかを調べるのに、データベース・パーティション・ワークステーションから **netstat -i** コマンドを使用します。スイッチがダウンしている場合は、データベース・パーティション名の横にアスタリスク (*) が表示されます。例えば、

```
css0* 65520 <Link>0.0.0.0.0.0
```

スイッチがオンの場合は、アスタリスクは表示されません。

トラブルシューティング・コマンド (Linux および UNIX)

以下のシステム・コマンドは、指定されていない場合は Linux、および AIX を含むすべての UNIX システム用です。

df **df** コマンドでは、ファイル・システムが満杯かどうかを表示できます。

- すべてのファイル・システム (マウントされたものを含む) に、どれだけのフリー・スペースがあるかを表示するには、**df** を使用します。
- 名前に "dev" を含むすべてのファイル・システムに、どれだけのフリー・スペースがあるかを表示するには、**df | grep dev** を使用します。
- ホーム・ファイル・システムに、どれだけのフリー・スペースがあるかを表示するには、**df /home** を使用します。
- ファイル・システム "tmp" に、どれだけのフリー・スペースがあるかを表示するには、**df /tmp** を使用します。
- マシンに十分なフリー・スペースがあるかどうかを確認するには、以下のコマンドからの出力を確認します。**df /usr**、**df /var**、**df /tmp**、および **df /home**

truss このコマンドは、1 つ以上のプロセスでのシステム呼び出しのトレースに便利です。

pstack Solaris 2.5.1 以上で使用可能です。**/usr/proc/bin/pstack** コマンドはスタック・トレースバック情報を表示します。**/usr/proc/bin** ディレクトリーには、中断されていると思われるプロセスをデバッグする、その他のツールが含まれています。

パフォーマンス・モニター・ツール

以下のツールは、システムのパフォーマンスをモニターするために使用できます。

vmstat このコマンドは、何かがか中断されているのか、またはただ時間がかかっているのかを判別するのに役立ちます。ページイン (pi) およびページアウト

(po) 列に検出されるページング率をモニターすることができます。その他の重要な列は、割り振られた仮想記憶域 (avm) および、空いている仮想記憶域 (fre) の量です。

iostat このコマンドは、I/O アクティビティのモニターに役立ちます。読み取りおよび書き込み率を使用して、特定の SQL 操作に必要な時間を見積もれます。(それらがシステムでの唯一のアクティビティの場合)

netstat このコマンドでは、各データベース・パーティション上のネットワーク・トラフィックと、検出されたエラー・パケットの数を知ることができます。これは、ネットワークの問題を切り分けるのに役立ちます。

system file

Solaris オペレーティング・システムで使用可能です。/etc/system ファイルには、一度にシステムで許可される最大ユーザー数、ユーザーごとの最大プロセス数、およびリソースのサイズおよび数に関するプロセス間通信 (IPC) 制限などの、カーネル構成限界の定義が含まれています。これらの限界は、Solaris オペレーティング・システム・マシンでの DB2 のパフォーマンスに影響を与えるため、重要です。

第 6 章 DB2 データベースのトラブルシューティング

一般にトラブルシューティング・プロセスには、問題を切り分け、識別してから、解決方法を探ることが求められます。このセクションでは、DB2 製品の特定のフィーチャーに関連したトラブルシューティング情報を提供します。

共通問題が識別されるにつれて、検出された事項がチェックリストの形でこのセクションに追加されていきます。チェックリストでは解決方法に到達できない場合は、追加の診断データを収集してご自分で分析し、そのデータを IBM ソフトウェア・サポートに分析用に提出することができます。

以下の質問により、適切なトラブルシューティング・タスクに誘導されます。

1. 既知のすべてのフィックスパックを適用しましたか? まだ適用していない場合、797 ページの『フィックスの取得』を考慮してください。
2. 問題が発生するのは以下の場合ですか?
 - DB2 データベース・サーバーまたはクライアントのインストール中。その場合は、631 ページの『インストール問題についてのデータの収集』を参照してください。
 - インスタンスまたは DB2 Administration Server (DAS) の作成、ドロップ、更新、またはアップグレード中。その場合は、609 ページの『DAS およびインスタンス管理の問題についてのデータの収集』を参照してください。
 - EXPORT、IMPORT、LOAD、または db2move コマンドを使用してデータを移動中。その場合は、608 ページの『データ移動問題についてのデータの収集』を参照してください。

問題がこれらのカテゴリーのいずれにも属さなくても IBM ソフトウェア・サポートに連絡を取る場合には基本的な診断データが必要になることがあります。詳しくは、『DB2 についてのデータの収集』を参照してください。

DB2 についてのデータの収集

時には、単に症状をトラブルシューティングするだけでは問題を解決できない場合があります。そのような場合、診断データを収集する必要があります。収集する必要がある診断データと、そのデータの収集元のソースは、調査中の問題のタイプにより異なります。以下の手順は、問題を IBM ソフトウェア・サポートに送信する際に通常提供する必要がある情報の基本セットを収集する方法を示しています。

始める前に

完全な出力を得るには、インスタンス所有者が **db2support** ユーティリティを呼び出す必要があります。

手順

圧縮ファイル・アーカイブ内の診断情報の基本セットを収集するには、次のように **db2support** コマンドを入力します。

```
db2support output_directory -s -d database_name -c
```

-s を使用すると、使用中のハードウェアとオペレーティング・システムに関するシステムの詳細が得られます。**-d** を使用すると、指定されたデータベースに関する詳細が得られます。**-c** を使用すると、指定されたデータベースに対する接続試行が可能になります。

出力は適切に編成されて ZIP アーカイブ db2support.zip に圧縮されるため、任意のシステムに転送して簡単に解凍することができます。

次のタスク

特定の症状、または製品の特定の部分の問題については、追加データを収集する必要があるかもしれません。問題別の『データの収集』文書を参照してください。

次に以下のいずれかのタスクを行うことができます。

- データを分析する
- IBM ソフトウェア・サポートにデータを送信する

データ移動問題についてのデータの収集

データ移動コマンドの実行中に問題が発生し、その問題の原因が判別できない場合は、問題を診断して解決するために、あなた自身または IBM ソフトウェア・サポートで使用できる診断データを収集してください。

以下のリストから、発生している事情に該当するデータ収集の指示に従ってください。

- **db2move** コマンドに関連した問題のデータを収集するには、コマンドを発行したディレクトリーに移動してください。コマンドで指定したアクションに応じて、以下のファイルを見つけます。
 - COPY アクションの場合、COPY.timestamp.ERR および COPYSCHEMA.timestamp.MSG というファイルを探します。さらに LOAD_ONLY または DDL_AND_LOAD モードのいずれかを指定した場合、LOADTABLE.timestamp.MSG というファイルも探してください。
 - EXPORT アクションの場合、EXPORT.out というファイルを探してください。
 - IMPORT アクションの場合、IMPORT.out というファイルを探してください。
 - LOAD アクションの場合、LOAD.out というファイルを探してください。
- **EXPORT**、**IMPORT**、または **LOAD** コマンドに関連した問題のデータを収集するには、コマンドに MESSAGES パラメーターが含まれているかどうかを判別します。含まれている場合、出力ファイルを収集します。これらのユーティリティーは、現行ディレクトリーおよびデフォルト・ドライブ以外を指定しない場合にはそれらを宛先として使用します。
- **REDISTRIBUTE** コマンドに関連した問題のデータを収集するには、Linux および UNIX オペレーティング・システムの場合には「*databasename.database_partition_groupname.timestamp*」、Windows オペレーティング・システムの場合には「*databasename.database_partition_groupname.date.time*」というファイルを探してください。それは、Linux および UNIX オペレーティング・システムの場合には `$HOME/sqllib/db2dump` ディレクトリー、Windows オペレーティング・システム

の場合には \$DB2PATH¥sqllib¥redist ディレクトリーに置かれています。\$HOME はインスタンス所有者のホーム・ディレクトリーです。

DAS およびインスタンス管理の問題についてのデータの収集

DB2 Administration Server (DAS) またはインスタンス管理を実行中に問題が発生しているものの、問題の原因が判別できない場合、問題を診断して解決するために、あなた自身または IBM ソフトウェア・サポートが使用できる診断データを収集してください。

重要: DB2 Administration Server (DAS) は、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。DAS は、DB2 pureScale環境ではサポートされていません。リモート管理のためには、Secure Shell プロトコルを使用するソフトウェア・プログラムを使用してください。詳しくは、『DB2 Administration Server (DAS) が推奨されなくなった』() を参照してください。

以下の手順は、問題を再現でき、Linux または UNIX 上で DB2 を使用している場合のみ、使用できます。

DAS またはインスタンス管理問題の診断データを収集するには、以下のようになります。

1. トレースまたはデバッグ・モードを有効にして、失敗したコマンドを繰り返します。コマンドの例を以下に示します。

```
db2setup -t trace.out
dascrt -u DASUSER -d
dasdrop -d
dasmigr -d
dasupdt -d
db2icrt -d INSTNAME
db2idrop INSTNAME -d
db2iupgrade -d INSTNAME
db2iupdt -d INSTNAME
```

2. 診断ファイルを見つけます。複数のファイルが存在する可能性があるため、該当するすべてのファイルを実際に取得するためにタイム・スタンプを比較してください。

デフォルトでは、出力は /tmp ディレクトリーに出力されます。

ファイル名の例として、dascrt.log、dasdrop.log、dasupdt.log、db2icrt.log.PID、db2idrop.log.PID、db2iupgrade.log.PID、db2iupdt.log.PID などがあります。ここで、PID はプロセス ID です。

3. 診断ファイルを IBM ソフトウェア・サポートに提出します。

問題が **db2start** または START DATABASE MANAGER コマンドの失敗である場合、insthome/sqllib/log ディレクトリーで (insthome はインスタンス所有者のホーム・ディレクトリー) db2start.timestamp.log という名前のファイルを探してください。同様に、問題が **db2stop** または STOP DATABASE MANAGER コマンドの失敗である場合、db2stop.timestamp.log という名前のファイルを探してください。これらのファイルは、**start_stop_time** データベース・マネージャー構成パラメーターで指定した長さの時間内にデータベース・マネージャーがコマンドに応答しなかった場合にのみ、出力されます。

特定のパフォーマンス上の問題に関する診断データの収集

パフォーマンス上の問題について、できるだけ特定の問題に直接関係した診断データを収集することにより、パフォーマンスに関連する診断データをフルセットで収集する場合の影響を回避できます。これは、リソースの制約を既に受けているシステムで特に重要です。特定の診断データの収集対象にできるパフォーマンス上の問題は、プロセッサの使用量、メモリー、またはデータベース接続に関連するものでなければなりません。

これらの問題に関する診断データの収集は、問題の症状が 1 つ以上存在することが観察されたらすぐに開始できます。問題の診断に必要なデータを収集するには、**db2fodc** コマンドに、**-connections**、**-cpu**、または **-memory** いずれかのパラメーターを指定して使用します。

ときどきしか発生しない問題の場合は、特定の問題の状態を検出する **db2fodc -detect** コマンドを使用してしきい値規則を作成し、指定したトリガー条件を超過したら診断データの収集を開始することもできます。

症状

存在する特定の問題によっては、データ・サーバーで次のような症状の一部が観察される場合があります。

- データベース接続に関連するパフォーマンス上の問題については、実行状態またはコンパイル状態のアプリケーションの数が急上昇したり、新しいデータベース接続が拒否されている状態が観察されたりする場合があります。
- プロセッサ使用量に関連するパフォーマンス上の問題については、プロセッサの使用率が高い、実行中のプロセス数が多い、またはプロセッサの待ち時間が長いという症状が観察される場合があります。
- メモリー使用量に関連するパフォーマンス上の問題については、使用可能な空きメモリーがない、スワップ・スペースが高い割合で使用されている、過剰なページングが発生している、またはメモリー・リークの疑いがあるなどの症状が観察される場合があります。

問題の診断

データベースへのアプリケーション接続の有無を確認するには、**db2pd -applications** コマンドを発行できます。Status フィールドでは、存在しているデータベースへの接続の数が分かるほか、アプリケーションが実行状態であるのかコンパイル状態であるのか分かります。データベースに接続するアプリケーションの数が時間の経過とともに減少することなく絶えず増加する場合、またはアプリケーション接続の数が急上昇する場合、詳しい診断が必要となる、接続に関連する問題が存在する可能性があります。

UNIX と Linux のオペレーティング・システムでは、**vmstat** コマンドにオペレーティング・システムごとの該当するパラメーターを指定して発行するとユーザーのプロセッサ使用値 (us) が返されるため、プロセッサの使用量に関連する潜在的な問題があるかどうかを確認できます。また、**sy** で返される値を確認して、システム・プロセッサの使用量、およびユーザーとシステムのプロセッサ使用量を組み合わせた情報を取得することもできます。**vmstat** コマンドを複数回発行すると、時間が経過しても問題の症状が持続するかについて、より正確な結論を出すこ

とができます。 `us` の値が 90% を超える頻度が高いことに気付いた場合は、プロセッサ使用量に関連する、より詳細に診断する必要がある問題が存在する可能性があります。 Windows オペレーティング・システムでは、類似する情報を取得するための `db2pd -vmstat` コマンドを使用できます。

メモリー関連の潜在的な問題があるかどうかを確認するには、使用されているスワップ・スペースの値を返す、該当するオペレーティング・システム・コマンドを発行できます (例えば Linux オペレーティング・システムでは、`free` コマンドを使用できます)。使用されているスワップ・スペースの値が常に 90% より大きい場合、メモリー関連のパフォーマンス上の問題が存在する可能性があります。

問題の解決

次の例では、`db2fodc` コマンドに、`-connection`、`-cpu`、`-memory`、および `-detect` をパラメーターとして指定して使用しています。これらのコマンドは実際に実行できますが、潜在的な影響があることに注意する必要があります。つまり、リソースの制約が大きいシステムでは、特定の問題との関係性が非常に高い診断データだけを収集する場合であっても、受け入れられないほどシステムへの要求が増大する可能性があります。これらのコマンドは、IBM サポートの指導の下で実行することをお勧めします。通常、収集されたデータは、分析のために IBM 技術サポートに送信されます。

- データベース接続に関連する問題の診断データを収集するには、次のコマンドを発行します。

```
db2fodc -connections
```

このコマンドは、接続関連の診断データを収集して

`FODC_Connections_timestamp_member` ディレクトリーに置きます。ここで、`timestamp` は `db2fodc -connections` コマンドが実行された時刻、`member` はコレクションが実行されたメンバーです。

- プロセッサ使用量に関連する問題について、既に関連する問題の症状が観察されているときに診断データを収集する場合、次のコマンドを発行できます。

```
db2fodc -cpu
```

`db2fodc -cpu` コマンドは、プロセッサ関連の診断データを収集して、

`FODC_Cpu_timestamp_member` ディレクトリーに置きます。ここで、`timestamp` は `db2fodc -connections` コマンドが実行された時刻、`member` はコレクションが実行されたメンバーです。問題が偶発的である場合、または特定の問題の条件が存在するときに診断データを収集するために事前にシステムを構成する場合には、代替りの方法として、次のコマンドのバリエーションを発行できます。

```
db2fodc -cpu basic -detect us">=90" rqueue"<=1000" condition="and" interval="2"
sleeptime="500" triggercount="3" iteration="4" duration="500" -member all
```

しきい値の規則が指定された `-detect` パラメーターは、しきい値の規則によって指定されたトリガー条件が検出されるまで、プロセッサ関連の情報の収集を遅らせます。 `-detect` パラメーターに独自の規則を指定し、診断データの収集をいつ開始するか決定できます。前の例では、3 回反復する過程の中で 3 回ともユーザーのプロセッサ使用量と実行キューの両方の条件を満たす必要があります。このことは、すべてのメンバーでの診断データの収集をトリガーするには、トリガー条件が合計で 6 秒間存在する必要があることを意味します (トリガー数 3 x

2 秒の間隔 = 存在する必要があるトリガー条件が 6 秒)。 iteration オプションは、トリガー条件の検出とその後の診断データの収集が、各反復の間に 500 秒のスリープ時間を取って 3 回実行されることを指定します。

- メモリー使用量に関連する問題について、既に関連する問題の症状が観察されているときに診断データを収集する場合、次のコマンドを発行できます。

```
db2fodc -memory
```

db2fodc -memory コマンドは、メモリー関連の診断データを収集して、`FODC_Memory_timestamp_member` に置きます。ここで、`timestamp` は **db2fodc -connections** コマンドが実行された時刻、`member` はコレクションが実行されたメンバーです。問題が偶発的である場合、または特定の問題の条件が存在するときに診断データを収集するために事前にシステムを構成する場合には、代替の方法として、次のコマンドのバリエーションを発行できます。

```
db2fodc -memory basic -detect free"<=10" connections">=1000" sleeptime="1" iteration="10" interval="10" triggercount="4" duration="5" -member 3
```

各規則が指定された **-detect** パラメーターは、これらの規則が検出されるまで収集を遅らせます。この例では、メンバー 3 で診断データの収集をトリガーするために、空きメモリーとデータベースへの接続数についてのトリガー条件が 40 秒存在する必要があります (トリガー数 4 x 10 秒の間隔 = 合計 40 秒)。検出と診断データの収集を 10 回反復して実行可能で、有効になる期間は 5 時間です。

- トリガー条件のみを検出し、診断データの収集を行わない場合は、**db2fodc -detect** コマンドに `nocollect` オプションを指定して使用できます。指定された問題の条件がしきい値のヒットとして検出される場合は常に、項目が `db2diag` ログ・ファイルに記録されます。詳細なしきい値のヒットについては診断データを収集しないことを選択した場合、`db2diag` ログの項目を使用して、しきい値規則を作成した問題の条件が検出されたかどうかを判別する必要があります。

```
2011-05-31-15.53.42.639270-240 I2341E790 LEVEL: Event
PID : 13279 TID : 47188095859472PROC : db2fodc
INSTANCE: kunxu NODE : 000
FUNCTION: DB2 UDB, RAS/PD component, pdFodcDetectAndRunCollection, probe:100
CHANGE :
```

```
Hostname: hotel36 Member(s): 0 Iteration: 0
Thresholds hit 0: cs(0)>=0 us(0)<=50 rQueue(0)>=0 free(7980936)>=100
avm(7297432)>=1 swapused(0)>=0 UOW_Executing(4)>=3
Thresholds hit 1: cs(0)>=0 us(0)<=50 rQueue(0)>=0 free(7981000)>=100
avm(7297528)>=1 swapused(0)>=0 UOW_Executing(4)>=3
Thresholds hit 2: cs(0)>=0 us(0)<=50 rQueue(0)>=0 bQueue(1)>=1
free(7981420)>=100 avm(7297596)>=1 swapused(0)>=0 UOW_Executing(4)>=3
Thresholds hit 3: cs(0)>=0 us(0)<=50 rQueue(0)>=0 free(7981436)>=100
avm(7297668)>=1 swapused(0)>=0 UOW_Executing(4)>=3
```

この例では、指定された問題の条件が、メンバー 0 で 4 回しきい値のヒットとして検出されます。各しきい値規則に対して指定した値と、検出された実際の値の両方がログに記録されます。

DB2 についてのデータの分析

データを収集した後、そのデータが現在の特定の問題を解決するためにどのように役立つかを判別する必要があります。分析のタイプは、調査中の問題のタイプと、収集したデータにより異なります。以下の手順は、基本的な DB2 診断データの調査を開始する方法を示しています。

手順

診断データを分析するには、以下のアクションを実行します。

- データのさまざまな断片が互いにどのように関連するかについて明確に理解するようにします。例えば、データが複数のシステムにまたがる場合、データのどの断片がどのソースのものなのかが分かるように、データの編成を工夫します。
- タイム・スタンプを調べて、問題が発生したタイミングと診断データの各断片とが関連していることを確認します。データのソースが異なると、タイム・スタンプの形式が異なる場合がある点に注意してください。さまざまなイベントが発生した時刻が分かるように、それぞれのタイム・スタンプ形式における各要素の表示順序を理解しておきます。
- 問題に関する情報が含まれる可能性が最も高いデータ・ソースを判別し、そこで分析を開始します。例えば、問題がインストールに関連している場合、製品全般またはオペレーティング・システムのログ・ファイルから分析を開始するのではなく、インストール・ログ・ファイルがあるならば、そこから分析を開始します。
- 分析の具体的な方法はデータ・ソースごとに固有のものですが、ほとんどのトレースおよびログ・ファイルに当てはまる 1 つのヒントは、問題が発生したデータ内のポイントをまず最初に突き止めることです。そのポイントを突き止めたら、問題の根本原因を解明するために、そのデータを時間をさかのぼって調べることができます。
- ある問題を調査していて、作動中の環境と作動していない環境におけるその問題の比較データがある場合には、各環境のオペレーティング・システムと製品構成の詳細を比較することから始めます。

持続されたトラップのリカバリー

DB2 インスタンスは、発生したトラップに対する First Occurrence Data Capture (FODC) パッケージを準備します。デフォルトでは、DB2 インスタンスには、トラップの回復力が構成されています。DB2 インスタンスは、トラップが持続可能なものかどうかを判別しています。「持続可能な」という用語は、トラップされた DB2 エンジン・スレッドが中断され、DB2 インスタンスは実行を続けることを意味します。

このタスクについて

デフォルトでは、DB2 インスタンスには、**DB2RESILIENCE** レジストリー変数のデフォルト設定に基づくトラップの回復力が構成されています。

持続されたトラップの認識

トラップは持続され、トラップ (DB2 のプログラミング・エラー) が発生し

た場合のデータベース・システムでの影響を最小限に抑えます。トラップが持続されると、次の診断が行われます。

1. **diagpath** データベース・マネージャー構成パラメーターで指定された完全修飾パスに基づき、FODC ディレクトリーが作成されます。
2. エラー・メッセージ ADM14013C が、管理通知ログ・ファイルおよび db2diag ログ・ファイルに記録されます。

注: トラップが持続できなかった場合は ADM14011C が記録され、インスタンスがシャットダウンされます。

3. エラー sqlcode -1224 がアプリケーションに戻されます。
4. EDU スレッドは中断され、**db2pd -edus** の出力で監視できます。

リカバリー

持続されたトラップが通常のインスタンス操作を妨げることはないと思われていますが、持続された EDU スレッドはいくつかのリソースを保持するので、次のステップに従い、インスタンスをできるだけ早く停止し再始動することをお勧めします。

1. タイムアウト期間内に COMMIT または ROLLBACK を発行するアクティブなアプリケーションをすべて終了すると、**db2start** コマンドの実行時にクラッシュ・リカバリーのリカバリー・ウィンドウが最小化されますが、これを行うには、次のコマンドを発行します。

```
db2 quiesce instance instance_name user user_name
      defer with timeout minutes
```

2. [オプション] ステップ 1 のタイムアウト期間中に COMMIT または ROLLBACK を実行しなかった任意のアプリケーション、およびタイムアウト期間の完了後にデータベースにアクセスした任意の新規アプリケーションを終了するには、次のコマンドを発行します。

```
db2 quiesce instance instance_name user user_name immediate
```

3. インスタンスをシャットダウンするために、次のコマンドを実行します。

```
db2stop force
```

4. 以下のいずれかのコマンドを使用して DB2 インスタンスを再始動します。

```
db2start
```

または

```
START DATABASE MANAGER
```

診断

diagpath データベース・マネージャー構成パラメーターで指定された FODC ディレクトリーを見つけます。FODC ディレクトリーの場所は、管理通知ログ・ファイルまたは db2diag ログ・ファイルのビューでも確認できます。FODC 情報を IBM ソフトウェア・サポートに送信します。

ロード操作に関する db2diag ログ項目の識別

ロード操作に関する診断情報を識別し、そのロード操作に関連した **db2diag** コマンド・ログ・メッセージを見つけることは、問題があるロード操作をトラブルシューティングする際の重要なステップです。

症状

複数のロード操作を実行すると、ロード操作が完了するのに要する時間が通常より長かったり、ハングしているように見えたりすることに気付く場合があります。

問題の解決

どのロード操作に問題があるかを判別するには、次のようにします。

1. **db2pd -utilities** コマンドを発行して、すべてのロード操作の ID を表示します。以下の出力例では、ID が **LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)** の 1 つのロード操作のみ行われています。

```
Database Partition 0 -- Active -- Up 0 days 00:16:15 -- Date 05/13/2011 13:00:33

Utilities:
Address      ID  Type  State  Invoker  Priority  StartTime          DBName  NumPhases  CurPhase  Description
0x000000020120E2A0 1  LOAD  0      0        0        Fri May 13 12:44:34  SAMPLE  2          2          [LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)]...

Progress:
Address      ID  PhaseNum  CompletedWork  TotalWork  StartTime          Description
0x000000020120E600 1  1         0 bytes       0 bytes     Fri May 13 12:44:34  SETUP
0x000000020120E7E0 1  2         0 rows        0 rows     Fri May 13 12:44:34  LOAD
```

2. **db2diag -g 'dataobj:=loadID'** コマンドを発行します。*loadID* は前のステップで検出されたロード操作の ID です。このコマンドは、指定されたロード操作に関連する、**db2diag** コマンド・ログからのすべての診断ログ・メッセージを表示します。以下の例は、前の例で識別されたロード操作 ID を指定してこのコマンドを発行する場合の表示内容を示しています。

```
$ db2diag -g 'dataobj:= LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)'
```

```
2011-05-13-14.27.03.134598-240 I1700E525          LEVEL: Warning

PID      : 29615          TID      : 47039995963712PROC : db2sysc

INSTANCE: vivmak          NODE     : 000          DB       : SAMPLE

APPHDL   : 0-7          APPID    : *LOCAL.vivmak.110513164421

AUTHID   : VIVMAK

EDUID    : 44          EDUNAME  : db2lrid

FUNCTION: DB2 UDB, database utilities, sqlulPrintPhaseMsg, probe:314

DATA #1 : String, 99 bytes

LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)

Completed LOAD phase at 05/13/2011 14:27:03.134463.
```

上記のステップの完了後には、問題のあるロード操作を十分識別できるだけの情報が得られます。しかし、ロード操作に関する情報がさらに必要な場合は、**db2pd -db <dbname> -load loadID="LOADID" stacks** コマンドを発行して、スタック・トレースを入手してください。**stacks** オプションは、UNIX および Linux オペレーテ

ィング・システムでのみ使用できます。以下の例は、前の例で識別されたロード操作 ID を指定して、サンプル・データベースに対してこのコマンドを発行する場合の表示内容を示しています。

```
$ db2pd -db sample -load loadID="LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)" stacks

Attempting to produce stack traces for LOAD edu 40
Attempting to produce stack traces for LOAD edu 41
Attempting to produce stack traces for LOAD edu 42
Attempting to produce stack traces for LOAD edu 44
Attempting to produce stack traces for LOAD edu 45

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:27 -- Date 05/13/2011 14:28:32

Node Number : 0
Database Name : SAMPLE

LoadID                EDUID  EDUNAME  TableName  SchemaName  AppHandl  [nod-index]  Application ID          StartTime              LoadPhase
LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  40      db21frm0  T1         VIVMAK      7          [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD
LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  41      db21frm1  T1         VIVMAK      7          [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD
LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  42      db21frm2  T1         VIVMAK      7          [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD
LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  44      db21rid   T1         VIVMAK      7          [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD
LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  45      db21bm0   T1         VIVMAK      7          [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD
```

db2pd -db <dbname> -load loadID="LOADID" stacks コマンドは、指定されたロード操作に関連した EDU 情報をすべて表示し、diagpath ディレクトリー内にスタック・トレース・ファイルを作成します。

取得したすべての情報を使用して、さらにロード操作のモニターや強制終了などのトラブルシューティング手法を実行できます。また、IBM 技術サポートが問題のあるロード操作をトラブルシューティングするために、この集められた情報を求めることもあります。

収集された情報を使用して **db2trc** コマンドを実行し、さらにトラブルシューティングを行うこともできます。取得した情報を使用し、特定のロード操作に対して **db2trc** コマンドを実行するには、次のようにします。

1. **db2pd -load** コマンドを実行して、対象にする特定のロード操作のアプリケーション ID を取得します。
2. **db2trc -appid** コマンドまたは **db2trc -apphdl** コマンドを実行して、ロード操作に関する情報をさらに記録します。

以下の例では、このトピックの前述の例から、ロード ID **LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)** のアプリケーション ID ***LOCAL.vivmak.110513164421** を使用して、**db2trc** コマンドを実行します。

```
$ db2trc on -appid *LOCAL.vivmak.110513164421
```

トレースはオンになります。

```
$ db2trc info
```

```
Marker                : @TRACE@
Trace version          :      7.0
Platform               : Linux/X
Build level            : n110612
maxBufferSize          : 33554432 bytes (32 MB)
```

```

auxBufferSize      : 524288 bytes (0 MB)
allocationCount    : 2
DB2TRCD pid       : 0
Trace destination  : <shared memory buffer>
numSuspended      : 0
Trace starting time : 2011-06-16-10.07.52.209999-240

Buffer size        : 33554432 bytes (32 MB)
Allow buffer to wrap : yes
Mask               : *.*.*.*.*
Timestamps         : disabled
PID.TID mask       : all
Fixed data mask #1 : all
Fixed data mask #2 : all
Max system errors  : infinite
Treat this rc as sys err: none
Member mask        : none
Application handle mask : none
Application ID mask : *LOCAL.vivmak.110513164421

```

次の例では、**db2pd -load** コマンドの出力から入手したアプリケーション・ハンドルを使用して、**db2trc** コマンドで有効になるトレース・オプションを変更します。

```
$ db2trc chg -apphd1 7
```

```
Trace has been changed
```

```
$ db2trc info
```

```

Marker            : @TRACE@
Trace version      : 7.0
Platform          : Linux/X
Build level       : n110612
maxBufferSize     : 33554432 bytes (32 MB)
auxBufferSize     : 524288 bytes (0 MB)
allocationCount   : 2
DB2TRCD pid       : 0
Trace destination  : <shared memory buffer>
numSuspended      : 0

```

```
Trace starting time      : 2011-06-16-10.10.11.816264-240

Buffer size             : 33554432 bytes (32 MB)
Allow buffer to wrap    : yes
Mask                   : *.*.*.*.*
Timestamps             : disabled
PID.TID mask           : all
Fixed data mask #1     : all
Fixed data mask #2     : all
Max system errors      : infinite
Treat this rc as sys err: none
Member mask            : none
Application handle mask : 7
Application ID mask    : none
```

管理用タスク・スケジューラーのトラブルシューティング

このチェックリストは、管理用タスク・スケジューラー内のタスクの実行時に発生する問題をトラブルシューティングするときに役立ちます。

手順

1. タスクが予期したとおりに実行されなかった場合、最初に行う必要があるのは、`ADMIN_TASK_STATUS` 管理ビューにある実行状況レコードを調べることです。
 - レコードがある場合、さまざまな値を調べます。特に、`STATUS`、`INVOCATION`、`SQLCODE`、`SQLSTATE`、`SQLERRMC`、および `RC` 列に注目します。その値が問題の根本原因を示していることがよくあります。
 - ビューに実行状況レコードがない場合は、タスクが実行されなかったということです。考えられる原因としては、以下のいくつかのものが 있습니다。
 - 管理用タスク・スケジューラーが使用不可になっている。管理用タスク・スケジューラーが使用不可になっていると、タスクは実行されません。スケジューラーを使用可能にするには、`DB2_ATS_ENABLE` レジストリー変数を設定します。
 - タスクが除去された。いずれかのユーザーによってタスクが除去された可能性があります。`ADMIN_TASK_LIST` 管理ビューを照会して、タスクが存在するかを確認してください。
 - スケジューラーがタスクを認識していない。管理用タスク・スケジューラーは、アクティブになっている各データベースに 5 分ごとに接続して、新しいタスクおよび更新されたタスクを検索します。この期間が経過するまで、スケジューラーはタスクを認識しません。少なくとも 5 分待機してください。

- データベースが非アクティブになっている。データベースがアクティブでないと、管理用タスク・スケジューラーは、タスクの検索や実行ができません。データベースをアクティブ化してください。
- トランザクションがコミットされていない。管理用タスク・スケジューラーは、コミットされていないタスクを無視します。タスクの追加、更新、または除去後は、必ずコミットするようにしてください。
- スケジュールが無効になっている。タスクのスケジュールによって、タスクの実行ができなくなっている可能性があります。例えば、タスクの呼び出し回数がすでに最大回数に達していることがあります。
ADMIN_TASK_LIST ビューでタスクのスケジュールを確認して、必要に応じてスケジュールを更新してください。

2. ADMIN_TASK_STATUS 管理ビューを調べても問題の原因が判別できない場合は、DB2 診断ログを参照してください。すべてのクリティカル・エラーは **db2diag** ログ・ファイルに記録されます。通知イベント・メッセージも、タスク実行時に管理用タスク・スケジューラー・デーモンによってログに記録されます。それらのエラーおよびメッセージは、「管理用タスク・スケジューラー」コンポーネントによって識別されます。

次のタスク

前述の手順を実行しても問題の原因を判別できない場合は、IBM ソフトウェア・サポートと共に問題管理レコード (PMR) を開くことを検討してください。これらの手順を実行したことを伝え、収集した診断データを送信してください。

データベースが現在使用中であるために操作が失敗する

データベースのドロップ、データベース接続の終了、およびデータベースのバックアップ・コピーの作成など、一部のデータベースの操作では、関連するデータベースを非アクティブにして、そのデータベースにアプリケーションが接続できないようにする必要があります。アプリケーションが接続されているアクティブなデータベースに対してそのような操作を実行しようとする、操作は失敗し、SQL コード -1035N でエラーが返されることがあります。

症状

データベースのドロップ、データベース接続の終了、およびデータベースのバックアップ・コピーの作成など、特定のデータベース操作を実行しようとしたとき、操作が失敗して、データベースが現在使用中であることを示すエラーが返されることがあります。

原因

この SQL1035N エラー・メッセージは、次のいずれかのシナリオで返される可能性があります。

1. データベースへの開いた接続が存在しているため、試行した操作が成功しません。これは、以下のような状況において行われます。
 - 排他使用が要求されましたが、そのデータベースは、(同じプロセスを使用する) 別のユーザーによって共有データベースとして既に使用されています。

- ・ 排他使用が要求されましたが、そのデータベースは、排他データベースとして既に使用されています。これは、2 つの異なるプロセスが同じデータベースにアクセスしようとしていることを意味します。
 - ・ データベースへの接続の最大数に達しました。
 - ・ データベースが、別のシステムの別のユーザーによって使用されています。
2. データベースが明示的にアクティブ化されたため、操作が成功しません。
 3. データベースは、WRITE SUSPEND 状態であるため、アクティブです。

問題の解決

1. 以下のアクションのいずれかを行います。
 - ・ データベースが使用中でなくなったときに、コマンドを再サブミットします。
 - ・ 現行ユーザーに一致するように許可を変更するか、またはデータベースが使用中でなくなるまで待機します。
 - ・ データベースが排他使用でなくなるまで待機します。
 - ・ 別のシステムの他のユーザーがデータベースから切断するまで待機します。
 - ・ **QUIESCE DATABASE DEFER WITH TIMEOUT** *<minutes>* コマンドと **CONNECT RESET** コマンドを発行して、データベースへの接続を解放し、アプリケーション・ユーザーが再接続できないようにします。DEFER オプションを使用すると、実行中のすべてのトランザクションをロールバックする代わりに、アプリケーションが現在の作業単位をコミットするまで待機します。
 - ・ **LIST APPLICATIONS** コマンドを発行して、データベースへの接続をリストします。**FORCE APPLICATION ALL** コマンドを発行して、データベースへの接続を解放します。

注: **FORCE APPLICATION ALL** コマンドは、非同期コマンドであるため、接続のクリーンアップが継続している場合でも正常終了として戻る可能性があります。コマンドが完了するまでには不確定の時間間隔が必要です。

2. **DEACTIVATE DATABASE** *<database-alias>* コマンドを使用して、データベースを非アクティブにします。
3. **SET WRITE RESUME FOR DATABASE** コマンドを発行して、データベースで書き込み操作を再開します。
4. 次のコマンドを発行して、オフライン操作が実行されていたメンバーを再開します。


```
db2start member <member-number>
```
5. 最初に失敗した操作を再発行します。

圧縮のトラブルシューティング

データ・コンプレッション・ディクショナリーが自動的に作成されない

大規模な表、または表用の大きな XML ストレージ・オブジェクトがありますが、データ・コンプレッション・ディクショナリーが作成されませんでした。データ・コンプレッション・ディクショナリーが期待どおりに作成されなかった理由を調べ

ます。この情報は、表オブジェクト用のコンプレッション・ディクショナリーと XML ストレージ・オブジェクト用のコンプレッション・ディクショナリーの両方に適用されます。

以下のような状況が起こり得ます。

- COMPRESS 属性を YES に設定した表があります。
- その表は一定の期間存在しており、データが追加されたり除去されたりしました。
- その表のサイズは、しきい値のサイズに近くなっています。データ・コンプレッション・ディクショナリーが自動的に作成されるようにしたいと考えます。
- 表のサイズがしきい値サイズよりも大きくなるように、表データ取り込み操作 (INSERT、LOAD INSERT、REDISTRIBUTE など) を実行します。
- データ・コンプレッション・ディクショナリーの自動作成が実行されません。データ・コンプレッション・ディクショナリーは作成されず、表に配置されませんでした。この時点で表に追加されたデータが圧縮されると期待していましたが、データは依然として圧縮解除されたままです。
- XML データの場合に、データが DB2 バージョン 9.7 ストレージ形式になっています。

表の XML ストレージ・オブジェクト内のデータの圧縮は、その表に DB2 バージョン 9.5 以前を使用して作成された XML 列が含まれている場合はサポートされません。そのような表でデータ行の圧縮を有効にした場合は、表オブジェクト内の表の行データだけが圧縮されます。挿入、ロード、または REORG 操作の際に XML ストレージ・オブジェクトを圧縮できない場合は、XML 列が DB2 V9 または DB2 V9.5 を使用して作成された場合に限り、メッセージが db2diag ログ・ファイルに書き込まれます。

なぜデータは圧縮されなかったのでしょうか。

データは、コンプレッション・ディクショナリーが自動的に作成されるしきい値サイズよりも大きかったものの、チェックされた別の条件があります。つまり、ディクショナリーが作成されるためには、オブジェクトの中に十分なデータが存在していなければならない、という条件です。この要件はメッセージ ADM5591W によって通知されます。データに対する過去のアクティビティーには、データの削除や除去が含まれていた可能性があります。つまり、オブジェクトの中にデータのない大きなセクションが存在していた可能性がある、ということです。このような場合に、オブジェクト・サイズのしきい値に達した (または、しきい値を超えた) 大きなオブジェクトがあっても、ディクショナリーが作成されるほどの十分なデータがオブジェクトの中に存在しないということがあります。

オブジェクトに対して多数のアクティビティーを実行している場合は、定期的にアクティビティーを再編成する必要があります。XML データの場合は、longlobdata オプションを指定して表を再編成する必要があります。そうしなければ、オブジェクト・サイズは大きくても、データがそれほど多く取り込まれていない、という状況が生じる可能性があります。オブジェクトを再編成すれば、フラグメント化したデータを除去して、オブジェクトのデータをコンパクトにすることができます。再編成後は、オブジェクトが小さくなり、データの取り込み具合が高密度になります。再編成したオブジェクトは、オブジェクトのデータ量をより正確に反映するこ

とになり、データ・コンプレッション・ディクショナリーが自動的に作成されるしきい値サイズよりも小さくなる可能性があります。

オブジェクトにデータがそれほど多く取り込まれていない場合は、**REORG TABLE** コマンド (XDA の場合は **LONGLOBDATA** オプションを使用) を使用して表の再編成を実行し、ディクショナリーを作成できます。デフォルトでは、**KEEPDICTIONARY** が指定されています。ディクショナリー作成を強制するには、**RESETDICTIONARY** を指定します。

表の再編成が必要かどうかを確認するには、**REORGCHK** コマンドを使用します。

表でデータ行の圧縮が有効になっていない場合、その表についてディクショナリーの自動作成 (Automatic Dictionary Creation: ADC) は発生しません。データベースについて ADC 処理が無効になっている場合は、メッセージ **ADM5594I** が返されてその理由が示されます。

表に DB2 バージョン 9.5 以前を使用して作成された XML 列が含まれている場合は、**ADMIN_MOVE_TABLE** ストアード・プロシージャを使用して表をアップグレードした後でデータ行の圧縮を有効にします。

行圧縮によって一時表用のディスク・ストレージ・スペースが削減されない

Storage Optimization フィーチャーのライセンス交付を受けている場合でも、一時表用のディスク・ストレージ・スペースが期待したほど節約されない状況が発生することが知られています。

症状

一時表に対して行圧縮を有効にすることによって実現可能なディスク・スペースの節約が、期待したとおりに実現していません。

原因

- ほとんどの場合、この状況は、それぞれがデータベース・マネージャー・メモリーの一部を消費する多数のアプリケーションが同時に実行されて一時表を作成することの結果として発生します。これは、コンプレッション・ディクショナリーの作成に使用可能なメモリーの不足につながります。この状況が発生した場合、通知は行われません。
- 行圧縮は、アルゴリズムに応じたディクショナリー・ベースのアプローチを使用して行われます。一時表の行がディスク・スペースの大幅な節約を実現できるほど大きい場合、その行は圧縮されます。一時表内の小さい行は圧縮されず、ディスク・ストレージ・スペースが期待したほど節約されない原因になります。この状況が発生した場合、通知は行われません。

リスク

行サイズがしきい値を満たしていない一時表で行圧縮が使用されていない場合を除いては、システムに対するリスクはありません。使用可能メモリーが大幅に制限された状態が続く場合、データベース・マネージャーに対して他の悪影響を与える可能性があります。

データ・レプリケーション・プロセスが、圧縮された行イメージを圧縮解除できない

圧縮された行イメージを含むログ・レコードをデータ・レプリケーション・ソリューションが、正常に圧縮解除できない状況が発生する場合があります。一時的な(一時)エラーの場合は返される SQL コードがエラーの原因に対応しますが、永続エラーは通常、SQL0204N 通知によって示されます。一時的なエラー状況の場合にのみ、その後でログ・レコード内の行イメージが正常に圧縮解除される可能性があります。db2ReadLog API は、ログ・レコードを圧縮解除できなくても、他のログ・レコードの処理を継続します。

症状

圧縮されたユーザー・データを含むログ・レコードをログ・リーダーで読み取っている際に、一時的なエラーや永続エラーが発生する可能性があります。以下に示すのは、圧縮されたデータ(行イメージ)を含むログ・レコードを読み取る際に発生することのある 2 種類のエラー・クラスの例のリストです。ただし、可能性のあるエラーすべてをリストしているわけではありません。

一時的なエラー:

- 表スペース・アクセスが許可されない
- 表にアクセスできない(ロック・タイムアウト)
- 必要なディクショナリーをロードして保管するためのメモリーが不足している

永続エラー:

- 表が存在する表スペースが存在しない
- ログ・レコードが属する表または表パーティションが存在しない
- 表または表パーティション用のディクショナリーが存在しない
- ログ・レコードに、表内のディクショナリーよりも古いディクショナリーを使用して圧縮された行イメージが含まれている

原因

レプリケーション・ソリューションや他のログ・リーダーが、データベース・アクティビティーに付いて行くことができなくなり、圧縮されたユーザー・データを含むログ・レコードを読み取る際にエラーを受け取ることがあります(シナリオ 1 を参照)。このようなケースは、読み取り中のレコードに含まれる圧縮済みユーザー・データが(ログの読み取り時に)表で使用可能なものより古いコンプレッション・ディクショナリーによって圧縮されている場合に、発生する可能性があります。

同様に、表がドロップされると、その表に関連付けられているディクショナリーもドロップされます。この場合、表の圧縮された行イメージは圧縮解除できません(シナリオ 2 を参照)。圧縮された状態にない行イメージは、表がドロップされても引き続き読み取りや複製が可能であるため、この制限は、圧縮された状態にない行イメージには適用されません。

いずれの 1 つの表についても、1 つのアクティブ・データ・コンプレッション・ディクショナリーと 1 つの履歴ディクショナリーだけが存在できます。

シナリオ 1:

表 t6 では圧縮が有効になっています。レプリケーションの目的で、この表について DATA CAPTURE CHANGES 属性が有効になっています。この表はデータ・レプリケーション・アプリケーションによって複製されており、ログ・リーダーは、圧縮されたデータ (行イメージ) を含むログ・レコードを読み取っています。

db2ReadLog API を使用するクライアント・ログ・リーダーが、REORG TABLE コマンドが発行された (表のディクショナリーが再構築された) 後、LOAD 操作が表 t6 に対して実行される際に最初の INSERT ステートメントの最初のログ・レコードを読み取ります。

既にコンプレッション・ディクショナリーが含まれていて DATA CAPTURE CHANGES 属性が有効になっている表 t6 に対して、以下のステートメントが実行されます。

```
-> db2 alter table t6 data capture changes
-> db2 insert into t6 values (...)
-> db2 insert into t6 values (...)
```

表 t6 用のデータ・コンプレッション・ディクショナリーが既に存在するため、ALTER の後の 2 つの INSERT は (表 t6 のコンプレッション・ディクショナリーを使用して) 圧縮されます。この時点で、ログ・リーダーはまだ最初の INSERT ステートメントに達していません。

以下の REORG TABLE コマンドを実行すると、表 t6 用の新規のコンプレッション・ディクショナリーが構築され、現在のコンプレッション・ディクショナリーが履歴ディクショナリーとして保持されることにより、ログ・リーダーは現在のコンプレッション・ディクショナリーの後ろにあるディクショナリーとなります (ただし、REORG の後で履歴ディクショナリーはメモリーにロードされません)。

```
-> db2 reorg table t6 resetdictionary
```

ログ・リーダーが INSERT ステートメント用の INSERT ログを読み取るにつれて、履歴ディクショナリーがメモリーに読み取られることが必要になり、表 t6 で LOAD 操作が発生します。

```
-> db2 load from data.del of del insert into table t6 allow no access
```

LOAD がソース表に対して実行されると、表 t6 は、指定された ALLOW NO ACCESS オプションのために Z ロックされます。INSERT ログ・レコードで見つかった行イメージを圧縮解除するため、ログ・リーダーは履歴ディクショナリーをメモリーにロードする必要がありますが、ディクショナリーをフェッチするには IN 表ロックが必要です。このケースでは、ログ・リーダーはこのロックの取得に失敗します。その結果、db2ReadLogFilterData 構造の sqlcode メンバーから SQL コード SQL2048N が返されます。これは一時的なエラーに対応します (つまり、API が再び呼び出されると、ログ・レコードは圧縮解除される可能性があります)。ログ・リーダーは圧縮された行イメージをログ・レコード内に返し、引き続き次のログ・レコードを読み取ります。

シナリオ 2:

表 t7 では DATA CAPTURE CHANGES 属性が有効になっています。ストレージ・コストを削減するため、この表では圧縮が有効になっています。この表はデー

タ・レプリケーション・アプリケーションによって複製されていますが、ログ・リーダーがソース表アクティビティに付いて行くことができなくなったため、ログ・リーダーがログ・レコードから再び読み取る前にデータ・コンプレッション・ディクショナリーが既に 2 回構築されました。

DATA CAPTURE CHANGES 属性が既に有効になっている表 t7 に対して以下のステートメントが実行されます。ここで、表圧縮は有効になっており、新規ディクショナリーが構築されます。

```
-> db2 alter table t7 compress yes
-> db2 reorg table t7 resetdictionary
-> db2 insert into t7 values (...)
```

クライアント・ログ・リーダーが、db2ReadLog API を使用して、以下のステートメント内の最初の INSERT ステートメントに対応する次のログを読み取ろうとしています。

```
-> db2 insert into t7 values (...)
...
-> db2 reorg table t7 resetdictionary
-> db2 insert into t7 values (...)
...
-> db2 reorg table t7 resetdictionary
```

このケースでは、ログ・リーダーが複数の **REORG RESETDICTIONARY** 操作に付いて行くことができなくなったため、db2ReadLog API はログ・レコードの内容を圧縮解除できません。ログ・レコード内の行イメージを圧縮解除するために必要なディクショナリーは表内で見つからず、2 番目の REORG のコンプレッション・ディクショナリーと最後の REORG のコンプレッション・ディクショナリーだけが表とともに保管されます。ただし、db2ReadLog API がエラーで失敗することはありません。代わりに、圧縮解除された行イメージがユーザー・バッファーに返され、ログ・レコードの前にある db2ReadLogFilterData 構造で、sqlcode メンバーから SQL コード SQL0204N が返されます。このコードは永続エラーに対応します (つまり、ログ・レコードは永久に圧縮解除できません)。

環境

このように、古いコンプレッション・ディクショナリーが欠落しているために、圧縮されたログ・レコードを正常に圧縮解除することに失敗する状況は、データ・レプリケーション・ソリューションが db2ReadLog API を使用し、DATA CAPTURE CHANGES 属性が表について設定されている任意のプラットフォームで発生する可能性があります。

問題の解決

一時的なエラーの場合は、読み取り要求を再発行してログを正常に読み取れる可能性があります。例えば、ログ・レコードが表スペース内の表に属しており、その表へのアクセスが許可されていない場合、そのログ・レコードを圧縮解除するためのディクショナリーにアクセスできない可能性があります (シナリオ 1 を参照)。表スペースが後で使用可能になった場合、その時点でログ読み取り要求を再発行すると、ログ・レコードを正常に圧縮解除できる可能性があります。

- 一時的なエラーが返された場合 (シナリオ 1 を参照)、エラー情報を確認して適切なアクションを実行します。これには、表操作の完了を待機することも含まれます。これにより、ログ・レコードを再読み取りして圧縮解除を正常に行える場合もあります。
- 永続エラーが発生した場合 (シナリオ 2 を参照)、ログ・レコード内の行イメージは、その行イメージを圧縮するために使用したコンプレッション・ディクショナリーが使用可能でなくなったため、圧縮解除できません。この場合、レプリケーション・ソリューションでは、影響を受ける (ターゲット) 表を再初期化しなければならぬことがあります。

グローバル変数問題のトラブルシューティング

一般に、グローバル変数に関連したアプリケーションのトラブルシューティングは、その問題を検出したユーザーがグローバル変数の READ 権限を持っている限り、問題にはなりません。READ 権限さえあれば、VALUES(Global Variable Name) ステートメントを実行して、グローバル変数の値を確認できるからです。場合によっては、アプリケーションを実行しているユーザーがグローバル変数の READ アクセス権を持っていないこともあります。

シナリオ 1 では、グローバル変数を参照するとき起こり得る問題と、その簡単な解決策を取り上げます。シナリオ 2 では、よくある状況として、グローバル変数の READ 権限を適切なユーザーに与えなければならない場合を取り上げます。

シナリオ 1

グローバル変数の参照は、正しく修飾する必要があります。名前が同じでも、スキーマの異なる変数が存在する場合に、PATH レジスター値の前のほうで正しくないスキーマが検出される可能性があります。1 つの解決策は、グローバル変数の参照を完全に修飾することです。

シナリオ 2

アプリケーション開発者 (developerUser) が、(データベース・グローバル変数ではなく) いくつかのセッション・グローバル変数の値に基づいて、プロシージャー、ビュー、トリガーなどの複雑なセットを作成します。それらのグローバル変数は、その開発者だけが読み取りアクセス権を持っています。その状況で、アプリケーションのエンド・ユーザー (finalUser) がログインして、developerUser によって作成された環境を使用して SQL を実行し始めます。finalUser は、developerUser に対して、参照できるはずのデータを参照できないと苦情を言います。この問題のトラブルシューティングとして、developerUser は、自分の許可 ID を finalUser の許可 ID に変更してから、finalUser としてログインし、finalUser と同じ SQL を実行してみます。developerUser は、finalUser の言い分が正しいことを確認できました。確かに問題があります。

developerUser は、自分が参照できるセッション・グローバル変数値と同じセッション・グローバル変数を finalUser が参照できるかどうかを確認しなければなりません。developerUser は、SET SESSION USER を実行して、finalUser が参照できるセッション・グローバル変数値を確認します。この問題を確認して解決するための推奨方法は、以下のとおりです。

developerUser は、finalUser として SET SESSION USER を使用する権限をセキュリティ管理者 (secadmUser) に申請します。developerUser は、本人としてログインしてから、SET SESSION AUTHORIZATION ステートメントを使用して、SESSION_USER 特殊レジスターに finalUser の許可を設定します。問題の SQL を実行した後に、別の SET SESSION AUTHORIZATION ステートメントを使用して、developerUser にスイッチバックします。developerUser は、VALUES ステートメントを実行して、セッション・グローバル変数の実際の値を確認します。

以下は、developerUser がデータベースで実行したアクションを示すサンプル SQL です。

```
#####
# developerUser connects to database and creates needed objects
#####

db2 "connect to sample user developerUser using xxxxxxxx"

db2 "create table security.users ¥
(userid varchar(10) not null primary key, ¥
firstname varchar(10), ¥
lastname varchar(10), ¥
authlevel int)"

db2 "insert into security.users values ('ZUBIRI', 'Adriana', 'Zubiri', 1)"
db2 "insert into security.users values ('SMITH', 'Mary', 'Smith', 2)"
db2 "insert into security.users values ('NEWTON', 'John', 'Newton', 3)"

db2 "create variable security.gv_user varchar(10) default (SESSION_USER)"
db2 "create variable security.authorization int default 0"

# Create a procedure that depends on a global variable
db2 "CREATE PROCEDURE SECURITY.GET_AUTHORIZATION() ¥
SPECIFIC GET_AUTHORIZATION ¥
RESULT SETS 1 ¥
LANGUAGE SQL ¥
SELECT authlevel INTO security.authorization ¥
FROM security.users ¥
WHERE userid = security.gv_user"

db2 "grant all on variable security.authorization to public"
db2 "grant execute on procedure security.get_authorization to public"
db2 "terminate"

#####
# secadmUser grants setsessionuser
#####
db2 "connect to sample user secadmUser using xxxxxxxx"
db2 "grant setsessionuser on user finalUser to user developerUser"
db2 "terminate"

#####
# developerUser will debug the problem now
#####

echo "-----"
echo " Connect as developerUser "
echo "-----"
db2 "connect to sample user developerUser using xxxxxxxx"

echo "-----"
echo " SET SESSION AUTHORIZATION = finalUser "
echo "-----"
db2 "set session authorization = finalUser"
```

```

echo "--- TRY to get the value of gv_user as finalUser (we must not be able to)"
db2 "values(security.gv_user)"

echo "--- Now call the procedure---"
db2 "call security.get_authorization()"

echo "--- if it works it must return 3 ---"
db2 "values(security.authorization)"

echo "-----"
echo " SET SESSION AUTHORIZATION = developerUser "
echo "-----"

db2 "set session authorization = developerUser"

echo "--- See what the variable looks like ----"
db2 "values(security.gv_user)"

db2 "terminate"

```

不整合のトラブルシューティング

データ不整合のトラブルシューティング

データベース内のどこにデータ不整合があるかを診断することは非常に重要です。データ不整合を特定する 1 つの方法は、**INSPECT** コマンドの出力を使用して、どこに問題があるかを確認することです。不整合が見つければ、その問題をどのように処理するかを決めます。

データ整合性の問題が確認された場合は、2 つのオプションがあります。

- IBM ソフトウェア・サポートに連絡して、データ不整合のリカバリーの支援を要請できます。
- データ整合性の問題があるデータベース・オブジェクトをいったんドロップしてから再作成できます。

データ不整合の証拠があるデータベース、表スペース、表をチェックするために、**INSPECT** コマンドの **INSPECT CHECK** バリエーションを使用します。**INSPECT CHECK** コマンドの結果が生成されたら、**db2inspf** コマンドを使用して、検査結果をフォーマットする必要があります。

INSPECT コマンドが完了しない場合は、IBM ソフトウェア・サポートに連絡してください。

索引とデータの不整合のトラブルシューティング

表の正しいデータに素早くアクセスするには、索引が正確である必要があります。そうでない場合は、データベースが破損しています。

索引とデータの不整合に関するオンライン・チェックを実行するには、オブジェクト相互チェック節の **INDEXDATA** オプションを使用して **INSPECT** コマンドを使用します。索引データ・チェックは、**INSPECT** コマンドの使用時にデフォルトでは実行されないため、明示的に要求する必要があります。

INSPECT が **INDEXDATA** 検査を実行しているときに、索引データ不整合のためにエラーが検出されると、エラー・メッセージ **SQL1141N** が返されます。このエラ

ー・メッセージが返されると、同時にデータ診断情報が収集され、**db2diag** ログ・ファイルにダンプされます。緊急メッセージも管理通知ログに記録されます。**db2diag** ログ・ファイル分析ツール (**db2diag**) は、**db2diag** ログ・ファイルの内容のフィルター処理とフォーマットに使用します。

ロックに関する影響

INSPECT コマンドの **INDEXDATA** オプションを使用して索引とデータの不整合をチェックしている間、検査対象の表は **IS** モードでのみロックされます。

INDEXDATA オプションを指定した場合のデフォルトでは、明示的に指定されたレベル節オプションの値だけが使用されます。明示的に指定されていないレベル節オプションについては、デフォルト・レベル (**INDEX NORMAL** と **DATA NORMAL**) が **NORMAL** から **NONE** に上書きされます。

不整合状態にあるデータベースのトラブルシューティング

症状

データベース操作を実行しようとしたときに、データベースが不整合状態にあることを示すエラー・メッセージを受け取る場合があります。

原因

この問題の考えられる原因は以下のとおりです。

- 前のセッションが異常終了した (電源障害など) ために、データベースがオフラインになっている。
- **db2ckupgrade** コマンドの実行時にエラーが検出された場合:
 - データベースがオンラインになっていて、データベース内のデータを変更した SQL が発行された。
 - データベースがオンラインになっていて、**HADR** が有効になっている。
- **DB2 pureScale**環境の場合のみ、以下の原因も考えられます。
 - 前のセッションが異常終了した結果、この **DB2** メンバーのデータベースがオフラインになっている。
 - 前のセッションが異常終了したために、**DB2 pureScale** 環境インスタンス全体でデータベースがオフラインになっている。
 - インスタンスで追加/ドロップ操作が実行された場合、リカバリー可能なデータベースが既にバックアップ・ペンディング状態に入っている。追加およびドロップ操作は、データベースのバックアップが完了するまで許可されません。
 - データベースが、バックアップ・ペンディング、リストア・ペンディング、ロールフォワード・ペンディングのいずれかの状態にあるのに、クラスター・トポロジーを変更しようとした (例えば、メンバーを追加または削除しようとした)。

問題の解決

- 前のセッションが異常終了したためにデータベースがオフラインになっている場合、以下のアクションで応答します。

1. **RESTART DATABASE** コマンドを使用して、データベースを再始動します。パーティション・データベース・サーバー環境では、このコマンドはすべてのデータベース・パーティションで実行する必要があります。
- **db2ckupgrade** コマンドの実行時にこのエラーが検出された場合は、以下のアクションで応答します。
 1. データベースのクリーン・シャットダウンを実行します。
 2. シャットダウン後に **HADR** がデータベースで有効になっている場合は、そのデータベースに対して **STOP HADR** コマンドを実行します。
 3. **db2ckupgrade** コマンドを再実行します。
- **DB2 pureScale** 環境環境の場合のみ:
 1. 前のセッションが異常終了したために、この **DB2** メンバーのデータベースがオフラインになっている場合は、以下のアクションで応答します。
 - a. デフォルトでは、メンバー・クラッシュ・リカバリーが **DB2 pureScale** 環境環境で自動的に開始されるので、ユーザー・アクションは不要です。メンバー・クラッシュ・リカバリーが自動的に有効にならない場合は、**RESTART DATABASE** コマンドを発行して、この **DB2** メンバーに対してメンバー・クラッシュ・リカバリーを実行します。
 - b. 一部のデータベース操作は、あるメンバーが不整合でも他の整合メンバーに対しては完了することができます。メンバー・クラッシュ・リカバリー中にデータベースにアクセスするには、アクティブな **DB2** メンバーに接続します。この特定のメンバーにアクセスするには、メンバー・クラッシュ・リカバリーが完了するまで待機します。
 2. 前のセッションが異常終了したために **DB2 pureScale** 環境インスタンス全体にわたってデータベースがオフラインになっている場合は、リカバリーが完了するまでデータベースを使用できなくなることをユーザーに警告します。次のステップは、グループ・クラッシュ・リカバリーが自動的に有効になっているかどうかによって異なります。有効になっている場合 (デフォルト)、ユーザー・アクションは不要です。自動グループ・クラッシュ・リカバリーが有効になっていない場合は、以下のアクションで応答します。
 - a. **RESTART DATABASE** コマンドを実行してグループ・クラッシュ・リカバリーを実行します。
 - b. リカバリーが完了した後に、未確定トランザクションを持つ他のメンバーに対してメンバー・クラッシュ・リカバリーを実行します。
 3. バックアップ・ペンディング状態にあるデータベースをバックアップし、追加またはドロップ操作を再サブミットします。
 4. データベースがトポロジー変更と互換性のない状態にある場合は、この問題に対処してからトポロジー変更要求を再度試みます。

DB2 データベース・システムのインストールのトラブルシューティング

DB2 データベース製品のインストール中に問題が発生した場合、ご使用のシステムがインストール要件を満たしていることを確認し、インストールの共通問題のリストを検討してください。

手順

DB2 データベース・システムのインストール問題をトラブルシューティングするには、以下を行います。

- ご使用のシステムがインストール要件のすべてを満たしていることを確認します。
- ライセンス交付エラーが発生している場合、適切なライセンスが適用されていることを確認します。

『ナレッジ・コレクション: DB2 ライセンス問題』(技術情報:

<http://www.ibm.com/support/docview.wss?rs=71&uid=swg21322757>) のよくある質問のリストを検討します。

- 資料および DB2 技術サポート Web サイト (www.ibm.com/software/data/db2/support/db2_9/troubleshoot.html) で、インストール問題のリストを検討します。

次のタスク

これらの手順を実行しても問題の原因を突き止めることができない場合には、さらに情報を取得するために診断データの収集を開始してください。

インストール問題についてのデータの収集

インストール問題が発生しているものの、問題の原因が判別できない場合、問題を診断して解決するために、あなた自身または IBM ソフトウェア・サポートが使用できる診断データを収集してください。

インストール問題の診断データを収集するには、以下のようにします。

1. オプション: トレースを有効にしてインストールの試行を繰り返します。以下に例を示します。

Linux および UNIX オペレーティング・システムの場合

```
db2setup -t /filepath/trace.out
```

Windows オペレーティング・システムの場合

```
setup -t %filepath%trace.out
```

2. インストール・ログ・ファイルを見つけます。

- Windows では、デフォルトのファイル名は `DB2-ProductAbbreviation-DateTime.log` です。例えば、`DB2-ESE-Wed Jun 21 11_59_37 2006.log` のようになります。インストール・ログのデフォルトのロケーションは、`"My Documents"%DB2LOG%` ディレクトリーです。

注: Microsoft Systems Center Configuration Manager (SCCM) や Tivoli® などを使い、SYSTEM ユーザーを使用して DB2 をインストールすると、インストール・ログは作成されません。SYSTEM ユーザーには、インストール・ログの作成場所の "My Documents" フォルダがありません。インストール・ログを表示するには、セットアップ・コマンドの `-1` オプションを使用して、別の場所にログ・ファイルを作成します。

- Linux および UNIX では、デフォルトのファイル名は、`db2setup.log`、`db2setup.his`、および `db2setup.err` です。

トレース (またはデバッグ・モード) を有効にした状態で問題が再現される場合、dascrt.log、dasdrop.log、dasupdt.log、db2icrt.log.PID、db2idrop.log.PID、db2iupgrade.log.PID、db2iupdt.log.PID などの追加ファイルが作成される場合があります。ここで、PID はプロセス ID です。

これらすべてのファイルのデフォルトのロケーションは /tmp ディレクトリーです。トレース・ファイル (trace.out) については、特に指定がないかぎりデフォルトのディレクトリーは存在しないため、ファイル・パスに、トレース出力ファイルが作成されたフォルダーを指定する必要があります。

3. オプション: データを IBM ソフトウェア・サポートに送信する場合、DB2 のデータも収集してください。追加情報については、『DB2 データの収集』を参照してください。

インストールの問題に関するデータの分析

インストールの問題に関する診断データを収集したら、データを分析して、問題の原因を判別できます。これらのステップはオプションです。問題の原因が簡単に判明しない場合は、データを IBM ソフトウェア・サポートに送信してください。

始める前に

以下のステップでは、インストール問題についてのデータの収集で説明されているファイルを入手済みであることを前提としています。

手順

1. 適切なインストール・ログ・ファイルを参照していることを確認します。ファイルの作成日や、ファイル名に含まれているタイム・スタンプ (Windows オペレーティング・システムの場合) を確認します。
2. インストールが正常に完了したかどうかを判別します。

- Windows オペレーティング・システムの場合、正常に完了したことは、インストール・ログ・ファイルの末尾にある、以下に類似したメッセージによって示されます。

```
Property(C): INSTALL_RESULT = Setup Complete Successfully
=== Logging stopped: 6/21/2006 16:03:09 ===
MSI (c) (34:38) [16:03:09:109]:
Product: DB2 Enterprise Server Edition - DB2COPY1 -- Installation operation
completed successfully.
```

- Linux および UNIX オペレーティング・システムの場合、正常に完了したことは、インストール・ログ・ファイル (デフォルトの名前は db2setup.log) の末尾にあるメッセージによって示されます。
3. オプション: エラーが発生したかどうかを判別します。インストールが正常に完了したものの、インストール・プロセス中にエラー・メッセージを受け取った場合、そのエラーをインストール・ログ・ファイル内で見つけます。

- Windows オペレーティング・システムの場合、大部分のエラーは先頭に ERROR: または WARNING: が付いています。以下に例を示します。

```
1: ERROR:An error occurred while running the command
"D:¥IBM¥SQLLIB¥bin¥db2.exe
CREATE TOOLS CATALOG SYSTOOLS USE EXISTING DATABASE TOOLSDB FORCE" to
initialize and/or migrate the DB2 tools catalog database.
The return value is "4".
```

1: WARNING:A minor error occurred while installing "DB2 Enterprise Server Edition - DB2COPY1" on this computer. Some features may not function correctly.

- Linux および UNIX オペレーティング・システムでは、Java によってエラーが戻された場合 (例えば、例外やトラップ情報)、デフォルト名が `db2setup.err` というファイルが存在します。

インストール・トレースを有効にしてあった場合は、インストール・ログ・ファイルに含まれる項目は多くなり、より詳細なものになります。

タスクの結果

このデータを分析しても問題解決の助けにならない場合、IBM ソフトウェア・サポートとの保守契約があれば、問題報告書をオープンできます。IBM ソフトウェア・サポートでは、収集したデータの送信を依頼します。また、実行した分析についてお尋ねすることもあります。

次のタスク

調査しても問題が解決しない場合は、データを IBM ソフトウェア・サポートに送信してください。

インスタンス作成の問題に関する診断情報を収集する

インスタンスの作成かアップグレードの際に問題が発生した場合、問題のトラブルシューティングに役立つ診断情報を収集する必要性が生じる可能性があります。

`db2fodc -c1p` コマンドは、IBM サポートが問題を分析してトラブルシューティングするのに必要な診断情報を即時に集めるのに役立ちます。

症状

インスタンスの作成中かアップグレード中に、インスタンスの作成の一部として DBM CFG を更新しようとする、エラーを受け取ることがあります。受け取る可能性のあるエラー・コードは DBI1281E です。しかし、このエラーは問題の根本原因を示していないことがあり、この場合はインスタンス作成の問題をさらにトラブルシューティングするために診断情報が必要になります。

問題の解決

問題のトラブルシューティングに必要な診断情報を即時に集めるには、IBM サポートの指導に従って、以下のステップを実行する必要があります。

1. `db2fodc -c1p` コマンドを発行します。このコマンドは、インスタンス作成の問題の診断を目標にして、環境と構成に関連した情報を収集します。収集の完了後に、`FODC_C1p_<timestamp>_<member>` という名前の新しく作成されたディレクトリーに情報が保管されます。

収集の完了後に、`FODC_C1p_<timestamp>_<member>` ディレクトリーを IBM サポートに送信して分析してもらうことができます。

認識済みの問題と解決策

システム WPAR (AIX) 上のデフォルト・パスに非 root ユーザーとして DB2 データベース製品をインストールする際のエラー

AIX 6.1 のシステム・ワークロード・パーティション (WPAR) 上のデフォルト・インストール・パス (/opt/IBM/db2/V9.7) に非 root ユーザーとして DB2 データベース製品をインストールすると、種々のエラーが発生する可能性があります。こうした問題を回避するには、WPAR に対してのみアクセス可能なファイル・システム上に DB2 データベース製品をインストールします。

症状

DB2 データベース製品をシステム WPAR の /usr または /opt ディレクトリーにインストールすると、ディレクトリーの構成方法によっては様々なエラーが発生する可能性があります。システム WPAR を、/usr および /opt ディレクトリーをグローバル環境と共有するように構成することもできれば (この場合、/usr および /opt ディレクトリーは WPAR から読み取り可能ですが、書き込みアクセスは不可になります)、/usr および /opt ディレクトリーのローカル・コピーを持つように構成することもできます。

最初のシナリオの場合に DB2 データベース製品をグローバル環境のデフォルト・パスにインストールすると、システム WPAR にこのインストール環境が表示されます。このため DB2 は WPAR にインストールされたかのように思われますが、DB2 インスタンスを作成しようとする試みは次のエラーを生じさせることとなります。DBI1288E プログラム db2icrt の実行に失敗しました。このプログラムが失敗したのは、ディレクトリーまたはファイル /opt/IBM/db2/V9.7/profiles.reg,/opt/IBM/db2/V9.7/default.env に対する書き込み権限がないためです。

2 番目のシナリオで DB2 データベース製品をグローバル環境のデフォルト・パスにインストールしてから、WPAR で /usr および /opt ディレクトリーのローカル・コピーを作成すると、DB2 データベース製品のインストール環境もコピーされます。これによって、システム管理者がこのデータベース・システムを用いようとする、予期しない問題が生じる可能性があります。DB2 データベース製品は別のシステムを対象としていたので、不正確な情報がコピーされる場合があります。例えば、もともとはグローバル環境で作成された DB2 インスタンスが WPAR に存在するかのように見えることがあります。これは、実際にどのインスタンスがシステムにインストールされているかをシステム管理者が判別する上で混乱の原因となります。

原因

こうした問題の原因は、DB2 データベース製品をシステム WPAR の /usr または /opt ディレクトリー上にインストールすることにあります。

問題の解決

DB2 データベース製品を、グローバル環境のデフォルト・パスにインストールしないでください。

WPAR に対してのみアクセス可能なファイル・システムをマウントし、そのファイル・システムに DB2 データベース製品をインストールします。

ベータ版および非ベータ版の DB2 データベース製品は共存できない

1 つの DB2 のコピーには、1 つ以上の異なる DB2 データベース製品を含めることができますが、ベータ版と非ベータ版の製品を含めることはできません。同じ場所にベータ版および非ベータ版の DB2 データベース製品をインストールしないでください。

この制限は、クライアントおよびサーバー・コンポーネント両方の DB2 データベース製品に適用されます。

問題の解決

バージョン 9.7非ベータ版の DB2 をインストールするか、異なるインストール・パスを選択する前にベータ版の DB2 をアンインストールしてください。

DB2 データベース製品をインストールする際のサービス名エラーの解決

DB2 データベース製品または DB2 インフォメーション・センターで使用するデフォルトでないサービス名またはポート番号を選択する場合は、指定する値が既使用中でないことを確認します。

症状

DB2 データベース製品または DB2 インフォメーション・センターのインストールを試みる際、DB2 セットアップ・ウィザードでは、「指定されたサービス名は使用中です」というエラーが報告されます。

原因

インストールする際には、DB2 セットアップ・ウィザードでは、ポート番号およびサービス名を選択するようにプロンプトが出されます。

- DB2 インフォメーション・センター
- クライアントからの TCP/IP コミュニケーションを受け入れる DB2 データベース製品
- データベース・パーティション・サーバーとして作業を行う DB2 データベース製品

このエラーは、デフォルト値を受け入れるのではなく、サービス名およびポート番号をご自身で選択する場合に発生する可能性があります。システムのサービス・ファイルに既に存在するサービス名を選択して、ポート番号のみを変更する場合、このエラーが生じます。

問題の解決

以下のアクションのいずれかを行います。

- デフォルト値を使用します。
- 既に services ファイルに存在するサービス名およびポート番号を使用します。

- 未使用のサービス名および未使用のポート番号を `services` ファイルに追加します。DB2 セットアップ・ウィザードで、これらの値を指定します。

ライセンス発行のトラブルシューティング

DB2 ライセンス準拠レポートの分析

ご使用の DB2 フィーチャーのライセンス準拠を確認するために、DB2 ライセンス準拠レポートを分析します。

始める前に

以下の各ステップでは、`db2licm` コマンドを使用して DB2 ライセンス準拠レポートを生成済みであることを想定しています。

手順

1. DB2 ライセンス準拠レポートが含まれたファイルを開きます。
2. 準拠性レポート内の各 DB2 フィーチャーの状況を調べます。レポートには、フィーチャーごとに以下の値のいずれかが表示されます。

準拠 違反がなにも検出されなかったことを示します。フィーチャーが使用されていて、適切にライセンス交付を受けています。

使用していません

この特定のフィーチャーを必要とするアクティビティをなにも実行していないことを示します。

違反 フィーチャーがライセンス交付を受けずに使用されていることを示します。

3. 違反が存在する場合は、`db2licm -l` コマンドを使用してライセンス情報を表示します。

DB2 フィーチャーが「ライセンスなし」の状況でリストされた場合は、そのフィーチャーのライセンスを取得する必要があります。ライセンス・キーとそれを登録するための説明は、DB2 フィーチャーの購入時に受け取るアクティベーション CD で入手できます。

注: DB2 Workgroup Server Edition および DB2 Express® Edition では、SAMPLE データベースにマテリアライズ照会表 (MQT) とライセンス違反の原因となるマルチディメンション・クラスタリング表 (MDC) が含まれています。この違反は、DB2 Enterprise Server Edition にアップグレードするだけで除去できます。

4. 以下のコマンドを使用して、ご使用の DB2 データベース製品内のどのオブジェクトまたは設定がライセンス違反を引き起こしているかを判別します。

- DB2 Advanced Access Control Feature の場合:

ラベル・ベースのアクセス制御 (LBAC) を使用する表が存在するかどうかを確認します。DB2 コピー内のすべてのインスタンスごとのすべてのデータベースに対して以下のコマンドを実行します。


```
SELECT TABSCHEMA, TABNAME
FROM SYSCAT.TABLES
WHERE SECPOLICYID>0
```

• DB2 Performance Optimization Feature の場合:

- マテリアライズ照会表が存在するかどうかを確認します。DB2 コピー内のすべてのインスタンスごとのすべてのデータベースに対して以下のコマンドを実行します。

```
SELECT OWNER, TABNAME
FROM SYSCAT.TABLES WHERE TYPE='S'
```

- マルチディメンション・クラスター表が存在するかどうかを確認します。DB2 コピー内のすべてのインスタンスごとのすべてのデータベースに対して以下のコマンドを実行します。

```
SELECT A.TABSCHEMA, A.TABNAME, A.INDNAME, A.INDSCHEMA
FROM SYSCAT.INDEXES A, SYSCAT.TABLES B
WHERE (A.TABNAME=B.TABNAME AND A.TABSCHEMA=B.TABSCHEMA)
AND A.INDEXTYPE='BLOK'
```

- ご使用のいずれかのインスタンスが照会並列処理 (照会間並列処理ともいう) を使用しているかどうかを確認します。DB2 コピー内のインスタンスごとに以下のコマンドを 1 回実行します。

```
SELECT NAME, VALUE
FROM SYSIBMADM.DBMCFG
WHERE NAME IN ('intra_parallel')
```

- 接続コンセントレーターが使用可能になっているかどうかを確認してください。DB2 コピー内の各インスタンスに対して以下のコマンドを実行します。

```
db2 get dbm cfg
```

このコマンドは、MAX_CONNECTIONS および MAX_COORDAGENTS を含むデータベース・マネージャー構成パラメーターの現行値を表示します。MAX_CONNECTIONS の値が MAX_COORDAGENTS の値より大きい場合、接続コンセントレーターは有効になっています。DB2 Enterprise Server Edition、DB2 Advanced Enterprise Server Edition、DB2 Connect Server 製品のいずれも使用していない場合は、接続コンセントレーターが無効になっていることを確認してください。これは、接続コンセントレーターが DB2 Enterprise Server Edition、DB2 Advanced Enterprise Server Edition、または DB2 Connect Server 製品でのみサポートされているためです。

• DB2 Storage Optimization Feature の場合:

- いずれかの表で行レベルの圧縮を使用可能にしているかどうかを確認します。DB2 コピー内のすべてのインスタンスごとのすべてのデータベースに対して以下のコマンドを実行します。

```
SELECT TABSCHEMA, TABNAME
FROM SYSCAT.TABLES
WHERE COMPRESSION IN ('R', 'B')
```

- 行レベルの圧縮が非アクティブである表に対してコンプレッション・ディクショナリーがまだ存在しているかどうかを確認します。DB2 コピー内のすべてのインスタンスごとのすべてのデータベースに対して以下のコマンドを実行します。

```
SELECT TABSCHEMA, TABNAME
FROM SYSIBMADM.ADMINTABINFO
WHERE DICTIONARY_SIZE <> 0 OR XML_DICTIONARY_SIZE <> 0
```

注: この照会はリソースを多く消費する可能性があり、時間がかかることがあります。この照会は、行レベルの圧縮が使用可能になっている表がないのに、Storage Optimization のライセンス違反が報告されている場合にのみ実行してください。

ロッキング問題の診断および解決

ロッキング問題を解決するには、SQL 照会パフォーマンスの低下、または照会完了の失敗、および関係する SQL ステートメントまたはステートメントの原因となっているロック・イベントのタイプを診断することにより開始する必要があります。ここでは、ロッキング問題のタイプを診断するのに役立つステップ、およびロッキング問題を解決するのに役立つステップについて説明します。

概要

タスクを完了する際にアプリケーションの失敗が生じている場合、またはロックのために SQL 照会のパフォーマンスの低下が生じている場合、ロッキング問題が起こっている可能性があります。したがって、理想的な目標は、アプリケーションがタスクを完了できない原因となる、いかなるロック・タイムアウトまたはデッドロックもデータベース・システムに置かないことです。

ロック待機は通常予期されるイベントですが、ロックの待機にかかった時間が長くなる場合は、SQL 照会パフォーマンスやアプリケーションの完了速度が低下する原因になる可能性があります。過度のロック待機期間はロック・タイムアウトになるリスクがあり、それが原因でアプリケーションがそのタスクが完了できなくなる可能性があります。

ロック・エスカレーションは、それがロック・タイムアウトを生じさせる原因となる場合、ロッキング問題としての考慮事項になります。いかなるロック・エスカレーションも生じさせないことが理想的な目標ですが、悪影響が生じないのであれば、ある程度のロック・エスカレーションは許容できます。

ロック待機、ロック・タイムアウト、およびデッドロック・ロッキング・イベントを常にモニターすることをお勧めします。通常、ロック待機のワークロード・レベル、およびロック・タイムアウトおよびデッドロックのデータベース・レベルでモニターすることをお勧めします。

生じているロッキング問題のタイプを診断して解決するには、情報の収集および診断標識の検索から開始します。以下のセクションでは、このプロセスについて説明します。

情報収集

一般に、処理の遅延やローパフォーマンスなどのシステムの異常動作が起きていることを客観的に評価できるようにするためには、システムの標準動作（ベースライン）を記述した情報を入手する必要があります。その後、疑わしい異常動作の観察とベースラインを比較できます。トラブルシューティング・プロセスで重要な

要素となるのは、定期的な操作モニター・タスクをスケジュールすることにより、ベースライン・データを収集することです。システムのベースライン操作の確立について詳しくは、「問題判別およびデータベース・パフォーマンスのチューニング」の『システム・パフォーマンスの操作モニター』を参照してください。

SQL 照会パフォーマンスの低下または照会完了の失敗の原因になっているロッキング問題のタイプを確認するには、関係しているロック・イベントのタイプ、このロックを要求または保留しているアプリケーション、このイベント中のアプリケーションの動作、および著しい低下に関係している SQL ステートメントまたはステートメントを識別するのに役立つ情報を収集する必要があります。

ロッキング・イベント・モニターの作成、表関数の使用、または **db2pd** コマンドの使用により、このタイプの情報を収集することができます。ロッキング・イベント・モニターにより集められた情報は、3 つのメイン・カテゴリーに分類できます。

- 当該のロックに関する情報
- このロックを要求しているアプリケーションおよびその現在のアクティビティーに関する情報。デッドロックの場合、これはビクティム (デッドロックの犠牲者) として参照されるステートメントに関する情報です。
- ロックを所有しているアプリケーションおよびその現在のアクティビティーに関する情報。デッドロックの場合、これはデッドロックの参加者として参照されるステートメントに関する情報です。

ロック待機、ロック・タイムアウト、およびデッドロックの各ロック・イベントをモニターする方法については、「データベースのモニタリング ガイドおよびリファレンス」の『ロック・イベントのモニター』を参照してください。

診断標識の検索

ロッキング・イベント・モニター、表関数、または **db2pd** コマンドの実行により、ロッキング問題の性質を分離するのに役立つ可能性のある情報を収集することができます。特に、以下のトピックには、発生している特定のタイプのロッキング問題を診断して確認するために役立つ、診断標識情報が含まれます。

- 長い待機時間が発生していて、ロック・タイムアウトではない場合、ロック待機問題である可能性があります。確認するには、640 ページの『ロック待機問題の診断』を参照してください。
- ベースライン数より多いデッドロック数が発生している場合、デッドロック問題である可能性があります。確認するには、643 ページの『デッドロック問題の診断』を参照してください。
- ロック・タイムアウト数の増加が発生していて、**locktimeout** データベース構成パラメーターがゼロ以外の時刻値に設定されている場合、ロック・タイムアウトの問題である可能性があります。それを確認するには (またはロック待機の問題について考慮するには)、646 ページの『ロック・タイムアウト問題の診断』を参照してください。
- 通常のロック待機数より高い数値が発生していて、ロッキング・イベント・モニターによりロック・エスカレーションが生じていることを示されている (はい) 場合、ロック・エスカレーション問題である可能性があります。確認するには、649 ページの『ロック・エスカレーション問題の診断』を参照してください。

ロック待機問題の診断

あるトランザクションが、別のトランザクションによってすでに保留されているリソースのロックを取得しようとするとき、ロック待機が発生します。ロック待機期間が延長される場合は、これにより SQL 照会実行が低下することになります。ロック待機時間が長い、想定外であり、ロック・タイムアウトが発生しない場合は、ロック待機問題がよく発生します。

始める前に

一般に、処理の遅延やローパフォーマンスなどのシステムの異常動作が起きていることを客観的に評価できるようにするためには、システムの標準動作（ベースライン）を記述した情報を入手している必要があります。その後、疑わしい異常動作の観察とベースラインを比較できます。定期的な運用モニター・タスクをスケジュールしてベースライン・データを収集することは、トラブルシューティング・プロセスのキー・コンポーネントです。システムのベースライン操作の確立について詳しくは、13 ページの『システム・パフォーマンスの操作モニター』を参照してください。

ロック待機ロック・イベントをモニターする方法の指示については、「データベースのモニタリング ガイドおよびリファレンス」の『ロック・イベントのモニター』を参照してください。

このタスクについて

診断 あるトランザクション (1 つ以上の SQL ステートメントで構成) が、別のトランザクションによって保留されているロックに対してモードが競合するロックを取得しようとするとき、ロック待機が発生します。ロック待機時間が長すぎると応答時間が多くの場合に長くなるので、モニターすることが重要です。通常、1 つのトランザクションにおけるロック待機時間は非常に短く、正規化された測定は処理が簡単であるので、ロック待機時間は 1000 トランザクションに最適に正規化されます。

各品質の診断を確認しようとするときに考慮する必要があるロック待機にはさまざまな品質があります。以下のリストは、ロック待機の 3 つの異なる品質と、その品質を診断する最適な方法を示しています。

- 長い、個別のロック待機
 - サービス・クラスとワークロードからピークのロック待機時間をチェックしてください。ワークロードのロック・イベント・モニターをセットアップして、その値を取得してください。
- 長いロック待機時間、短い個別のロック待機
 - 通常、ロック案内の結果 `db2pd -locks wait` コマンドを使用して、待機チェーンを検出してください。
- 待機中のロックのタイプ
 - これをチェックすることで、問題を判断できる場合があります。ロック・タイプの情報を取得するために、ロック上で待機しているエージェントを検索してください。ロック・タイプ情報を使用して、見てすぐ分かることが発生しているか判断してください。例えば、パッケージ・ロックにより、そのパッケージのユーザーと競合する `BIND/REBIND`

コマンドまたは DDL が示され、内部 c (カタログ・キャッシュ) ロックにより、ステートメント・コンパイルと競合する DDL が示される可能性があります。

標識サイン

以下のようなロック待機の標識サインを見つけてください。

- ロック待機数が増加しています (**lock_waits** モニター・エレメント値の増加)
- ロック上で待機するアクティブなエージェントの高いパーセント (例: アクティブなエージェントの合計の 20% 以上)。この情報の取得方法の詳細については、次のセクション『モニター対象』を参照してください。
- データベースまたはワークロード・レベルで取り込まれたロック待機時間 (**lock_wait_time** モニター・エレメント) の増加値

モニター対象

他の各種 DB2 モニター・データの多くとは異なり、ロック情報は極めて一時的なものです。現在の合計を示す **lock_wait_time** は別として、他のロック情報のほとんどは、ロック自体が解放されると無くなります。したがって、ロックおよびロック待機イベント・データを、一定の時間にわたり定期的に収集するなら、進展中の状況をより良く理解するのに非常に役立ちます。

ロック上で待機するアクティブなエージェントの情報を収集するには、**WLM_GET_SERVICE_CLASS_AGENTS** 表関数を使用してください。ロックを待っているエージェントは、以下の属性値ペアがあるエージェントによって示されます。

- **EVENT_OBJECT = LOCK**
- **EVENT_TYPE = ACQUIRE**

アプリケーション・スナップショット、ロック管理ビュー、または **db2pd -wlocks** コマンドのロック待機オプションを使用しても、ロック上で待機するアクティブなエージェントについての情報を取得できます。

重要な標識となるモニター・エレメントは以下のとおりです。

- **lock_waits** 値は増加しています
- 大きな **lock_wait_time** 値

ここにリストする 1 つ以上の標識サインが確認されている場合、ロック待機の問題がよく発生します。『次の作業』セクションのリンクに従い、この問題を解決してください。

次のタスク

ロック待機が発生している問題の原因である可能性があることを診断した後、問題 642 ページの『ロック待機問題の解決』を解決するための手順を行ってください。

ロック待機問題の解決

次のステップは、ロック待機問題を診断後、長時間ロックを待つ必要があるアプリケーションで発生する問題を解決しようとすることです。ロック待機問題を解決でき、今後このような問題が発生するのを防ぐにあたって支援できる指針がここに記載されています。

始める前に

638 ページの『ロッキング問題の診断および解決』で概略されているロック問題について必要な診断手順を行って、ロック待機問題が発生していることを確認してください。

このタスクについて

ここに記載されている指針は、発生しているロック待機問題を解決し、今後このような問題が発生するのを防ぐのに役立ちます。

手順

受諾不能なロック待機問題の原因を診断し、対処するには、以下のステップに従ってください。

1. エージェントが長時間ロックを待っているすべての表に関する情報を、管理通知ログから取得します。
2. 管理通知ログ内の情報を使用して、ロック待機問題の解決方法を判別します。ロック競合およびロック待機時間の削減に役立つ、いくつかの指針があります。以下の選択肢を考慮してください。
 - 非常に長いトランザクションおよび **WITH HOLD** カーソルを、できるだけ避けてください。ロックが長く保留されるほど、他のアプリケーションと競合する可能性は高くなります。これは、高い分離レベルを使用している場合のみ問題です。
 - すぐに以下のアクションをコミットすることがベスト・プラクティスです。
 - 削除、挿入、および更新などの書き込みアクション
 - ALTER、CREATE、および DROP ステートメントなどのデータ定義言語 (DDL) ステートメント
 - **BIND** および **REBIND** コマンド
 - ALTER または DROP DDL ステートメントを実行後、**SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS** プロシージャを実行して、データ・オブジェクトと **db2rbind** コマンドをもう一度妥当性検査し、パッケージを再バインドしてください。
 - 必要以上に長い結果セットのフェッチを避けてください。特に、反復可能読み取り (RR) 分離レベル下では避けてください。行が多く関わるほど、保留されるロックが多くなり、他のユーザーによって保留されているロックと衝突する可能性が高まります。多くの場合、実際に行うことは、多数の行を戻してアプリケーションでフィルタリングする代わりに、行選択基準を **SELECT** ステートメントの **WHERE** 節内に移動することです。例えば、

```
exec sql declare curs for
select c1,c2 from t
where c1 not null;
```

```

exec sql open curs;
do {
  exec sql fetch curs
    into :c1, :c2;
} while( P(c1) != someVar );

```

==>

```

exec sql declare curs for
  select c1,c2 from t
  where c1 not null
  and myUdfP(c1) = :someVar;
exec sql open curs;
exec sql fetch curs
  into :c1, :c2;

```

- 必要以上に高い分離レベルの使用を避けてください。アプリケーション内で結果セットの保全性を保持するため、反復可能読み取りが必要な場合があります。しかしこれは、ロック保留、および起こりうるロック競合という点で余分の負担となります。
- アプリケーションでのビジネス・ロジック上適切であれば、**DB2_EVALUNCOMMITTED**、**DB2_SKIPDELETED**、および **DB2_SKIPINSERTED** レジストリー変数によるロック動作の変更を検討してください。これらのレジストリー変数により、DB2 データベース・マネージャーが状況によって、ロックをかけるのを遅らせたり回避したりすることが可能となります。こうして、競合が削減されますので、スループット改善の可能性がります。
- 可能な限り、ロック・エスカレーションを除去してください。

次のタスク

1 つまたは複数のアプリケーションを再実行し、管理通知ログでロック関連項目をチェックするか、適切なワークロード、接続、サービス・サブクラス、作業単位、およびアクティビティ・レベルのロック待機およびロック待機時間メトリックをチェックして、ロック問題が除去されたことを確認してください。

デッドロック問題の診断

デッドロックは、2 つのアプリケーションが他方が必要とするデータをロックし、その結果、いずれのアプリケーションもデッドロック検出機能を介入させないと実行を継続できないときに作成されます。デッドロックは、デッドロックの検出を待機中の参加者のトランザクション速度を低下させ、ピクティム・トランザクションをロールバックしてシステム・リソースを浪費し、処理全体における余分なシステム処理およびトランザクションのログ・アクセスを引き起こします。デッドロック問題は、デッドロックの数がベースラインの数を超え、トランザクションが再実行される場合に発生することがあります。

始める前に

一般に、見つかったいずれのデッドロックも異常と見なされます。処理の遅延やローパフォーマンスなどのシステムの異常動作が起きていることを客観的に評価できるようにするためには、システムの標準動作（ベースライン）を記述した情報を入手している必要があります。その後、疑わしい異常動作の観察とベースラインを比較

できます。定期的な運用モニター・タスクをスケジュールしてベースライン・データを収集することは、トラブルシューティング・プロセスのキー・コンポーネントです。

デッドロック・ロック・イベントをモニターする方法の指示については、「データベースのモニタリング ガイドおよびリファレンス」の『ロック・イベントのモニター』を参照してください。

このタスクについて

診断 デッドロックは、2 つのアプリケーションが他方が必要とするデータをロックし、その結果、いずれのアプリケーションもデッドロック検出機能を介入させないと実行を継続できないときに作成されます。ピクティム・アプリケーションは、直前にデッドロックされたトランザクションをシステムが自動的にロールバックした後、トランザクションを最初から再実行する必要があります。これが生じる比率をモニターすると、DBA が認識することなく、数多くのデッドロックがかなり余分の負荷をシステムに生じさせるのを回避するのに役立ちます。

標識サイン

以下のようなデッドロックの標識サインを見つけてください。

- 1 つ以上のアプリケーションが、トランザクション再実行することがある
- 管理通知ログのデッドロック・メッセージの項目
- **deadlocks** モニター・エレメントについて表示されたデッドロック回数の増加
- **int_deadlock_rollbacks** モニター・エレメントについて表示されたロールバック回数の増加
- **log_disk_wait_time** モニター・エレメントについて表示された、ログ・レコードがディスクにフラッシュされるのをエージェントが待機していた時間の増加

モニター対象

デッドロックのコストは一樣ではなく、ロールバック・トランザクションの長さに正比例します。それでも、いずれのデッドロックも、通常、問題があることを表します。

デッドロック・イベントを検出するには、基本的に次の 3 つの方法があります。

1. ロッキング・イベント・モニターおよび **mon_deadlock** データベース構成パラメーターを設定し、データベース全体で発生するすべてのデッドロック・イベントの詳細をキャプチャーする
2. 管理通知ログで、デッドロック・メッセージおよび付属する基本情報をモニターする

注: デッドロックのメッセージを管理通知ログ・ファイルに書き込めるようにするには、**mon_lck_msg_lvl** データベース構成パラメーターの値を 2 に設定します。

3. 表関数で、標識モニター・エレメントをモニターする

ほとんどのユーザーは、最初の方法を採用します。重要な標識となるモニター・エレメントをモニターしてデッドロックの発生時期を検出すると、ユーザーは、イベント・モニターで収集した情報を確認することで情報の詳細を取得できます。

重要な標識となるモニター・エレメントは以下のとおりです。

- **deadlocks** 値がゼロ以外
- **int_deadlock_rollbacks** が、デッドロック・イベントによるロールバック回数の増加を示している
- **log_disk_wait_time** が、ログがディスクにフラッシュされるのをエージェントが待機していた時間の増加を示している

ここにリストされた標識サインが 1 つ以上見つかった場合、デッドロック問題が発生している可能性があります。『次の作業』セクションのリンクに従い、この問題を解決してください。

次のタスク

問題の起きている原因がデッドロックにあるようだと診断されたら、対策を講じて問題を解決します。『デッドロック問題の解決』

デッドロック問題の解決

次のステップは、デッドロック問題を診断後、互いに他方のアプリケーションで必要なリソースをロックしている、並行して実行中の 2 つのアプリケーション間で発生するデッドロック問題を解決しようとすることです。ここに記載されている指針は、発生しているデッドロック問題を解決し、今後このような問題が発生するのを防ぐのに役立ちます。

始める前に

638 ページの『ロッキング問題の診断および解決』で概略されているロック問題について必要な診断手順を行って、デッドロック問題が発生していることを確認してください。

このタスクについて

ここに記載されている指針は、発生しているデッドロック問題を解決し、今後このような問題が発生するのを防ぐのに役立ちます。

手順

受諾不能なデッドロック問題の原因を診断し、対処するには、以下のステップに従ってください。

1. エージェントでデッドロックが発生しているすべての表に関する情報を、ロック・イベント・モニターまたは管理通知ログから取得します。
2. 管理通知ログ内の情報を使用して、デッドロック問題の解決方法を判別します。ロック競合およびロック待機時間の削減に役立つ、いくつかの指針があります。以下の選択肢を考慮してください。
 - 各アプリケーション接続では、独自の行のセットが処理され、ロック待機が回避される必要があります。

- 全アプリケーションが共通データに同じ順番でアクセスするようにすることで、デッドロックの頻度を低減できることがあります。例えば、全アプリケーションが、表 A の行、表 B の行、表 C の行といった順序でアクセス (つまりロックも) するようにします。2 つのアプリケーションが、同じオブジェクトに対して、両立しないロックを異なる順番でかけるならば、デッドロックのリスクはかなり大きくなります。
 - ロック・タイムアウトは、デッドロックとたいして変わりません。なぜなら、両方ともトランザクションのロールバックを引き起こすからです。しかし、デッドロック数を最小化する必要がある場合、潜在的な関連デッドロックが検出される前に、通常、ロック・タイムアウトが発生するようにすることでデッドロック数を最小化できます。そうするには、**locktimeout** データベース構成パラメーターの値 (秒単位) を **dlchktime** データベース構成パラメーターの値 (ミリ秒単位) よりもかなり短く設定してください。そうではなく、**locktimeout** が **dlchktime** インターバルよりも長い場合は、デッドロック状態が始まった後にデッドロック検出機能がウェイクアップし、ロック・タイムアウトが発生する前にデッドロック検出機能でデッドロックを検出できる可能性があります。
 - 可能であれば、並行 DDL 操作はしないようにします。例えば、表自体の他に、表索引、主キー、チェック制約などに対して行が削除される必要がある場合があるので、DROP TABLE ステートメントにより多数のカタログ更新が行われる可能性があります。他の DDL 操作によりオブジェクトが廃棄または作成されている場合、ロック競合が起こる可能性があり、デッドロックさえも時々起こる可能性があります。
 - すぐに以下のアクションをコミットすることがベスト・プラクティスです。
 - 削除、挿入、および更新などの書き込みアクション
 - ALTER、CREATE、および DROP などのデータ定義言語 (DDL) ステートメント
 - BIND および REBIND コマンド
3. デッドロック検出機能では、以下の状態を検出したり、解決したりできないので、アプリケーション設計でこうしたことに対してガードする必要があります。アプリケーション、特にマルチスレッド化されたアプリケーションでは、DB2 ロック待機、およびセマフォなど、DB2 ソフトウェアの外部のリソースの待機を伴うデッドロックが生じる場合があります。例えば、接続 A は接続 B によって保留されるロックを待っている場合があり、接続 B は接続 A によって保留されるセマフォを待っている場合があります。

次のタスク

アプリケーションを再実行し、管理通知ログでロック関連の項目をチェックして、ロック問題が除去されたことを確認してください。

ロック・タイムアウト問題の診断

リソースのロックを待つトランザクションの待ち時間が、**locktimeout** データベース構成パラメーターで指定された待ち時間の値を超えると、ロック・タイムアウトが発生します。このロック・タイムアウトでは時間が消費され、SQL 照会パフォーマンスの低下の原因となります。ロック・タイムアウト問題は、ロック・タイムア

ウトの発生回数が増え、**locktimeout** データベース構成パラメーターがゼロ以外の時刻値に設定されている場合に発生することがあります。

始める前に

一般に、処理の遅延やローパフォーマンスなどのシステムの異常動作が起きていることを客観的に評価できるようにするためには、システムの標準動作（ベースライン）を記述した情報を入手する必要があります。その後、疑わしい異常動作の観察とベースラインを比較できます。定期的な運用モニター・タスクをスケジュールしてベースライン・データを収集することは、トラブルシューティング・プロセスのキー・コンポーネントです。

ロック・タイムアウト・ロック・イベントをモニターする方法の指示については、「データベースのモニタリング ガイドおよびリファレンス」の『ロック・イベントのモニター』を参照してください。

このタスクについて

診断 ロック待機の状態は、トランザクションのロールバックの原因となるロック・タイムアウトにつながる場合があります。ロック待機がロック・タイムアウトになるまでの時間は、データベース構成パラメーター **locktimeout** で指定します。過剰な回数のロック・タイムアウトは、デッドロックと同様にシステムにダメージを及ぼします。ほとんどの実動システムでデッドロックは比較的まれなことです。ロック・タイムアウトはもっと一般的に生じます。通常、アプリケーションではこれらは同様の仕方で扱う必要があります。最初からトランザクションを再実行しなければなりません。これが生じる比率をモニターすると、DBA が認識することなく、数多くのロック・タイムアウトがかなり余分の負荷をシステムに生じさせるのを回避するのに役立ちます。

標識サイン

以下のようなロック・タイムアウトの標識サインを見つけてください。

- アプリケーションがトランザクションを頻繁に再実行している
- **lock_timeouts** モニター・エレメント値が上昇している
- 管理通知ログのロック・タイムアウト・メッセージの項目

モニター対象

イベントのロックには比較的一時的な性質があるので、ロック・イベントのデータを一定の時間にわたり定期的に収集すると、進展中の状況をより良く理解するのに非常に役立ちます。

管理通知ログでロック・タイムアウト・メッセージをモニターできます。

注: ロック・タイムアウトのメッセージを管理通知ログ・ファイルに書き込めるようにするには、**mon_lck_msg_lvl** データベース構成パラメーターの値を 3 に設定します。

イベント・モニターを作成し、ワークロードまたはデータベースのロック・タイムアウト・データを収集します。

重要な標識となるモニター・エレメントは以下のとおりです。

- **lock_timeouts** 値が上昇している
- **int_rollbacks** 値が上昇している

ここにリストされた標識サインが 1 つ以上見つかった場合、ロック・タイムアウト問題が発生している可能性があります。『次の作業』セクションのリンクに従い、この問題を解決してください。

次のタスク

問題の起きている原因がロック・タイムアウトにあるようだと診断されたら、対策を講じて問題を解決します。『ロック・タイムアウト問題の解決』

ロック・タイムアウト問題の解決

次のステップは、ロック・タイムアウト問題を診断後、ロック・タイムアウト期間が経過するまでロックを待っている 1 つまたは複数のアプリケーションで発生する問題を解決しようとすることです。ここに記載されている指針は、発生しているロック・タイムアウト問題を解決し、今後このような問題が発生するのを防ぐのに役立ちます。

始める前に

638 ページの『ロッキング問題の診断および解決』で概略されているロック問題について必要な診断手順を行って、ロック・タイムアウト問題が発生していることを確認してください。

このタスクについて

ここに記載されている指針は、発生しているロック・タイムアウト問題を解決し、今後このような問題が発生するのを防ぐのに役立ちます。

手順

受諾不能なロック・タイムアウト問題の原因を診断し、対処するには、以下のステップに従ってください。

1. エージェントでロック・タイムアウトが発生しているすべての表に関する情報を、ロック・イベント・モニターまたは管理通知ログから取得します。
2. 管理通知ログ内の情報を使用して、ロック・タイムアウト問題の解決方法を判別します。ロック・タイムアウト数を削減できる、ロック競合およびロック待機時間の削減に役立つ、いくつかの指針があります。以下の選択肢を考慮してください。
 - **locktimeout** データベース構成パラメーターをご使用のデータベース環境に適切な秒数に調整してください。
 - 非常に長いトランザクションおよび **WITH HOLD** カーソルを、できるだけ避けてください。ロックが長く保留されるほど、他のアプリケーションと競合する可能性は高くなります。
 - すぐに以下のアクションをコミットすることがベスト・プラクティスです。
 - 削除、挿入、および更新などの書き込みアクション
 - ALTER、CREATE、および DROP などのデータ定義言語 (DDL) ステートメント

- BIND および REBIND コマンド

- 必要以上に長い結果セットのフェッチを避けてください。特に、反復可能読み取り (RR) 分離レベル下では避けてください。行が多く関わるほど、保留されるロックが多くなり、他のユーザーによって保留されているロックと衝突する可能性が高まります。多くの場合、実際に行うことは、多数の行を戻してアプリケーションでフィルタリングする代わりに、行選択基準を SELECT ステートメントの WHERE 節内に移動することです。例えば、

```
exec sql declare curs for
  select c1,c2 from t
  where c1 not null;
exec sql open curs;
do {
  exec sql fetch curs
    into :c1, :c2;
} while( P(c1) != someVar );
```

==>

```
exec sql declare curs for
  select c1,c2 from t
  where c1 not null
  and myUdfP(c1) = :someVar;
exec sql open curs;
exec sql fetch curs
  into :c1, :c2;
```

- 必要以上に高い分離レベルの使用を避けてください。アプリケーション内で結果セットの保全性を保持するため、反復可能読み取りが必要な場合があります。しかしこれは、ロック保留、および起こりうるロック競合という点で余分の負担となります。
- アプリケーションでのビジネス・ロジック上適切であれば、**DB2_EVALUNCOMMITTED**、**DB2_SKIPDELETED**、および **DB2_SKIPINSERTED** レジストリー変数によるロック動作の変更を検討してください。これらのレジストリー変数により、DB2 データベース・マネージャーが状況によって、ロックをかけるのを遅らせたり回避したりすることが可能となります。こうして、競合が削減されますので、スループット改善の可能性がります。

次のタスク

1 つまたは複数のアプリケーションを再実行し、管理通知ログでロック関連項目をチェックするか、適切なワークロード、接続、サービス・サブクラス、作業単位、およびアクティビティー・レベルのロック待機およびロック待機時間メトリックをチェックして、ロック問題が除去されたことを確認してください。

ロック・エスカレーション問題の診断

ロック・エスカレーションは、ロック (ロック・スペース) に割り振られているメモリーを削減するために、多数の行レベルのロックが単一のメモリー節約型の表ロックにエスカレートするときに発生します。この状態は自動的に発生してロック専用のメモリー・スペースを節約しますが、並行性が、受諾不能なレベルにまで削減される場合もあります。ロック・エスカレーション問題は、ロック待機の数が増え、管理通知ログ項目にロック・エスカレーションが生じていることが示されている場合に発生することがあります。

始める前に

一般に、処理の遅延やローパフォーマンスなどのシステムの異常動作が起きていることを客観的に評価できるようにするためには、システムの標準動作（ベースライン）を記述した情報を入手している必要があります。その後、疑わしい異常動作の観察とベースラインを比較できます。定期的な運用モニター・タスクをスケジュールしてベースライン・データを収集することは、トラブルシューティング・プロセスのキー・コンポーネントです。システムのベースライン操作の確立について詳しくは、13 ページの『システム・パフォーマンスの操作モニター』を参照してください。

このタスクについて

診断 複数の行レベルのロックから単一表レベル・ロックへのロック・エスカレーションは、次の理由で発生する場合があります。

- 表に対して行われた多数の行レベル・ロックの合計メモリー消費量が、ロックの保管に割り当てられているメモリー合計量の比率を超えた
- ロック・リストがスペースを使い果たしたロック・リストの消耗の原因となったアプリケーションは、そのアプリケーションがロックを最も多く所有していなかったとしても、ロック・エスカレーション・プロセスによって強制的にロックされます。

ロックの保管に割り当てられているメモリー合計量のしきい値の比率（アプリケーションがこれを超えるとロック・エスカレーションが発生する）は **maxlocks** データベース構成パラメーターで定義され、ロックに割り当てられるメモリーは **locklist** データベース構成パラメーターで定義されます。適切に構成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。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った場合には、問題を分析し、最善の対処方法を見極めることができます。

メモリー・スペースの観点から見ると、自己調整メモリー・マネージャー（STMM）でメモリーのロックを管理している場合、ロック・エスカレーションはたいした問題ではありませんが、そうではない場合、ロックのメモリーは単に **locklist** データベース構成パラメーターで割り当てられていることとなります。STMM は、ロックが空いているメモリー・スペースを使い果たす時に、自動的にそのメモリー・スペースを調整します。

標識サイン

以下のようなロック・エスカレーションの標識サインを見つけてください。

- 管理通知ログのロック・エスカレーション・メッセージの項目

モニター対象

イベントのロックには比較的一時的な性質があるので、ロック・イベントのデータを一定の時間にわたり定期的に収集すると、進展中の状況をより良く理解するのに非常に役立ちます。

ロック・エスカレーションが、SQL 照会パフォーマンスを低下させる要因となっている場合があることの標識となる次のモニター・エレメントをチェックします。

- **lock_escals**

ここにリストされた標識サインが 1 つ以上見つかった場合、ロック・エスカレーション問題が発生している可能性があります。『次の作業』セクションのリンクに従い、この問題を解決してください。

次のタスク

問題の起きている原因がロック・エスカレーションにあるようだと診断されたら、対策を講じて問題を解決します。『ロック・エスカレーション問題の解決』

ロック・エスカレーション問題の解決

ロック・エスカレーション問題を診断したら、次のステップでは、データベース・マネージャーがロックを行レベルから表レベルに自動的にエスカレートさせたことよって起きた問題の解決を試みます。このガイドラインは、現在発生しているロック・エスカレーション問題の解決と、今後の発生防止に役立ちます。

始める前に

638 ページの『ロッキング問題の診断および解決』で概略されているロック問題について必要な診断手順を行って、ロック・エスカレーション問題が発生していることを確認してください。

このタスクについて

このガイドラインは、現在発生しているロック・エスカレーション問題の解決と、今後の発生防止に役立ちます。

目的は、ロック・エスカレーションを可能な限り最小限に抑える、または除去することです。正しいアプリケーション設計とロック処理のデータベース構成を組み合わせると、ロック・エスカレーションを最小限に抑える、または除去することができます。ロック・エスカレーションは、並行性の削減やロック・タイムアウトの可能性につながる場合があるので、ロック・エスカレーションへの対応は重要な作業です。`lock_escal` モニター・エレメントおよび管理通知ログに書き込まれたメッセージは、ロック・エスカレーションの識別と修正に使用できます。

まず、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。

`mon_lck_msg_lvl` データベース構成パラメーターの値を 1 (デフォルト設定) に設定します。ロック・エスカレーション・イベントが発生すると、ロック、ワークロード、アプリケーション、表、およびエラーの `SQLCODE` に関する情報が記録されます。照会が現在実行中の動的 SQL ステートメントであるなら、その照会も記録されます。

手順

受諾不能なロック・エスカレーション問題の原因を診断し、対処するには、以下のステップに従ってください。

1. ロックがエスカレートされたすべての表、および関係するアプリケーションに関する情報を、管理通知ログから収集します。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各表の表 ID および表名
 - 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 または X ロックが獲得されます。
 - 新しい表レベル・ロックの獲得に関連付けられた内部戻りコード
2. ロック・エスカレーションに関するアプリケーションについての管理通知ログの情報を使用し、エスカレーション問題の解決方法を決定します。以下の選択肢を考慮してください。
- **maxlocks** または **locklist** データベース構成パラメーター (あるいはその両方) を確認または調整します。パーティション・データベース・システムでは、すべてのデータベース・パーティションでこの変更を行ってください。**locklist** 構成パラメーターの値が、現行のワークロードには小さすぎる場合があります。複数のアプリケーションでロック・エスカレーションが発生している場合、このことが、ロック・リストのサイズを増大させる必要があることの標識となる場合があります。ワークロードの拡大または新規アプリケーションの追加が行われると、ロック・リストが小さくなり過ぎることがあります。1 つのみのアプリケーションにロック・エスカレーションが発生している場合は、**maxlocks** 構成パラメーターを調整することでこの問題を解決できます。ただし、**maxlocks** を増加するときには、同時に **locklist** の増加も考慮してください。1 つのアプリケーションでさらに多くのロック・リストを使用することが許可された場合、その他のすべてのアプリケーションで、ロック・リスト内で使用可能な残りのロックを使い果たし、エスカレーションが発生する場合がありますからです。
 - RR、RS、CS、または UR などの、アプリケーションおよび SQL ステートメントが実行中の場合の分離レベルを考慮してください。ロックは、COMMIT が発行されている間は保留されるため、RR 分離レベルおよび RS 分離レベルではさらに多くのエスカレーションを引き起こされる傾向があります。CS 分離レベルおよび UR 分離レベルでは、COMMIT が発行されるまではロックを保留しないので、ロック・エスカレーションは少なくなる傾向があります。アプリケーションで許容されるできるだけ低い分離レベルを使用します。
 - ビジネス・ニーズおよびアプリケーションの設計で許可される場合、アプリケーションでのコミットの頻度を増やします。コミットの頻度が増えると、任意の時点で存在するロックの数が減ります。こうすると、アプリケーションが、ロック・エスカレーションを引き起こす **maxlocks** 値に到達することや、すべてのアプリケーションがロック・リストを使い果たすことを防ぐのに役立ちます。
 - アプリケーションを変更し、LOCK TABLE ステートメントを使用して、表ロックを取得することができます。多数のアプリケーションおよびユーザーによる同時アクセスがそれほど重要ではない場合、この方法は表に関する良い戦略です。例えば、アプリケーションで永続的な作業表 (例えば DGTT 以外で、このアプリケーションのインスタンスにちなんだ固有の名前が付けられている) が使用されている場合です。この場合、表ロックの取得は良い戦略です。これにより、アプリケーションで保持されるロックの数が減り、作業表内でアクセスされる行で行ロックを取得したりリリースしたりする必要がなくなり、パフォーマンスが改善されるからです。

アプリケーションに作業表がなく、**locklist** 構成パラメーターまたは **maxlocks** 構成パラメーターの値を増やすことができない場合は、アプリケーションで表ロックを取得することができます。しかし、表またはロックする表を選択するには注意が必要です。多数のアプリケーションおよびユーザーにアクセスされる表は避けてください。こうした表のロックは、応答時間に影響を及ぼす並行性の問題や、最悪の場合は、アプリケーションでのロック・タイムアウトにつながるからです。

次のタスク

アプリケーションを再実行し、管理通知ログでロック関連の項目をチェックして、ロック問題が除去されたことを確認してください。

SQL パフォーマンスのトラブルシューティング

個別の SQL 照会は正常に実行されるが、複数の照会の実行時にパフォーマンスが低下する

グループとして実行すると遅い SQL 照会のパフォーマンスを改善するために実行できる手順がいくつかあります。

テスト環境で単独で実行すると正常に動作する照会が、実稼働環境で他の照会と同時に実行すると遅いことがあります。

症状

単独では正常に実行される SQL 照会を、実動システムで他の照会が同時に実行するとスローダウンします。これは、毎日特定の時間に発生する場合も、ランダムに発生しているように見える場合もあります。

SQL ステートメントを確認すると、偶発的なパフォーマンス上の問題の原因として明らかなものはありません。

原因

このようなタイプのパフォーマンスの問題は、ソート・ヒープ・メモリーの割り振りの問題の結果として発生することがよくあります。

ソート・ヒープ・メモリーはさまざまな照会 (そのリストは **sortheap** の文書に記載) で使用されるので、次の構成パラメーターの設定を考慮することが重要です。

- ソート・ヒープ・サイズ (**sortheap**)。これは、個別のソートごとに割り振られるメモリー量を指定します。
- ソート・ヒープしきい値 (**sheapthres**)。これは、そのインスタンス全体で実行されているすべてのソートで使用可能な合計メモリー量を指定します。

同時に実行している照会の数が増えると、システムが **sheapthres** しきい値に達し、パフォーマンスの問題が発生することがあります。

問題の解決

多くの場合、DB2 セルフチューニング・メモリー・マネージャー (STMM) は、ソート・メモリーの割り振りを調整して、照会の効率的な実行を維持できます。詳しくは、**self_tuning_mem** データベース構成パラメーターの文書を参照してください。システムで STMM を実行していない場合は、次のオプションを検討してください。

SQL の変更

SQL を変更することによって、使用されるソート・ヒープ・メモリーの量を削減できる場合があります。

1. 照会に対して適切な索引がすべて作成されていますか? 例えば、列に固有値が含まれていることが分かっている場合、ユニーク索引を作成すると、必要なソート・ヒープ・メモリーが削減される場合があります。
2. 1 つ以上の SQL ステートメントを変更してソート・メモリーの使用量を削減できるかどうかを判断します。すべての **ORDER BY** 節が照会に必要ですか? 副照会で **DISTINCT** の代わりに **GROUP BY** を使用できますか?
3. 列グループの統計または統計ビューを利用します。このようにすると、DB2 照会オプティマイザーでソート・ヒープ・メモリーの使用量が少ないアクセス・プランを判別できることがあります。

sheapthres のサイズを増やす

SQL に対してこれまでの変更のどれも可能ではない場合、またはパフォーマンスが向上しない場合、**sheapthres** パラメーターの値を増やすことを次に検討します。このパラメーターは、任意の時点で実行するすべての専用ソートで使用されるメモリーの合計量を定義します。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に割り当てられたメモリーが相当量削減されます。**sheapthres** のしきい値を超えると、削減されたメモリー量で実行する必要があるソート要求を追跡するために、**post_threshold_sorts** モニター・エレメントが使用されます。

すべてのソートに対して割り振られるメモリー量を増やす方法、および SQL ソートのパフォーマンスとメモリー使用量のバランスを取る方法について詳しくは、**sheapthres** の文書を参照してください。任意の単独のソートのために確保しておくメモリー量 (**sortheap**) と、パフォーマンスが低下し始めるときにシステムに存在すると想定される標準的なソート数を検討する必要があります。**sheapthres** の値を増やしたときの効果を確認するには、変更後の **post_threshold_sorts** の数の減少を確認する必要があります。

sortheap のサイズを減らす

最後の方法として、ソート・ヒープ・メモリー全体 (**sheapthres**) を増やすことができず、小さなソートが多数存在する場合には、代わりに個別のソート・ヒープ・サイズの **sortheap** パラメーターを減らすと効果的であることがあります。この結果、個別のソートごとに割り振られるメモリーは減少しますが、1 回で実行される可能性のあるソートの数が増加します。

しかし、このオプションは、多数の小さいソートがあり、減らした後の小さいソート・ヒープでもそれぞれのソートを実行できることを確認できる場合にのみ検討してください。**sortheap** の設定が小さすぎると、ソートでディスクへのスピルが発生し始め、ディスク入出力はメモリー内の操作より遅い

ため、パフォーマンスが低下します。 `db2pd` コマンドを実行したときに `-sort` パラメーターの一部として返される `NumSpills` の値を使用して、照会における個別のソート・ヒープ・サイズである `sortheap` の設定が小さすぎないか判断できます。

パフォーマンスの概要

パフォーマンスとは、コンピューター・システムが特定のワークロードに応答する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、およびリソースの使用率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有の程度

一般に、システムを調整することにより、その費用対効果の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
- 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・チューニングにより、リソースをより効率的に使用できたり、システムに対してより多くのユーザーを追加できたりするなどのはっきりとした効果が現れる場合があります。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には、はっきりとは分からないものもあります。

パフォーマンス・チューニングのガイドライン

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際に、以下の指針に留意してください。

- **収穫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。
- **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うと、実際にはそれ以降のチューニング作業が行いにくくなる可能性があります。
- **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リソース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、その変更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。パフォーマンス・チューニングでは、様々なシステム・リソース間のトレードオフが必要になります。例えば、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。

- **一度に 1 つのパラメーターを変更する:** 一度に複数の要素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。
- **レベルごとに測定と構成を行う:** 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。システム・レベルには、以下のものがあります。
 - 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 および XQuery 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。サービスをハードウェアまたはソフトウェアに適用する前に、システムのモニターとチューニングに余分な時間をかけないでください。
- **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ンネルがないことに気付く場合があります。
- **チューニングを開始する前にフォールバック手順を適所に配置する:** チューニングをしたものの予期しない性能低下が生じる場合、別の方法を試みる前に実行した変更内容を元に戻す必要があります。元の設定を保存して、維持しない変更内容を簡単に取り消せるようにします。

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、反復的アプローチです。このパフォーマンス・モニターの結果に基づいて、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、データを使用するアプリケーションの種類に関する知識、およびデータ・アクセスのパターンについての理解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。

パフォーマンスの改善プロセスすべてには、以下の基本的なステップが含まれています。

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 つの調整を行います。

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、ハードウェアをアップグレードする時期である可能性があります。

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でない場合は、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。以下のような幾つかの簡単な質問に答えることから開始します。

- 「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？
- 特定のタイプ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？
- 何らかの発生パターンがありますか？ 例えば、その問題は一日の特定の時刻に、または連続的に発生しますか？

パフォーマンス・チューニングの限界

パフォーマンス・チューニングの効果には限界があります。システム・パフォーマンスの改善にける必要のある時間と費用について考慮する際、余分の時間と費用をどの程度かけるとシステムのユーザーにとって役立つかを評価してください。

システムで応答時間やスループットに関する問題が生じる場合には、チューニングによってパフォーマンスが改善される場合が少なくありません。しかし、さらに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。その場合、目標と期待値を見直す必要があります。パフォーマンスをさらに大幅に改善するには、ディスク装置の増加、CPUs 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。

ソート・パフォーマンスのチューニング

照会において、ソート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・ヒープを正しく構成することは、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場合 (例えば、ORDER BY 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る場合

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因子があります。

- 次の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**)、および共有ソートのソート・ヒープしきい値 (**sheapthres_shr**)。これらは、そのインスタンス全体をソートするために使用できる合計メモリー量を制御します。
- 大量のソートを必要とするワークロードにおけるステートメントの数
- 不要なソートを避けるのに役立つ索引の存在と不在
- ソートの必要を最小限に抑える機能がないアプリケーション論理の使用
- 並列ソート。これはソートのパフォーマンスを向上させますが、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しか行えません。
- ソートがオーバーフロー するかどうか。ソートされたデータがソート・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) に収まらない場合は、データベースが所有する一時表にデータがオーバーフローします。
- ソートの結果がパイプ接続 されるかどうか。ソートされたリストを保管する一時表を使わなくてもソートされたデータを直接返せる場合は、パイプ・ソートになります。

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・ヒープは解放されません。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。

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。

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。

1. 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。
2. 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。

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もできます。

3. これらのステートメントを設計アドバイザーへの入力として使用します。これにより、ソートの必要を減らす索引が識別され、作成できます。

セルフチューニング・メモリー・マネージャー (STMM) を使用すると、ソートに必要なメモリー・リソースを、自動的かつ動的に、割り振りおよび割り振り解除できます。この機能を使用するには、以下のことを行ってください。

- **self_tuning_mem** 構成パラメーターを ON に設定して、データベースのセルフチューニング・メモリーを使用可能にします。
- **sortheap** および **sheapthres_shr** 構成パラメーターを AUTOMATIC に設定します。
- **sheapthres** 構成パラメーターを 0 に設定します。

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して、**sortheap**、**sheapthres_shr**、および **sheapthres** 構成パラメーターの設定に役立てることもできます。データベース・マネージャーごとに、またデータベースごとに、次のことを実行してください。

1. 代表的なワークロードを設定し稼働します。
2. 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・ヒープ合計 (**sort_heap_allocated** モニター・エレメントの値)
 - アクティブ・ソートおよびアクティブ・ハッシュ結合 (**active_sorts** および **active_hash_joins** モニター・エレメントの値)
3. データベースごとに、**sortheap** を 使用中のソート・ヒープ合計 の平均値に設定します。

注: ソートで長いキーが使用される場合、**sortheap** 構成パラメーターの値を増やす必要があるかもしれません。

4. **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。
 - a. 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。
 - b. 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。

- c. **sheapthres** には、アクティブ・ソートの平均数に、上記で算出したソート・ヒープの平均サイズを掛けた値を設定します。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。

IBM InfoSphere Optim Query Workload Tunerは、単一の SQL ステートメントのパフォーマンスと SQL ステートメント・グループのパフォーマンス (照会ワークロードと呼ばれる) を向上するツールを提供しています。この製品について詳しくは、<http://www.ibm.com/software/data/optim/query-workload-tuner-db2-luw/index.html>の製品概要のページを参照してください。また、バージョン 3.1.1 以降では、ワークロード設計アドバイザーを使用して、DB2 設計アドバイザー・ウィザードで利用できた多くの操作を実行できます。詳しくは、<http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.datatools.qrytune.workloadtunedb2luw.doc/topics/genrecdsngn.html>のワークロード設計アドバイザーの資料を参照してください。

RUNSTATS の影響の最小化

RUNSTATS のパフォーマンスを改善するのに使用可能な方法がいくつかあります。

このユーティリティーのパフォーマンスへの影響を最小化するには、次のようにします。

- **COLUMNS** 節を使用して統計が収集される列を制限します。照会ワークロードにおいて、列の多くは述部によって参照されることがないため、統計を必要としません。
- データの分散が均一である傾向があるなら、分散統計が収集される列を制限します。分散統計の収集は、基本的な列統計の収集よりも多くの CPU およびメモリーを必要とします。しかし、列の値が均一に分散しているかどうかを判断するには、既存の統計があること、またはデータを照会することが求められます。また、この方法では、表が変更されてもデータが均一に分散したままであることが前提となります。
- ページ・レベルまたは行レベルの表サンプリングを使用し (**TABLESAMPLE SYSTEM** または **TABLESAMPLE BERNOULLI** 節を指定)、またページ・レベルまたは行レベルの索引サンプリングを使用して (**INDEXSAMPLE SYSTEM** または **INDEXSAMPLE BERNOULLI** 節を指定) 処理される、ページおよび行の数を制限します。
TABLESAMPLE SYSTEM(10) および **INDEXSAMPLE SYSTEM(10)** を指定して、10% のページ・レベル・サンプルから始めます。統計の正確性を確認し、アクセス・プランの変更によりシステム・パフォーマンスが低下していないかどうかを確認します。低下しているなら、**TABLESAMPLE BERNOULLI(10)** を使用して、10% の行レベル・サンプルを代わりに試します。同様に、索引サンプリングに適正な比率を取得するために、**INDEXSAMPLE** パラメーターを試します。統計の正確性が不十分であるなら、サンプルの量を増やします。**RUNSTATS** ページ・レベルまたは行レベルのサンプルを使用する場合、結合する表についても同じサンプリング・レートを使用してください。これは、結合列の統計で同レベルの正確性を確保する上で重要です。
- **CREATE INDEX** ステートメントで **COLLECT STATISTICS** オプションを指定することにより、索引の作成中に索引統計を収集します。これは、索引の作成後に **RUNSTATS** 操作を別に行うよりも速い方法です。また、作成直後に生成された統計を新規索引で確保し、索引使用のコストをオプティマイザーによって正確に見積もることが可能となります。
- **LOAD** コマンドの実行時に、**REPLACE** オプションを使用して統計を収集します。これは、ロード操作後に **RUNSTATS** 操作を別に行うよりも速い方法です。また、データのロード直後に最新の統計を表で確保し、表を使用するコストをオプティマイザーによって正確に見積もることが可能となります。

パーティション・データベース環境において、**RUNSTATS** ユーティリティーは、単一のデータベース・パーティションから統計を収集します。表が存在するデータベース・パーティションで **RUNSTATS** コマンドを発行すると、そこで統計が収集されます。表が存在しない場合、表のデータベース・パーティション・グループにおける 1 番目のデータベース・パーティションで、統計が収集されます。統計の整合性を取るには、同じデータベース・パーティションから結合表の統計を収集するようにしてください。

データ圧縮とパフォーマンス

データ圧縮を使用すると、ディスクから読み取らなければならないデータ量やディスクに書き込まなければならないデータ量を減らして、入出力コストを削減することができます。

一般的に、データ行中に存在する反復パターンが多ければ多いほど、圧縮率は高くなります。データに反復ストリングが含まれていない場合 (数値データが主である場合や、インライン BLOB データが入っている場合など) は、圧縮してもストレージの節約の点でそれほど効果が得られない可能性があります。さらに、データがあまり圧縮されないにも関わらず、データを圧縮しようとするによりデータベース・マネージャーのパフォーマンスに影響が及び、ストレージに関する利点がコンプレッション・ディクショナリーを保持する負担に見合わない場合があります。圧縮によって得られるストレージ推定節約量を確認するには、`ADMIN_GET_TAB_COMPRESS_INFO` 表関数を使用します。

以下の 2 つの形式のデータ圧縮を使用できます。

行圧縮 以下の 2 つのタイプの行圧縮を使用できます。

クラシック行圧縮

クラシック行圧縮では、1 つの行の中で複数の列値にまたがる反復パターンをより短いシンボル・ストリングに置き換えます。反復データのインスタンスを見つけるために、サンプリングした行がスキャンされます。このスキャンから、表レベルのコンプレッション・ディクショナリーが作成されます。このディクショナリーを使用して、反復データをより短いシンボル・ストリングで置き換えます。

表レベルのコンプレッション・ディクショナリーは静的です。つまり、これらのディクショナリーは、最初に作成してしまうとその後は、従来の表再編成中に再作成しない限り変更されません。

アダプティブ行圧縮

アダプティブ行圧縮では、クラシック行圧縮とページ・レベルの圧縮の 2 つの圧縮手法が使用されます。ページ・レベルの圧縮では、データの単一ページ内の複数の列値にまたがる反復パターンが、より短いシンボル・ストリングで置き換えられます。ページがデータで埋まると、反復データがないかページ圧縮ロジックがそのページをスキャンします。このスキャンから、ページ・レベルのコンプレッション・ディクショナリーが作成されます。このディクショナリーを使用して、反復データをより短いシンボル・ストリングで置き換えます。

ページ・レベルのディクショナリーは動的であり、これらは、必要に応じて自動的に再作成されます。

値圧縮 値圧縮では、値の重複項目を除去し、1 つのコピーだけを保管し、格納された値への参照の場所に関するトラッキングを保持します。

行圧縮は一時表にも使用されます。一時表を圧縮すると、大規模で複雑な照会に必要な一時ディスク・スペースの量が減り、照会のパフォーマンスが向上します。DB2 Storage Optimization Feature では、一時表の圧縮が自動的に使用可能になります。行圧縮に適格な各一時表には、それぞれの表のコンプレッション・ディクショ

ナリーの作成用にさらにメモリーが 2 から 3 MB 必要になります。このメモリーは、コンプレッション・ディクショナリーの作成が完了するまで割り振られたままになります。

索引オブジェクトを圧縮することもできます。これにより、ストレージ・コストが削減されます。この種の圧縮が特に役立つのは、非常に大きな索引を数多く使用することが一般的である大規模なオンライン・トランザクション処理 (OLTP) 環境やデータウェアハウス環境の場合です。どちらの場合も、索引圧縮を行うことにより、入出力制約のある環境では大幅にパフォーマンスが改善され、CPU 制約のある環境ではパフォーマンスの低下はほとんど、あるいは全く見られません。

XML 列のある表で圧縮が使用可能である場合、XDA オブジェクトに保管されている XML データも圧縮されます。XML データの別個のコンプレッション・ディクショナリーは、XDA オブジェクトに保管されています。DB2 バージョン 9.7 以降を使用して、DB2 バージョン 9.5 を使用して作成された表に XML 列を追加すると、これらの XML 列は圧縮されます。ただし、DB2 バージョン 9.5 を使用して表に追加された XML 列は圧縮されません。そのような表ではデータ・オブジェクトだけが圧縮されます。

最適化のトラブルシューティング

最適化ガイドラインおよびプロファイルのトラブルシューティング

最適化ガイドライン (最適化プロファイルにより渡される) の診断サポートは、EXPLAIN 表で行われます。

オブティマイザーが最適化ガイドラインを適用しない場合には、SQL0437W 警告、理由コード 13 を受け取ります。最適化ガイドラインが適用されなかった理由を説明する診断情報が EXPLAIN 表に追加されます。オブティマイザーの診断出力を受け取るための EXPLAIN 表には次の 2 つがあります。

- EXPLAIN_DIAGNOSTIC - この表の各項目は、特定のステートメントの最適化に関する診断メッセージを表します。各診断メッセージは、数字コードを使用して示されます。
- EXPLAIN_DIAGNOSTIC_DATA - この表の各項目は、EXPLAIN_DIAGNOSTIC 表内の特定の診断メッセージと関連した診断データです。

診断用 Explain 表を作成するための DDL を 663 ページの図 39 に示します。

以下のステップは、最適化ガイドラインの使用時に発生する問題をトラブルシューティングするときに役立ちます。

1. 「問題判別およびデータベース・パフォーマンスのチューニング」の『最適化ガイドラインが使用されていることの確認』を参照します。
2. 組み込みの「管理ルーチンおよびビュー」の『EXPLAIN_GET_MSGS 表関数』を使用して完全なエラー・メッセージを調べます。

これらの手順を完了しても問題の原因を突き止めることができない場合には、診断データの収集を開始して、IBM ソフトウェア・サポートに連絡を取ることを考慮してください。

```

CREATE TABLE EXPLAIN DIAGNOSTIC
( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
  EXPLAIN_TIME       TIMESTAMP   NOT NULL,
  SOURCE_NAME        VARCHAR(128) NOT NULL,
  SOURCE_SCHEMA      VARCHAR(128) NOT NULL,
  SOURCE_VERSION     VARCHAR(64)  NOT NULL,
  EXPLAIN_LEVEL      CHAR(1)     NOT NULL,
  STMTNO             INTEGER      NOT NULL,
  SECTNO             INTEGER      NOT NULL,
  DIAGNOSTIC_ID     INTEGER      NOT NULL,
  CODE               INTEGER      NOT NULL,
  PRIMARY KEY (EXPLAIN_REQUESTER,
               EXPLAIN_TIME,
               SOURCE_NAME,
               SOURCE_SCHEMA,
               SOURCE_VERSION,
               EXPLAIN_LEVEL,
               STMTNO,
               SECTNO,
               DIAGNOSTIC_ID),
  FOREIGN KEY (EXPLAIN_REQUESTER,
               EXPLAIN_TIME,
               SOURCE_NAME,
               SOURCE_SCHEMA,
               SOURCE_VERSION,
               EXPLAIN_LEVEL,
               STMTNO,
               SECTNO)
  REFERENCES EXPLAIN_STATEMENT ON DELETE CASCADE);

```

```

CREATE TABLE EXPLAIN_DIAGNOSTIC_DATA
( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
  EXPLAIN_TIME       TIMESTAMP   NOT NULL,
  SOURCE_NAME        VARCHAR(128) NOT NULL,
  SOURCE_SCHEMA      VARCHAR(128) NOT NULL,
  SOURCE_VERSION     VARCHAR(64)  NOT NULL,
  EXPLAIN_LEVEL      CHAR(1)     NOT NULL,
  STMTNO             INTEGER      NOT NULL,
  SECTNO             INTEGER      NOT NULL,
  DIAGNOSTIC_ID     INTEGER      NOT NULL,
  ORDINAL            INTEGER      NOT NULL,
  TOKEN              VARCHAR(1000),
  TOKEN_LONG         BLOB(3M) NOT LOGGED,
  FOREIGN KEY (EXPLAIN_REQUESTER,
               EXPLAIN_TIME,
               SOURCE_NAME,
               SOURCE_SCHEMA,
               SOURCE_VERSION,
               EXPLAIN_LEVEL,
               STMTNO,
               SECTNO,
               DIAGNOSTIC_ID)
  REFERENCES EXPLAIN_DIAGNOSTIC ON DELETE CASCADE);

```

注: EXPLAIN_REQUESTER、EXPLAIN_TIME、SOURCE_NAME、SOURCE_SCHEMA、SOURCE_VERSION、EXPLAIN_LEVEL、STMTNO、SECTNO の各列は、EXPLAIN_STATEMENT 表の外部キーと、EXPLAIN_DIAGNOSTIC と EXPLAIN_DIAGNOSTIC_DATA の親子関係を設定するために、両方の表に含まれていません。

図 39. 診断用 Explain 表を作成するための DDL

この DDL は、sqllib ディレクトリーの misc サブディレクトリーに置かれている EXPLAIN.DDL ファイルに組み込まれています。

パーティション・データベース環境のトラブルシューティング

127.0.0.2 に関連した FCM 問題 (Linux および UNIX)

パーティション・データベース環境では、`/etc/hosts` ファイルに 127.0.0.2 の項目があると、高速コミュニケーション・マネージャー (FCM) が問題を検出する可能性があります。

症状

環境によっては、さまざまなエラー・メッセージが発生します。例えば、データベースを作成するときに、「SQL1229N システム・エラーのため、現在のトランザクションがロールバックされました。SQLSTATE=40504」というエラーが発生する可能性があります。

原因

この問題の原因は、`/etc/hosts` ファイルに IP アドレス 127.0.0.2 の項目が存在することです (127.0.0.2 は、マシンの完全修飾ホスト名にマップされています)。例えば、

```
127.0.0.2 ServerA.ibm.com ServerA
```

「ServerA.ibm.com」は、完全修飾ホスト名です。

環境

この問題は、パーティション・データベース環境だけに限定されます。

問題の解決

`/etc/hosts` ファイルから項目を除去するか、その項目をコメントに変換します。例えば、

```
# 127.0.0.2 ServerA.ibm.com ServerA
```

暗号化ファイル・システムにおけるデータベース・パーティションの作成 (AIX)

AIX オペレーティング・システムでは、JFS2 ファイル・システムまたはファイル・セットの暗号化機能がサポートされています。この機能では、DB2 データベース製品のパーティション化されたデータベース環境ではサポートされていません。AIX 上で、パーティション化されたデータベース環境を EFS (暗号化ファイル・システム) を使用して作成しようとする、SQL10004C エラーが生じます。

症状

複数パーティション・データベース環境にある暗号化ファイル・システムでデータベースを作成しようとする、次のエラーを受け取ります。SQL10004C データベース・ディレクトリーのアクセス中に入出力エラーが発生しました。SQLSTATE=58031

原因

この時点で、EFS (暗号化ファイル・システム) を使用して AIX 上にパーティション化されたデータベース環境を作成することはできません。パーティション化されたデータベース・パーティションは **rsh** または **ssh** を使用するの、EFS 内の鍵ストアが失われると、暗号化ファイル・システムに格納されているデータベース・ファイルにデータベース・パーティションはアクセスできなくなります。

問題の診断

DB2 診断 (**db2diag**) ログ・ファイルには以下のエラー・メッセージが組み込まれます。OSERR : ENOATTR (112) "属性がありません (No attribute found)".

問題の解決

パーティション化されたデータベース環境でデータベースを正常に作成するには、すべてのマシンで使用できるファイル・システムがあること、およびそれが暗号化ファイル・システムではないことが必要です。

データ再配分時の表の状態のトラブルシューティング

再配分中操作を開始する前に、データベース・パーティション・グループ内のすべての表がフルアクセス・モードで、通常の状態であることを確認してください。

症状

表の状態が原因で再配分に失敗する場合は、データベース・パーティション・グループを再配分できないこと、あるいはその操作が許可されていないことがエラー・メッセージで示されます。例えば、SQL02436N、SQL6056N、および SQL0668N メッセージがこの問題の徴候である可能性があります。

注: エラー・メッセージに表の名前がリストされる場合は、データベース・パーティション・グループ内の問題のある表が 1 つだけでない可能性があります。データベース・パーティション・グループ内のすべての表の状態をトラブルシューティングすると、再配分の試行が何度も失敗しないようにすることができます。

問題の診断

ユーザーの処置:

1. 作動不能な状態 (SYSCAT.TABLES.STATUS='X') の表を判別します。

次の照会を発行します。

```
SELECT TABNAME
FROM SYSCAT.TABLES AS TABLES, SYSCAT.TABLESPACES AS TABLESPACES
WHERE TABLES.TBSPACE = TABLESPACES.TBSPACE AND TABLES.STATUS = 'X'
AND TABLESPACES.DBPGNAME = 'IBMDEFAULTGROUP'
```

IBMDEFAULTGROUP はデータベース・パーティション・グループ名です。

2. 「SET INTEGRITY ペンディング」状態 (SYSCAT.TABLES.STATUS='C') の表を判別します。

次の照会を発行します。

```

SELECT TABNAME
FROM SYSCAT.TABLES AS TABLES, SYSCAT.TABLESPACES AS TABLESPACES
WHERE TABLES.TBSPACE = TABLESPACES.TBSPACE AND TABLES.STATUS = 'C'
AND TABLESPACES.DBPGNAME = 'IBMDEFAULTGROUP'

```

IBMDEFAULTGROUP はデータベース・パーティション・グループ名です。

3. 通常の状態、フルアクセス・モードでない表を判別します。

次の照会を発行します。

```

SELECT DISTINCT TRIM(T.OWNER) || '¥'.¥' || TRIM(T.TABNAME)
AS NAME, T.ACCESS_MODE, A.LOAD_STATUS
FROM SYSCAT.TABLES T, SYSCAT.DBPARTITIONGROUPS N,
SYSIBMADM.ADMINTABINFO A
WHERE T.PMAP_ID = N.PMAP_ID
AND A.TABSCHHEMA = T.OWNER
AND A.TABNAME = T.TABNAME
AND N.DBPGNAME = 'IBMDEFAULTGROUP'
AND (T.ACCESS_MODE <> 'F' OR A.LOAD_STATUS IS NOT NULL)
AND T.STATUS='N'

```

IBMDEFAULTGROUP はデータベース・パーティション・グループ名です。この照会の実行に長時間要する場合は、照会を終了し、この照会に含まれているすべての表に対して **RUNSTATS** コマンドを発行してから、照会を再発行します。

問題の解決

ユーザーの処置:

1. 以下の必要な修正処置を取ります。
 - a. 作動不能な状態の表ごとに、**LOAD QUERY** コマンドを使用して、特定の表の状態を判別します。
 - 「ロード進行中」状態の表の場合、ロード操作が終了するまで待機します。**LOAD QUERY** コマンドを使って、ロード操作の進行状態をモニターできます。
 - 「ロード・ペンディング」状態の表の場合、**RESTART** または **TERMINATE** コマンド・パラメーターを指定して **LOAD** コマンドを発行し、直前に失敗したロード操作を再始動するか終了します。
 - 「読み取りアクセスのみ」状態の表の場合、**LOAD QUERY** コマンドを使用して、表がロード中かどうかをチェックします。ロード中の場合は、ロード・ユーティリティーが完了するまで待機するか、または必要に応じて、直前に失敗したロード操作を再始動または終了してください。ロード操作が現在進行中でない場合は、**IMMEDIATE CHECKED** オプションを指定した **SET INTEGRITY** ステートメントを発行します。このアクションにより、表の新しくロードされた部分で制約の妥当性検査が行われます。
 - 「REORG ペンディング」状態の表の場合、従来の再編操作を実行して、表をアクセス可能にします。
 - 「ロード再始動不可」状態の表の場合、**LOAD TERMINATE** コマンドか **LOAD REPLACE** コマンドを発行します。
 - 「使用不可」状態の表の場合、表をドロップするか、バックアップからリストアします。

- b. 「SET INTEGRITY ペンディング」状態の表の場合、IMMEDIATE CHECKED オプションを指定して SET INTEGRITY ステートメントを実行します。
- c. フルアクセス・モードでない表の場合、従属即時マテリアライズ照会表とステージング表に対して、IMMEDIATE CHECKED オプションを指定して SET INTEGRITY ステートメントを実行します。

注: 問題の解決を据え置くことを選択し、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドで **EXCLUDE** パラメーターを指定して、再配分操作から表を一時的に省くこともできます。この場合、再配分操作は正常に完了しますが、データベース・パーティション・グループは部分的にのみ再配分されます。その結果、データベース・パーティション・グループ内の再配分済みの表が、まだ再配分されていない表とは別のパーティション・マップを使用する可能性があります。さらに、再配分操作の前に、再配分済みの表と再配分されていない表の間の連結が存在した場合、それらの表の間の連結プロパティは一時的に無効にされます。完全に再配分されるまで、照会のパフォーマンスが最適にならない可能性があります。

- 2. 前の再配分操作が失敗した場合は、**CONTINUE** オプションか **ABORT** オプションを使用して再サブミットします。**CONTINUE** は前に異常終了した再配分操作を完了し、**ABORT** は前に異常終了した操作の影響を取り消します。

スクリプトのトラブルシューティング

データベース・エンジンで実行されているプロセスに基づく内部ツールまたはスクリプトが存在することもあります。すべてのエージェント、プリフェッチャー、ページ・クリーナーが、1 つのマルチスレッド・プロセスに含まれている各スレッドと見なされるようになると、それらのツールやスクリプトが機能しなくなる可能性があります。

スレッド化プロセスに合わせて、内部ツールやスクリプトを変更する必要があります。例えば、**ps** コマンドを開始してプロセス名をリストしてから、特定のエージェント・プロセスに対してタスクを実行するスクリプトがあるとします。その場合は、スクリプトを作成し直す必要があります。

問題判別データベース・コマンド **db2pd** には、すべてのエージェント名とそれぞれのスレッド ID をリストするための新規オプション **-edu** (「engine dispatchable unit」、つまりエンジン・ディスパッチ可能単位の略) があります。**db2pd -stack** コマンドは引き続き、スレッド化エンジンを操作して、個々の EDU スタック、または現在のノードのすべての EDU スタックをダンプできます。

フィックス・パック 1 の適用後にセクション実行時統計を収集するための静的セクションの再コンパイル

DB2 バージョン 9.7 フィックスパック 1 を適用した後、フィックスパックを適用する前にコンパイルした静的セクションのセクション実行時統計を収集することはできません。フィックスパック 1 の適用後にセクション実行時統計を収集するには、静的セクションを再コンパイルする必要があります。

症状

EXPLAIN_FROM_ACTIVITY ルーチンを実行する際に、セクション実行時統計が収集されません。

原因

フィックスパックを適用する前にコンパイルした静的セクションのセクション実行時統計を収集することはできません。

問題の解決

DB2 V9.7 フィックスパック 1 をインストールした後、フィックスパックの適用以降、**REBIND** コマンドを使用して静的セクションが再バインドされていることを確認してください。これを行うには、SYSCAT.PACKAGES カタログ・ビューの **LAST_BIND_TIME** 列を確認してください。

ストレージ・キー・サポートのトラブルシューティング

ストレージ保護キー (スレッド・レベルのハードウェア・キー) は、無効なアクセスの試行からメモリーを保護することにより、DB2 エンジンの回復力を高めるために使用されます。このフィーチャーを使用可能にする際に発生したエラーを解決するには、次のセクションの指示を実行してください。

このタスクについて

レジストリー変数 「データベース: 管理の概念および構成リファレンス」の『**DB2_MEMORY_PROTECT**』を設定したときに、「値が無効です」(DBI1301E) エラーが返されました。このエラーは、以下のいずれかの理由によって生じます。

- レジストリー変数に指定された値が無効である。有効な値については、**DB2_MEMORY_PROTECT** のレジストリー変数の使用法を参照してください。
- ハードウェアおよびオペレーティング・システムがストレージ保護キーをサポートしておらず、フィーチャーを使用可能にできない可能性があります。

第 7 章 DB2 pureScale Feature のトラブルシューティング

このトラブルシューティング・セクションには、IBM DB2 pureScale Feature で発生する問題を理解、特定、および解決するうえで役立つ情報が含まれています。ここに掲載される情報は、DB2 インフォメーション・センターにある DB2 のトラブルシューティングと DB2 pureScale のモニターに関する情報を補足するものです。

注: DB2 pureScale ソフトウェア・スタックの一部としてパッケージ化される IBM 製品のサポートが必要な場合は、DB2 pureScale Feature のサービス要求または Problem Management Record (PMR) を開きます。DB2 pureScale Feature の PMR を開くと、問題をより効率的に解決できます。

DB2 pureScale Feature での問題の診断方法

問題が見つかった場合、やみくもに解決しようとするのではなく、問題の内容を明確にして正しく切り分けることが重要です。このトピックの診断手順を使用することにより、DB2 pureScale Feature が備えるトラブルシューティング・ツールを利用した問題判別プロセスに従います。多くの場合、問題を自分で解決できる可能性があります。

トラブルシューティングの資料に記載されている主な分野には、次のものがあります。

- ログとトレース両方による診断情報の取得
- DB2 pureScale Feature のインストール、概要、およびアンインストール
- ホストやメンバーの障害などのコンポーネントの障害。状態、アラート、および再始動条件の特定と解決の方法を含む
- 通信障害 (uDAPL)
- クラスタ・ファイル・システムの問題と障害 (GPFS™)
- クラスタ・マネージャー・ソフトウェアの障害 (Tivoli SA MP)
- IBM サービスを呼ぶ必要がある問題のシナリオ

それは現在発生している問題かそれとも繰り返し発生する問題か?

DB2 pureScale Feature をインストールして稼働状態にすると、問題はだまかに 2 つのカテゴリのいずれかに分類できます。つまり、直ちにサーバーに影響を与える問題と、過去のある時点で発生したもので、DB2 pureScale Feature によって自動的に回復された可能性がある問題の 2 つです。

直ちにサーバーに影響を与える問題

通常、これらの問題は、何らかの方法でサーバーに影響を与える 1 つ以上のはっきりした症状があります。例えば、パフォーマンスのスローダウンがあるとユーザーから報告を受けることや、システムの能力が失われていることを自分で確認できることがあります。全体的に、システムは継続して利用できます。そのような症状は、メンバーまたはクラスタ・キャッシング・ファシリティ (CF) に問題があることを示しているため、直接調査できます。

システムが完全に停止した場合、原因として考えられる点はいくつかありますが、最初に確認すべき点は、少なくとも 1 つの CF が稼働している、重要なインフラストラクチャー・サービスをインスタンスに提供しているかどうかという点です。CF が使用できなければ、メンバーのロックとキャッシングを処理するコンポーネントがないため、インスタンスは実行できません。

過去のある時点で発生した問題

通常、これらの問題は、サーバーへの現在の影響はないように見えますが、過去に問題があったことを示している可能性があります。例えば、診断の必要がある何らかの問題がシステムで発生していることを示す、説明のないログ項目が示される場合があります。DB2 pureScale Feature の高可用性とフォールト・トレラントな特性によって、いくつかの問題が隠れてしまうことがあります。なぜなら、コンポーネントに繰り返し障害が発生する場合であっても、インスタンスはユーザー介入を必要とせずほとんどのコンポーネント障害から継続して復旧するためです。実際には、偶発的または繰り返し発生する問題をしばらくモニターして、何度も自己修復するが後で再び発生する隠された問題がサーバーにあるかどうかを判断し、根本原因を解決する必要があります。

確認する必要がある診断情報

問題定義と使用可能な診断データに基づき、問題が発生するタイミングと理由を理解する必要があります。診断データを確認するときに、問題定義に示された症状と考えられる原因を関連付けるイベントやアラートを探します。

トラブルシューティングのために使用する DB2 pureScale 固有のコマンドを、任意のメンバー・ホストまたはクラスター・キャッシング・ファシリティ・ホストから実行して、インスタンス全体に関する情報を取得できます。データベースが利用できない、またはインスタンスが停止している場合でも、コマンドを発行するホストで DB2 pureScale クラスター・マネージャー・コンポーネントを使用できるならば、たいいていは何らかのデータを報告することができます。

DB2 pureScale Feature の診断手順について推奨される順序は、次のとおりです。

1. **db2instance -list** コマンドを発行して、インスタンスに影響を与えている現在の問題を特定します。ホスト・コンピューターに影響を与えるアラートを最初に確認し、次に CF とメンバーでアラートを確認します。出力の中で、メンバー、クラスター・キャッシング・ファシリティ、およびホストの現在の状態を見つけます。アラート、エラー、またはリカバリー関連の状態が存在するかどうかがこの出力に示されます。
2. **db2cluster -cm -list -alert** コマンドを発行して、これらのアラートに対して推奨されるアクションを確認します。**db2cluster** コマンドの出力には、どのようなアラートが存在し、アラート条件の影響がどのようなもので、解決するために何をすべきかが示されます。推奨アクションを使って問題を解決できるかどうか考慮します。また、このコマンドを使用してアラートをクリアすることもできます。
3. 利用可能な診断ログの中に、問題の考えられる原因を示している可能性があるヒントがあるかどうかを確認します。問題がいつ始まったかについて大まかに知っておくと、対応するタイム・スタンプを使用してログ項目を確認することで、検索を絞り込むことができます。

- `DIAGPATH` ファイル・パスの中に生成された診断ファイルがあるかどうかを確認し、`FODC` ディレクトリーを確認します。
- `CF_DIAGPATH` ファイル・パスの中に、クラスター・キャッシング・ファシリテーターの診断データが生成されているかどうか確認します
- `db2diag.log` 診断ログ・ファイルの中に最近の診断情報とエラー・メッセージがあるかどうかを確認します。
- 通知ログの中に問題の開始点を示す可能性がある最近のメッセージがあるかどうかを確認します。
- `errpt -a` コマンドを実行してシステム・エラー・ログを表示します。障害の発生時刻に関連するログ項目を探します。

次のステップ

利用可能な診断情報をステップごとに確認すると、次に確認すべきトラブルシューティング・シナリオが分かります。問題の範囲を絞り込んだ後、`DB2 pureScale` のトラブルシューティング資料を調べて、問題定義に最も当てはまる文脈を探します。トラブルシューティングの内容には、たいてい 2 つタイプがあります。つまり、トラブルシューティングのよくある質問 (FAQ) に対する回答のように、非常に具体的な文脈を扱うものと、診断データの解釈方法と問題の解決方法を示す、より包括的なトラブルシューティング・シナリオです。

例えば、確認した診断情報によると、あるメンバーにアラートがあり、リカバリーの目的でゲスト・ホストにフェイルオーバーした後、ホームのホストへのフェイルバックを待機しています (`restart light` という操作)。この場合、トラブルシューティング資料で `DB2 pureScale` インスタンスの操作シナリオを確認してから、アラートがあるメンバーまたはホストに関するサブセクションを確認することによって、関連する障害のシナリオを見つけることができます。起こりうる障害のシナリオすべてが取り上げられているあるわけではありませんが、多くのシナリオが扱われています。

手動によるトレースおよびログ・ファイルの収集

トレースおよびログ・ファイルには、問題のトラブルシューティングに役立つ情報が含まれています。 `db2support` コマンドを使って `IBM DB2 pureScale Feature` の診断データを収集できない場合には、トレースおよびログ情報を手動で集めることができます。

重要: `-install` パラメーターを指定して `db2support` コマンドを使用できる場合には、このトピックの手操作の説明に従う必要はありません。手操作の説明に従うと、同様の診断情報を集めることができますが、それを完了するにはより多くの手順を行う必要があります。

`IBM` サポートに以下のファイルを提出する必要があります (ファイルが入手可能な場合)。これらのファイルをすべて含む単一の `.zip` ファイルまたは `.tar` アーカイブを作成してください。

- `/tmp` ディレクトリー内の以下のファイル:
 - インストール開始ホスト (IIH) の `/tmp` ディレクトリーにある現在のインストール済み環境またはインスタンスのログおよびトレース・ファイル。エラーを

戻した DB2 インストール/構成コマンドと関係のあるトレースおよびログ・ファイルを確実に収集するには、トレース・オプションをオンにして (通常は **-d** または **-t** パラメーターを含めて)、障害の発生したコマンドをすべて再発行してください。例えば **db2icrt** コマンドを再発行する場合、コマンド・パラメーター **-d** を含めることによりトレースをオンにします。トレース出力ファイルは `/tmp/db2icrt.trc.pid` という名前になります (*pid* はプロセス ID)。

- 以下に示す、`/tmp` ディレクトリー内の他のトレースおよびログ・ファイル:
 - `/tmp/db2trc.trc.*` ファイル。
 - IIIH の `/tmp/db2log.pid` ディレクトリーにあるファイル。
 - インストール時に IIIH 上に生成された `/tmp/ibm.db2.cluster.*` ファイル。
 - すべてのリモート・ホストの `/tmp/db2log` ディレクトリーにあるファイル。
- `$HOME/sqllib` ディレクトリー内の以下のファイル:
 - `$HOME/sqllib/db2nodes.cfg` ファイル
 - `$HOME/sqllib/db2dump/db2diag.log` ファイル。

`$HOME/sqllib` ディレクトリーが存在しない場合は、`/tmp/db2nodes.cfg*` および `/tmp/db2diag.log*` ファイルを収集してください。
- 各ホストの `/etc/services` ファイル。
- `/var/ct/db2domain/log/mc/*` ディレクトリーおよび `/tmp/db2_cluster_manager_spooling/` トレース・スプーリング・ディレクトリーにある、IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) のエラー・ログとトレース・ログ。
- 以下のコマンドの出力:
 - `db2instance -list` (インスタンス所有者として発行)
 - `lssam > /tmp/lssam.out`
 - `lslpp -l all | grep "sam.*" > /tmp/ls1pp_sam.out`
 - `lslpp -l all | grep gpfs > /tmp/ls1pp_gpfs.out`
 - `lslpp -l all | grep rsct > /tmp/ls1pp_rsct.out`
 - `lsrsrc IBM.Equivalency > /tmp/lsrsrc_IBM.Equivalency.out`
 - `lsrsrc IBM.Application > /tmp/lsrsrc_IBM.Application.out`
 - `db2greg -dump` (すべてのホスト)
 - `ps -fe | grep db2` および `ipcs` (すべてのホスト)
 - `id` (すべてのホストで、インスタンス所有者および `fenced` ユーザーとして発行)
 - `uname -a` (すべてのホスト)
 - `errpt -a` (すべてのホスト)
- `/var/adm/ras/` ディレクトリーにある GPFS クラスタ・ファイル・システム・ログ。
- インスタンス所有者の `db2dump` ディレクトリー (`$HOME/sqllib/db2dump`) にあるすべてのファイルを含む 1 つの `.zip` ファイルまたは `.tar` アーカイブ。

インストール、インスタンス作成、およびロールバック

DB2 インストーラーは DB2 バイナリーをインストールし、全ホストでのインスタンスを作成し (すべての DB2 メンバーと CF での GPFS クラスタおよび RSCT ピア・ドメインのセットアップ)、オプションでユーザーとグループを作成します。

このセクションには、DB2 pureScale 環境のインストール、インスタンス作成、およびロールバック処理中に発生する可能性がある問題の理解と解決に役立つ情報が含まれています。

DB2 セットアップ・ウィザードが起動しない

DB2 セットアップ・ウィザードの起動が失敗した場合、*DB2DIR/tmp* ディレクトリにあるエラー・ログを調べてください (*DB2DIR* は DB2 コピーのインストール場所を表しています)。

DB2 セットアップ・ウィザードが起動しない理由として、以下のような可能性があります。

- ご使用のシステムがインストールの前提条件を満たさない。インストールの前提条件を確認して、問題があれば対処してください。
- XServer が X Window クライアント・マシン上で実行されていない。
- ディスプレイがまだ正しくエクスポートされていない可能性があります。以下のコマンドを使ってディスプレイをエクスポートします。

```
export DISPLAY=IP_Address:0.0
```

ここで *IP_Address* は、インストールを起動するために使われる X Window クライアント・マシンの IP アドレスを表しています。

インストールおよびインスタンス作成時の問題に関する診断データの収集

DB2 ソフトウェアのインストール時、またはインスタンスの作成時に発生した問題を解決するために、診断データの収集が必要な場合があります。DB2 インストーラーによって DBI2047E エラー・メッセージが生成され、インストールまたはインスタンスの作成が失敗したという通知が出された場合は、**db2support** コマンドを使用して、診断データを自動収集できます。

始める前に

可能な限り多くのソースから診断データを収集できるようにするには、**root** 権限が必要です。非 **root** 権限を使用して **db2support** コマンドを発行すると、一部のソースの診断データを収集できないことがあります。

-host パラメーターを使用してリモート・ホストについての情報を収集するには、SSH 接続が必要です。

このタスクについて

db2support -install コマンドを使用して収集した情報は、IBM サポートが分析を実行するために必要になることがあります。診断データは、**db2support** コマンドを発行することにより、ディスク上のログ・ファイルと、その他診断コマンドの出力の両方から収集されます。

DB2 インストール・イメージとともに出荷される **db2support** コマンドは、DB2 製品のインストール後に使用できるコマンド・パラメーターの一部しかサポートしません。DB2 製品のインストールが完了するまで、**db2support** コマンド・パラメーターのうち、使用できるものは **-install** と **-host** パラメーターのみです。

手順

診断データを収集するには、以下のようにします。

1. DBI2047E エラー・メッセージの情報を参考にして障害がどのホスト（複数の場合もあり）で発生したのかを特定します。
2. **db2support** コマンドは、以下のいずれかの方法で発行します。

- ローカル・ホストの診断データを収集するには、次のコマンドを発行します。

```
db2support -install
```

- リモート・ホストの診断データを収集するには、次のコマンドを発行します。

```
db2support -install -host hostname
```

hostname は診断データの収集対象であるリモート・ホストの名前です。例えば、ホスト *hotellnx96* の診断データを収集するには、次のコマンドを発行します。

```
db2support -install -host hotellnx96
```

- 複数のホストの診断データを収集するには、次のコマンドを発行します。

```
db2support -install -host hostname_list
```

hostname_list は診断データの収集対象であるホストのコンマ区切りリストです。例えば、ホスト *hotellnx96*、*hotellnx97*、および *hotellnx98* の診断データを収集するには、次のコマンドを発行します。

```
db2support -install -host hotellnx96,hotellnx97,hotellnx98
```

タスクの結果

診断データは *db2support.zip* ファイルに収集されます。このファイルは現行ディレクトリーが書き込み可能であれば、そこに作成されます。書き込み可能でない場合、このファイルはホーム・ディレクトリーに入れられます。

例

以下の例に、**db2support -install** コマンドの一般的な出力を示します。この場合、診断データはローカル・ホストで収集されます。

```
_____ D B 2   S u p p o r t   _____
```

```
Output file is "/home/hotellnx96/db2docs1/db2support.zip"  
Time and date of this collection: 2010-11-01-10.06.16.559562
```

```
Creating directory /home/hotel1nx96/db2docs1/DB2SUPPORT
Starting collection on host hotel1nx96
  Creating directory /home/hotel1nx96/db2docs1/DB2SUPPORT/hotel1nx96_IIH
Collecting resources group information
Collecting user identity information
Collecting current process information
Collecting active interprocess communications facilities information
Collecting system information
Collecting detailed data on system and hardware errors
Collecting registry contents
Collecting GPFS information
Collecting configuration, log and trace information for RSCT
Collecting information about installed DB2 products
Collecting information about state of db2 instance
Collecting "/home/hotel1nx96/db2docs1/sql1lib/db2dump/db2diag.log"
Collecting /etc/services
Collecting /home/hotel1nx96/db2docs1/sql1lib/db2nodes.cfg
Collecting directory /home/hotel1nx96/db2docs1/sql1lib/db2dump
Collection on host hotel1nx96 completed
Compressing files in /home/hotel1nx96/db2docs1/DB2SUPPORT directory
Collecting /home/hotel1nx96/db2docs1/db2support.log
Collecting db2support.out
```

db2support is now complete.

An archive file has been produced: "/home/hotel1nx96/db2docs1/db2support.zip"

次のタスク

DB2 インストールまたはインスタンス作成上の問題のトラブルシューティングに IBM サポートと共同であたっている場合には、サポート担当者による分析のために db2support.zip ファイルをアップロードする方法が指示されることがあります。

インスタンス・リソースの作成でエラーが戻される

このシナリオ例では、インストール・プロセスの一部として DB2 インスタンスおよびクラスター・サービスのリソースを作成するときに障害が発生します。ここでの目的は、問題を判別して障害の原因を診断することです。

症状

初期の症状は、インストールの一部としてインスタンスを作成する手順で戻される次のようなエラーです。

```
Creating resources for the instance "db2inst1" has failed.
There was an error with one of the issued cluster manager commands. Refer to
the db2diag log file and the DB2 Information Center for details.
```

診断と解決

- /tmp/db2diag.log の中に次のようなメッセージがあるかどうか確認します。
 - Line # : 6884---2610-403 The resource is stale. または
 - Line # : 9578---2610-422 Cannot execute the command on node <hostname>. The resource manager IBM.RecoveryRM is not available.

注: これらのエラーが表示された場合、これは、IBM Tivoli System Automation for Multiplatforms (SA MP) リカバリー・リソース・マネージャー・デーモンで問題が発生したことを示しています。このデーモンは Tivoli SA MP の意思

決定エンジンとして機能し、システムでは IBM.RecoveryRM として識別されます。問題を診断するための診断データが Tivoli SA MP によって書き込まれます。

- Tivoli SA MP の診断データは、/var/ct/db2domain/log/mc/ (エラー・ログ)、/var/ct/db2domain/run/mc/ (コア・ダンプ)、および /tmp/db2_cluster_manager_spooling/ (デフォルト・トレース・ディレクトリー) の各ディレクトリーに書き込まれます。
- IBM サービス・チームおよび開発チームは、トラブルシューティングのためにトレース・ファイルとコア・ファイルを使用します。診断データを分析するよう IBM サービスに依頼する必要がある場合は、671 ページの『手動によるトレースおよびログ・ファイルの収集』にリストされているデータを収集してください。
- 以下の説明に従って、IBM Technical Support にデータをアップロードしてください。
 - Submitting diagnostic information to IBM Technical Support for problem determination
- IBM 技術サポートの Web サイトには役立つ情報が提供されており、症状やエラー・ログ・メッセージに基づいて既知の問題を識別できます。
 - DB2 for Linux UNIX and Windows サポート

DB2 コマンドによる Tivoli SA MP の状態の調査

このトピックでは、Tivoli SA MP および RSCT によって示される状態と、DB2 ツールによって戻される状態とをマップする方法について詳しく説明します。

シナリオ

情報を表示する Tivoli SA MP および RSCT のコマンドは安全に実行することができますが、Tivoli SA MP または RSCT で状態を変更したり、アクションを発生させたりするコマンドを実行することは推奨されません。これらのリソースは DB2 クラスタ・サービスでポリシー・セットアップを介して制御されるため、外部的な変更によって予期しないアクションが発生する可能性があります。

注: このトピックでは、非常に重要なマッピングだけを示します。

この情報を表示する DB2 クラスタ・サービス・ツール・コマンドは、次のとおりです。

- **db2cluster -list -alert**
- **db2instance -list**

Issam コマンドの実行時に、以下のシナリオが発生します。

- **リソースが障害発生オフライン状態**
 - 出力で示される内容によっては、障害発生オフライン・リソースは問題を示していない可能性があります。これが問題にマップされるかどうか確認するには、以下のコマンドを実行します。

–

1. **db2instance -list**

2. db2cluster -list -alert

- **db2cluster -list -alert** コマンドを実行して、コマンドから返される指示に従ってください。問題を解決できない場合には、次の指示に従ってください。
 - **db2start** コマンドが発行された場合は、688 ページの『インスタンスが開始しない』を参照してください。
 - **db2start** コマンドが発行されなかった場合は、IBM サービスに連絡してください。
- **db2cluster -list -alert** コマンド出力にアラートが示されている場合は、**db2cluster -clear -alert** コマンドを使って「オフラインに失敗」状態をクリアすることができます。Tivoli SA MP または RSCT コマンドを使ってこれらの状態をクリアすることは推奨されません。
- アイドル・リソースでの「オフラインに失敗」状態は、**db2cluster -cm -clear -alert** コマンドを実行してもクリアできない場合があります。詳しくは、710 ページの『アイドル・リソースでの「オフラインに失敗」状態のクリア』を参照してください。
- **リソースがオンライン保留中**
 - **db2start** コマンドが発行された場合、またはリソースの再始動が行われている場合には、リソースがオンラインになるのを待ちます。
 - 後で状態が「オフラインに失敗」に変わった場合は、『リソースが障害発生 オフライン状態』を参照してください。
 - **db2instance -list** を実行することにより、インスタンスが始動済みかどうかを確認します。インスタンスが停止済みで、ユーザーによる **db2start** 発行を待機している場合には、`lssam` 出力に「オンライン保留中」状態が示されることがあります。
- **リソースがオフライン状態**
 - **db2stop** が発行され、既に完了しています。
 - **db2start** が既に発行され、従属リソースが使用不可である可能性があります。クリアまたは調査すべきアラートが存在するかどうか確認するには、**db2cluster -list -alert** を発行してください。例えばネットワーク・アダプターが誤動作を起こしていたり、ホストが使用不可である可能性があります。
- **リソースがオフライン保留中**
 - **db2stop** が発行された場合、リソースがオフラインに変わるのを待ちます。
 - **db2stop** が発行されなかった場合、これは、メンバーがホーム・ホストに戻った結果である可能性があります。再稼働を確認するには **db2instance -list** を実行します。オフライン保留中を発生させた可能性のある問題が存在しないことを確認するには、**db2cluster -list -alert** を実行します。オフライン保留中がオフラインに移行した場合は、『リソースがオフライン状態』のセクションを参照してください。
- **リソースが不明**
 - これは過渡的な状態です。`lssam` を再実行して、この状態が自然にクリアされるかどうか確認します。
 - マシンがオフライン状態で、その結果としてモニターを実行できない可能性があります。

- ホストがアクティブ状態であるかどうか確認するために **db2instance -list** を実行します。ホストがアクティブでない場合、IBM サービスに連絡してください。
- **リソース・グループの状態が不適切**
 - これは過渡的な状態である可能性があります。 **lssam** を再実行して、この状態が自然にクリアされるかどうか確認します。
 - 問題が続く場合は、次を取るべき手順を詳しく調べるために **db2instance -list** および **db2cluster -list -alert** を実行してください。

インスタンス・リソースの作成でインストールがハングする

このシナリオ例では、インストール・プロセスの一部として DB2 インスタンスおよびクラスター・サービスのリソースを作成するときにハングします。ここでの目的は、問題を判別してハングの原因を診断することです。このシナリオには 2 つのケースがあります。

ケース 1: インストール・ログまたは db2diag ログ・ファイルの中にメッセージが存在する

症状

このケースの初期症状は、インストール・プロセス中のハングです。より具体的には、インスタンス作成中にハングが発生します。

診断と解決:

- /tmp/db2setup.log または /tmp/db2icrt.log (またはその両方) を検査します。この例では、以下のメッセージが存在します。

```
Creating resources for the instance "db2inst1" has failed.
There was an error with one of the issued cluster manager commands.
Refer to the db2diag log file and the DB2 Information Center for details.
```

これと類似したメッセージがあるかどうか確認してください。

- /tmp/db2diag.log の中に次のようなメッセージがあるかどうか確認します。
 - Line # : 6884---2610-403 The resource is stale. または
 - Line # : 9578---2610-422 Cannot execute the command on node <hostname>. The resource manager IBM.RecoveryRM is not available.
- これらのエラーが表示された場合、これは、IBM Tivoli System Automation for Multiplatforms (SA MP) リカバリー・リソース・マネージャー・デーモンで問題が発生したことを示しています。このデーモンは Tivoli SA MP の意思決定エンジンとして機能し、システムでは IBM.RecoveryRM として識別されます。問題を診断するための診断データが Tivoli SA MP によって書き込まれます。
- Tivoli SA MP の診断データは、/var/ct/db2domain/log/mc/ (エラー・ログ用)、/var/ct/db2domain/run/mc/ (コア・ダンプ)、および /tmp/db2_cluster_manager_spooling/ (デフォルト・トレース・ディレクトリー) の各ディレクトリーに書き込まれます。
- IBM サービス・チームおよび開発チームは、トラブルシューティングのためにトレース・ファイルとコア・ファイルを使用します。診断データを分析するよう

IBM サービスに依頼する場合は、671 ページの『手動によるトレースおよびログ・ファイルの収集』のトピックでリストされているデータを収集してください。

- 以下の説明に従って、IBM Technical Support にデータをアップロードしてください。
 - Submitting diagnostic information to IBM Technical Support for problem determination
- IBM 技術サポートの Web サイトには役立つ情報が提供されており、症状やエラー・ログ・メッセージに基づいて既知の問題を識別できます。
 - DB2 for Linux UNIX and Windows サポート

ケース 2: インストール・ログや db2diag ログ・ファイルの中にエラーやメッセージが存在しない

症状

- 初期症状は、インストール・プロセス中のハングです。
- ハング状態が発生したとき、/tmp/db2setup.log インストール・ログの中にメッセージが報告されていないか、あるいはログが存在しないこともあります

診断と解決:

- /tmp/db2setup.log または /tmp/db2icrt.log (またはその両方) が存在する場合、以下のようなメッセージが存在するかどうか確認します。
 - Creating resources for the instance "db2inst1" has failed.
 - There was an error with one of the issued cluster manager commands. Refer to the db2diag log file and the DB2 Information Center for details.
- /tmp/db2diag.log が存在する場合、以下のようなメッセージが存在するかどうか確認します。
 - Line # : 6884---2610-403 The resource is stale. または
 - Line # : 9578---2610-422 Cannot execute the command on node <hostname>. The resource manager IBM.RecoveryRM is not available.
- これらのエラーが表示された場合、これは、Tivoli SA MP リカバリー・リソース・マネージャー・デーモンで問題が発生したことを示しています。このデーモンは Tivoli SA MP の意思決定エンジンとして機能し、システムでは IBM.RecoveryRM として識別されます。問題を診断するための診断データが Tivoli SA MP によって書き込まれます。
- Tivoli SA MP の診断データは、/var/ct/db2domain/log/mc/ (エラー・ログ用)、/var/ct/db2domain/run/mc/ (コア・ダンプ)、および /tmp/db2_cluster_manager_spooling/ (デフォルト・トレース・ディレクトリー) の各ディレクトリーに書き込まれます。
- /tmp/db2setup.log ファイルや /tmp/db2diag.log ファイルが存在しない場合、または空である場合には、671 ページの『手動によるトレースおよびログ・ファイルの収集』のトピックでリストされている残りのデータをできるだけ多く収集してください。IBM サービスに連絡して支援を依頼してください。

- 以下の説明に従って、IBM Technical Support にデータをアップロードしてください。
 - Submitting diagnostic information to IBM Technical Support for problem determination

DB2 pureScale Feature のホスト検証の問題に関するよくある質問

以下のセクションでは、リモート・ホストを検証しようとしたときに発生する可能性のある問題についての解決策を示します。

指定したユーザー ID が無効な場合、どうすればよいでしょうか？

DB2 インストーラーは、指定された既存のユーザー ID を妥当性検査します。指定したユーザー ID が無効である場合、インストール開始ホスト (IIH) および各リモート・ホストにそのユーザー ID が存在することを確認してください。ユーザー ID の uid、gid、グループ名、およびディレクトリー・パスが正確に同じであることを確認します。ユーザー ID を作成するには、『DB2 pureScale Feature インストールに必要なユーザー』のトピックを参照してください。

指定したポートが無効な場合、どうすればよいでしょうか？

インスタンスの作成前に、DB2 インストーラーは指定されたポートを検証します。指定されたポートが使用不可の場合、/etc/services ファイルを開くことにより、IIH およびリモート・ホストで既に使用中の TCP/IP ポートを確認してください。空きポートの範囲を判別した後、「ホストの追加 (Add Host)」パネルの「拡張設定 (Advanced Settings)」パネルで、適用されるポートのフィールドに新しいポートを入力し、DB2 インストーラーによってそれらを検証します。

ホストが別の GPFS クラスタに既に属している場合、どうすればよいでしょうか？

各ホストは 2 つの GPFS クラスタのメンバーになることはできません。IIH が GPFS クラスタに含まれない場合、リモート・ホストは GPFS クラスタに属してはなりません。IIH が GPFS クラスタに含まれる場合、リモート・ホストは GPFS クラスタに属さないか、または同じ GPFS クラスタに属していなければなりません。これを解決した後、ホスト・リストにホストを追加するよう再試行してください。

ホスト上のグローバル・レジストリー変数レコードに、GPFS クラスタが既に存在することが示される場合、どうすればよいでしょうか？

場合によっては、グローバル・レジストリーのクリーンアップが完了せず、ホスト上に DB2 で使用される GPFS クラスタが存在することを示すレコード (GPFS_CLUSTER) が残ることがあります (実際には存在しないにもかかわらず)。IBM ソフトウェア・サポートに連絡してください。

IIIH がターゲット・ホストに接続できない場合、どうすればよいでしょうか？

DB2 インストーラーが IIIH からターゲット・ホストへの ssh 通信を検証しようとしたときに、このエラーが発生する可能性があります。ssh および scp の設定が正しくセットアップされていることを確認してください。ssh と scp は、root ユーザーに関するホスト間のパスワード・プロンプトなしでセットアップされる必要があります。詳しくは、『OpenSSH on AIX のインストールとセットアップ』のトピックを参照してください。

DB2 インストーラーの外部で問題が発生する場合、さまざまな項目を検査して問題の発生源を判別することができます。例えば物理接続の問題、ネットワーク・アダプター・ドライバーの欠陥や無応答、不適切なネットワーク構成などの結果としてこの通信障害が生じることがあります。

ターゲット・ホストとの通信試行がタイムアウトになった場合、どうすればよいでしょうか？

ホストを検証する試行がタイムアウト期間内に完了しない場合、タイムアウトが発生します。ホストとの接続を検査して、再び追加を試みてください。また、タイムアウト変数を変更して、インストール・コマンドを再実行することもできます。

ホスト上の既存の RSCT ピア・ドメインが DB2 インストーラーによって検出された場合、どうすればよいでしょうか？

DB2 インストール時には、ただ 1 つの RSCT ピア・ドメインだけがアクティブであることができます。IBM DB2 pureScale Feature によって使われる RSCT ピア・ドメインを作成する前に、DB2 インストーラーによって作成されたものではないピア・ドメインを停止または除去する必要があります。

ホスト db2host1 上で db2cluster コマンドを使って RSCT ピア・ドメインを停止するには、同じアクティブ RSCT ピア・ドメインに属するホストにログオンして、db2cluster -cm -stop -host db2host1 コマンドを実行します。db2cluster コマンドを使用できない場合、stoprpdomain <domain name> コマンドを実行します。指定すべきドメイン・ネームを判別するには lsrpdomain コマンドを実行します。

ホストを検証する試行がタイムアウト期間内に完了しない場合、タイムアウトが発生します。ホストとの接続を検査して、再び追加を試みてください。また、タイムアウト変数を変更して、インストール・コマンドを再実行することもできます。

RSCT ピア・ドメインを除去するには、次のようにします。

1. ホスト・リストからリモート・ホストを除去します。
2. リモート・ホスト "db2host1" が別の DB2 pureScale インスタンスに属する場合、"db2iupdt -drop" コマンドを使用して、その DB2 pureScale インスタンスからそれを除去します。

DB2 pureScale インスタンスからホストを削除するには、次のようにします。

1. 同じ DB2 pureScale インスタンスに属する別のホストにログオンします
2. 以下を実行します

```
db2iupdt [-d] -add -m|cf <host_name>:<interconnect_name> -u <fenced_id>  
<instance_owner>
```

DB2 インスタンスに属さないリモート・ホストを除去するには、以下のいずれか 1 つを実行します。

- `db2cluster -cm -remove -host <hostname>`
- `rmrpnode <hostname>`

DEFAULT_INSTPROF レコードと、指定したインスタンス共有ディレクトリーの間競争がある場合、どうすればよいでしょうか？

DB2 インストーラーは DEFAULT_INSTPROF レコードと、指定されたインスタンス共有ディレクトリーの間競争を検出しました。インスタンス共有ディレクトリーを指定しないでください。グローバル・レジストリー内の DEFAULT_INSTPROF レコードは、インスタンス共有ファイル・システムが DB2 によって既にセットアップ済みであることを示しています。

この場合、以下のオプションやキーワードは必要ありません。

- 応答ファイル・インストールの場合、INSTANCE_SHARED_DEVICE_PATH および INSTANCE_SHARED_DIR
- db2icrt または db2iupdt の場合、instance_shared_dev および instance_shared_dir。

INSTANCE_SHARED_DIR / instance_shared_dir の値が既存のインスタンス共有ファイル・システムのマウント・ポイントとマッチする場合、DB2 インストーラーは正常に処理を進めます。しかし値がマッチしない場合には、インストールが失敗します。

インストール開始ホスト (IIH) での装置パスの PVID がターゲット・ホスト上に存在しない場合、どうすればよいでしょうか？

IIH およびすべてのホストで PVID がセットアップされていることを確認します。詳しくは、『DB2 pureScale Feature インスタンス用の PVID の構成』のトピックを参照してください。

指定したクラスター相互接続ネット名が無効な場合、どうすればよいでしょうか？

このエラーは、DB2 インストーラーがクラスター相互接続ネット名を識別できないために発生します。以下の方法で問題の解決を試みてください。

1. コマンドのタイプミス・エラーを確認します。
2. マシンのネットワーク構成を確認します (例えば /etc/hosts を確認します)。
3. ping および nslookup ツールを使用して、すべてのホストで同じ IP アドレスにクラスター相互接続ネット名がマッピングされることを確認します。逆に、すべてのホストで同じクラスター相互接続ネット名に IP アドレスがマッピングされることを確認します。

ターゲット・ホストのクラスター相互接続ネット名が IIH の ping に失敗した場合、どうすればよいでしょうか？

新しいホストを追加するとき、DB2 インストーラーは、その新しいホストが IIH に小さなデータ・パケットを送って応答を受け取ることができるかどうかを検査します。通常、この送信/返信テストを ping と呼びます。このメッセージは ping が失敗した結果として出されます。問題が存在する場合、リモート・ホストのコンソールから ping IIH address_or_name というコマンド行を実行して、結果を検証することができます。

このテストが失敗する場合、IIH とリモート・ホストの間の ssh 通信が正しくセットアップされていることを確認してください。

DB2 インストーラーの外部で問題が発生していることが確認された後には、問題の発生源を見つけるためにさまざまな事柄を検査できます。例えば物理接続の問題(ケーブルの緩みなど)、ネットワーク・アダプター・ドライバーの欠陥、不適切なネットワーク・セットアップなどです。ネットワーク・アダプターとケーブルを検査するか、別のものを選んでください。

ターゲット・ホストのクラスター相互接続ネット名が IIH と同じサブネットでない場合、どうすればよいでしょうか？

ネットワーク・アダプターを再構成するか、別のものを選ぶことができます。このエラーは、ホストのクラスター相互接続ネット名がインストール開始ホストと同じサブネット上にない場合に発生します。パフォーマンス上の理由で、CF のすべてのクラスター相互接続は同じサブネットに存在する必要があります(同じサブネット内のすべてのホストは、通常、1 回のルーティング・ホップで到達可能です)。

例えば、ネットワーク・アドレス 192.168.0.0/24 を使ってクラスター相互接続ネットワークが構成される場合、すべてのクラスター相互接続ネット名アドレスは 192.168.0 で始まる必要があります(例えば 192.168.0.1、192.168.0.2 など)。

新しいホスト上のネットワーク・カード構成を検査して(例えば ifconfig -a コマンドを実行)、アドレスの代わりに名前が使われていないか /etc/hosts を確認します。

DB2 pureScale Feature のインストール、インスタンス作成、およびロールバックの問題に関するよくある質問

この「よくある質問」リストの答えを使用すると、IBM DB2 pureScale Feature のインストール・プロセス中に発生する可能性がある問題を解決するのに役立つ可能性があります。

デバッグに役立つトレース・ファイルを生成するために、デバッグ (-d) パラメーター付きですべてのインストール・コマンドまたは作成コマンドを発行してください。

インストール・ログ・ファイルはどこにありますか？

インストールの問題のトラブルシューティングのために最も重要な情報源は、インストール・ログ・ファイルです。DB2 セットアップ・ログ・ファイル(デフォルト

では db2setup.log) は、エラーを含むすべての DB2 インストール情報を取り込みます。DB2 エラー・ログ・ファイル db2setup.err は、Java アプリケーションによって戻されるエラー出力 (例えば、例外やトラップの情報) をすべて取り込みます。

デフォルトではログ・ファイルは *DB2DIR/tmp* ディレクトリーにあります。

db2setup コマンドの発行時にこのデフォルトをオーバーライドした可能性もあります。DB2 インストーラーは DB2 セットアップ・ログ・ファイルを *DB2DIR/install/logs/* ディレクトリーにコピーして、db2install.history という名前に変更します。インストール開始ホスト (III) 上のログ・ファイルだけでは詳細情報が不足する場合、各リモート・ホストでのアクションを詳しく記述したホスト・ログを確認することもできます。これらは *DB2DIR/tmp/db2log.xxxx* ディレクトリー (xxxx はプロセス ID を表す) および *DB2DIR/tmp/db2log* ディレクトリーにあります。

ロールバックとは何ですか？

操作が失敗して DB2 インストーラーが環境をクリーンアップするときには、ロールバックが発生します。インスタンス・ロールバックおよび DB2 バイナリー・ロールバックが発生する可能性があります。DB2 インストーラーが DB2 インスタンスの作成に失敗した場合、すべてのホストでのインスタンスのロールバックを起動します。インスタンス・ロールバックによって DB2 バイナリーがロールバックされることはありません。

インスタンスのセットアップが可能であっても、それを特定のホストに拡張できない場合には、部分的なロールバックが発生します。

1 つ以上のホストで DB2 バйнаリーのインストールが失敗した場合、どうすればよいのでしょうか？

III 上のインストール・ログを確認します。障害に関するさらに詳しい情報が必要な場合は、該当するリモート・ホストのログを確認します。エラーを解決して、インストールを再開します。

インスタンスが作成されなかった場合、どうすればよいのでしょうか？

インスタンスの作成が失敗した場合、関連するすべてのホストでのインスタンス・ロールバックが発生します。詳しくはインストール・ログを参照してください。DB2 バイナリーが正常にホストにインストールされた場合、それらはロールバックされません。インストール・ログに記述されたエラーを解決し、適切なコマンド (**db2isetup** または **db2icrt** コマンド) を実行してインスタンスを再作成してください。DB2 インストーラーを起動する必要はあまりせん。

インスタンスの一部だけが作成された場合、どうすればよいのでしょうか？ インスタンスが作成されても、いくつかのメンバーや CF が作成されなかった場合には、どうすればよいのでしょうか？

インストール中に、一部のターゲット・ホストでのみインスタンスが作成されることがあります。インスタンス作成が完了しなかったホスト上ではインスタンス・ロ

ールバックが発生します。その際、バイナリー・ロールバックは起動されません。エラーのためにインスタンスに含まれなかったホストを示す、インストール後のメッセージが受信されます。

インストール・ログに記述されたエラーをすべて解決してください。問題を解決した後、**db2isetaup** または **db2iupdt** コマンドを実行してメンバーまたは CF を次のように追加できます。

- (**db2isetaup** コマンドを発行して) DB2 インスタンス・セットアップ・ウィザードを使用すると、ホストを検証してインスタンスにそれを追加することができます。
- 既にインスタンスに属するホストから **db2iupdt -add** コマンドを発行することにより、インスタンスを拡張できます。

DB2 クラスタ・ファイル・システムの問題に関するよくある質問

以下の情報を使用すると、(GPFS に基づく) DB2 クラスタ・ファイル・システムの問題の解決策を判別することができます。

GPFS クラスタの作成が失敗した場合、どうすればよいでしょうか？

インスタンス作成時に、GPFS クラスタの作成が失敗することがあります。この失敗により、インスタンス作成のロールバックが起動します。DB2DIR/tmp ディレクトリーにあるログ・ファイルを確認してください。インスタンスを作成するには **db2cluster** コマンドが使用されます。**db2cluster** コマンド・ログ・ファイルは DB2DIR/tmp/ibm.db2.cluster.* にあります。このログ・ファイルの内容を確認した後、IBM ソフトウェア・サポートに連絡してください。

ターゲット・ホストへの GPFS クラスタの拡張が失敗した場合、どうすればよいでしょうか？

ホストへの GPFS クラスタ拡張が失敗した場合、DB2DIR/tmp ディレクトリーにあるログ・ファイルを確認して、そのホストで拡張できなかった理由を判別することができます。

ターゲット・ホスト上の "GPFS_CLUSTER" グローバル・レジストリー変数レコードが作成されなかった場合、どうすればよいでしょうか？

他のエラーの場合と同様に、インストール・ログで詳細を調べてください。GPFS クラスタが適切に作成されていることを確認します。指定されたホスト上で GPFS_CLUSTER グローバル・レジストリー変数レコードが作成されなかったことがログに示されている場合、IBM ソフトウェア・サポートに連絡してください。

インストールの失敗時に GPFS クラスタが削除されなかった場合、どうすればよいでしょうか？

DB2 製品のインストールが失敗した結果としてロールバックが発生した場合、DB2 インストーラーは、新規作成された GPFS ファイル・システムを除去できないことがあります。GPFS クラスタが除去されなかった理由を判別するには、

DB2DIR/tmp ディレクトリーにあるログ・ファイルを確認してください。

RSCT ピア・ドメインの問題に関するよくある質問

以下のセクションでは、RSCT ピア・ドメインの問題についての解決策を示します。

ターゲット・ホスト上の "PEER_DOMAIN" グローバル・レジストリー変数レコードが作成されなかった場合、どうすればよいでしょうか？

他のエラーの場合と同様に、インストール・ログで詳細を調べてください。指定されたホスト上で PEER_DOMAIN グローバル・レジストリー変数レコードが作成されなかったことがログに示されている場合、IBM ソフトウェア・サポートに連絡してください。

Tivoli SA MP リソースの作成が失敗した場合、どうすればよいでしょうか？

インスタンス作成時に Tivoli SA MP リソースの作成が失敗しました。このエラーにより、インスタンスのロールバックが起動しました。ログ・ファイルを調べて、Tivoli SA MP リソースが失敗した理由を判別してください。

ログ・ファイルは *instance_user/sql1lib/db2dump/db2diag.log* にあります。ロールバック処理により *instance_user/sql1lib* の内容がロールバックされるため、ファイルは */tmp/db2diag.log.process_id* としてバックアップされます。このエラーの原因はいくつかあります。1つの原因として、*db2nodes.cfg* が間違っているか、その作成が失敗した可能性があります。このファイルは *instance_user/sql1lib/db2nodes.cfg* にあります。

問題を修正し、**db2icrt** コマンドを実行してください。問題を解決できない場合には、すべてのログ・ファイルとトレース・ファイルを用意して IBM ソフトウェア・サポートに連絡してください。

DB2 pureScale Feature での応答ファイル・インストールの問題に関するよくある質問

このトピックは、インストール時の応答ファイルの問題を解決するうえで役立ちます。

生成された応答ファイルはどこに保存されますか？

デフォルトでは、DB2 セットアップ・ウィザードまたは DB2 インスタンス・ウィザードによって生成された応答ファイルは *DB2DIR/tmp/* ディレクトリーに保存され、*db2ese_dsf.rsp* という名前になります。*DB2DIR* は DB2 コピーのインストール場所を表しています。

DB2 セットアップ・ウィザードまたは DB2 インスタンス・ウィザードの「インストールおよび応答ファイルの作成を選択」パネルの「応答ファイル名」フィールドで、応答ファイルの名前と場所を指定することができます。DB2 メンバーとして指定されるホストからの応答ファイルを保存する必要があります。

IBM DB2 pureScale Feature のサンプル応答ファイルはどこにありますか？

DB2 pureScale Feature のサンプル応答ファイル `db2dsf.rsp` は `DB2DIR/install/db2/platform/samples` ディレクトリーにあります (`platform` は該当するオペレーティング・システム)。

DB2 pureScale Feature のインストールでは、どのキーワードが必須ですか？

DB2 pureScale Feature のリリースで導入された新しいキーワードの中には、作成する環境に応じて必須となるものがあります。環境によってどのキーが必須となるかを判別するには、キーワードのリストを確認してください。詳しくは『応答ファイルのキーワード』のトピックを参照してください。

DB2 pureScale Feature のライセンスの問題に関するよくある質問

このトピックの情報を使用すると、IBM DB2 pureScale Feature のライセンスの問題を解決するうえで役立ちます。

ホストがライセンス交付を受けていない場合、どうすればよいでしょうか？

以下のいずれかの状況が生じて、ライセンスの問題が見つかることがあります。

- 行レベル、表、または索引の圧縮を使用しようとして、ライセンス交付エラー・メッセージ `SQL8029N` が出力に表示され、`db2diag` ログ・ファイルに記録される
- `db2licm -g` コマンドを実行して準拠レポートを生成しようとする、警告メッセージ `ADM12026W` がレポートに表示される。

DB2 pureScale Feature のライセンスは DB2 インストーラーによって自動的にインストールされます。DB2 pureScale Feature のインストール・イメージには DB2 Enterprise Server Edition コードのコピーが含まれますが、DB2 Enterprise Server Edition バージョン 9.7 のライセンス資格は付属していません。クラスター内のすべてのユーザーやプロセッサを対象とするのに十分な DB2 Enterprise Server Edition バージョン 9.7 ライセンス資格を予め別途購入していただく必要があります。また、DB2 pureScale Feature のホストで DB2 Enterprise Server Edition ライセンスを登録する必要もあります。

DB2 バージョン 9.7 のライセンス・キーはどこにありますか？

ライセンス・キーは、製品パッケージに含まれるバージョン 9.7 アクティベーション CD に入っています。または、Passport Advantage[®]の Web サイト (<http://www.ibm.com/software/lotus/passportadvantage>) からダウンロードすることもできます。その後、各ホストで `db2licm` コマンドを使ってライセンスを登録できます。

すべてのホストのライセンス状況をどのように確認できますか？

`db2licm` コマンドを各ホスト上で実行する必要があります。ライセンス状況をリストするには、`DB2DIR/adm/db2licm -l` コマンドを実行します。コマンド出力に「ラ

ライセンス登録なし」と表示された場合、そのホストで DB2 Enterprise Server Edition ライセンスを登録する必要があります。

欠落しているライセンスをどのように登録できますか？

欠落しているライセンスを登録するには、適切なライセンス資格がないホスト上で、以下のように追加オプションを使って **db2licm** コマンドを実行する必要があります。

```
DB2DIR/adm/db2licm -a license-certificate-file-name
```

インストール後

このセクションには、IBM DB2 pureScale Feature のインストール後に発生する可能性がある問題の理解、切り分け、および解決に役立つ情報が含まれています。

インスタンスが開始しない

インスタンスが開始しない場合、コンポーネントの障害が原因である可能性があります。このトピックの情報は、メンバー・コンポーネントまたはクラスター・キャッシング・ファシリティ (CF) コンポーネントに障害が発生したかどうか診断するうえで役立ちます。

症状

db2start コマンドを実行したときに、DB2 インスタンスの開始が失敗します。

メンバーの始動の障害を診断する

- **db2start** コマンド出力の中の **SQLCODE** を参照します。
- **db2instance -list** コマンドを発行することにより、各メンバーの状態を表示できます。始動が失敗した場合、障害がいつ発生したかに応じて、**STOPPED** または **ERROR** 状態のメンバーがこの情報に示される可能性があります。
- **db2instance -list** コマンドの出力例を以下に示します。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	...
0	MEMBER	ERROR	host01	host01	
1	MEMBER	STARTED	host02	host02	
2	MEMBER	STARTED	host03	host03	
128	CF	PRIMARY	host04	host04	
129	CF	PEER	host05	host05	

ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME	...
YES	0	0	host01-ib0	
NO	0	0	host02-ib0	
NO	0	0	host03-ib0	
NO	-	0	host04-ib0	
NO	-	0	host05-ib0	

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
host01	ACTIVE	NO	NO
host02	ACTIVE	NO	NO

```

host03      ACTIVE                NO          NO
host04      ACTIVE                NO          NO
host05      ACTIVE                NO          NO

```

- リストされるアラートについて詳細情報を得るには、**db2cluster -cm -list -alert** コマンドを実行します。例えば、**db2cluster -cm -list -alert** コマンドによって以下のようなアラートが返される可能性があります。

```

Alert: DB2 member '0' failed to start on its home host 'host01'. The
cluster manager will attempt to restart the DB2 member in restart
light mode on another host. Check the db2diag log file for messages
concerning failures on hosts 'host01' for member '0'

```

Action:

```

This alert must be cleared manually with the command:
db2cluster -cm -clear -alert.

```

```

Impact: Member 0 will not be able to service requests until
this alert has been cleared and the member returns to its home host.

```

- <instance_owner>.nfy log を調べて、障害がいつ発生したかについての情報を探します。障害が発生した理由の詳細について、このメンバーの db2diag ログ・ファイルで項目を確認します。db2diag ログ・ファイルで db2rstar または db2rstop に関連するエラー・メッセージを探します。
- さらに、エラーの原因がまだ分からない場合は、影響を受けたホストのシステム・エラー・ログを調べることもできます。例えば、
 - 前述の出力では、メンバー 0 は開始されていません。
 - host01 にログインし、**errpt -a** コマンドを実行するか (AIX の場合)、/var/log/messages を確認して (Linux の場合)、システム・エラー・ログを表示します。
 - システム・エラー・ログで、障害が発生した時点の該当するログ項目を探します。
- **db2cluster -list -alert** からアラートが示された場合は、問題を解決した後で **db2cluster -clear -alert** を実行し、**db2start** コマンドを再実行します。

CF サーバーの障害の診断

- **db2start** コマンド出力の中の SQLCODE を参照します。
- CF がまだ始動済みでないことを判別するために、**db2instance -list** を実行します。始動が失敗した場合、障害がいつ発生したかに応じて、STOPPED または ERROR 状態のCFがこの情報に示される可能性があります。
 - **db2instance -list** の出力例を以下の例に示します。

```

ID      TYPE      STATE      HOME_HOST  CURRENT_HOST  ...
--      ----      -
0       MEMBER    STOPPED    host01     host01
1       MEMBER    STOPPED    host02     host02
2       MEMBER    STOPPED    host03     host03
128     CF        STOPPED    host04     host04
129     CF        STOPPED    host05     host05

```

```

ALERT      PARTITION_NUMBER  LOGICAL_PORT  NETNAME  ...
-----
NO         0                  0             host01-ib0
NO         0                  0             host02-ib0
NO         0                  0             host03-ib0
NO         -                  0             host04-ib0
NO         -                  0             host05-ib0

```

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
-----	-----	-----	-----
host01	ACTIVE	NO	NO
host02	ACTIVE	NO	NO
host03	ACTIVE	NO	NO
host04	ACTIVE	NO	NO
host05	ACTIVE	NO	NO

- 何らかのアラートが存在する場合、詳細情報を調べるために **db2cluster -cm -list -alerts** を実行します。アラートは、修正が必要と思われる点 (例えばネットワーク・アダプターまたはホストのオフライン状態) に関する詳細情報を示したり、`cfdiag*.log` ファイルで詳細情報が得られることを示したりします。
- CF の `db2diag` ログ・ファイルの中で、**db2start** コマンドが実行されたときの処理に関連するエラーを探します。

```
2009-11-09-02.32.46.967563-300 I261372A332          LEVEL: Severe
PID      : 1282088                TID   : 1          KTID   : 4751433
PROC     : db2start
INSTANCE: db2inst1              NODE   : 000
HOSTNAME: host04
EDUID    : 1
FUNCTION: DB2 UDB, base sys utilities, sqlcIssueStartStop, probe:3973
MESSAGE  : Failed to start any CF.
```

- CF が始動しなかった詳しい理由を、`db2diag` ログ・ファイルの、直前のトレース・ポイントより前のセクションで探します。例えばクラスター・サービスが CF を始動できない場合、`db2diag` ログ・ファイルは次のようになります。

```
2009-11-09-02.12.40.882897-300 I256778A398          LEVEL: Error
PID      : 737522                TID   : 1          KTID   : 2371807if
PROC     : db2havend
INSTANCE: db2inst1              NODE   : 000
EDUID    : 1
FUNCTION: DB2 UDB, high avail services, db2haOnlineResourceGroup, probe:5982
DATA #1  : <preformatted>
Timeout waiting for resource group ca_db2inst1_0-rg to be online, last known
OpState is 2
```

- 各 CF は `cfdiag*.log` に情報を書き込み、より詳しい診断データをダンプします (必要に応じて)。これらのファイルの格納場所は、データベース・マネージャ構成パラメーター **cf_diagpath** で設定されたディレクトリです。設定されていない場合は **diagpath** の場所、または `$INSTHOME/sqlllib_shared/db2dump/ $m` (デフォルト) です。

- CF 診断ログ・ファイル (`cfdiag-<timestamp>.<cf_id>*.log`)

- これらの各ファイルには、CF に関連するアクティビティのログが保持されます。イベント、エラー、警告、または追加のデバッグ情報がログに記録されます。このログの構造は `db2diag` ログ・ファイルと同様です。CF が始動するたびに、新しいログが作成されます。ロギング・レベルは、**cf_diaglevel** データベース・マネージャ構成パラメーターによって制御されます。
- それぞれの CF について常に最新の診断ロギング・ファイルを指し示し、`cfdiag.<cf_id>.log` という形式を持つ、静的 CF 診断ログ名があることに注意してください。

- CF 出力ダンプ診断ファイル `cfdump.out.<cf_pid>.<hostname>.<cf_id>`

- これらのファイルには、CF の始動と停止に関する情報が含まれます。これらのファイルの中に、追加の出力が含まれる場合があります。

- Management LightWeight Daemon 診断ログ・ファイル (mgmnt_lwd_log.<cf_pid>)
 - このログ・ファイルは、特定の CF の LightWeight Daemon (LWD) プロセスに関するログ項目を示します。このログ・ファイル内のエラーは、LWD が正しく始動しなかったことを示しています。
- CF のスタック・ファイル (CAPD.<cf_pid>.<tid>.thrstk)
 - これらは、CF がシグナルを検出したときに生成されるスタック・ファイルです。これらのファイルは CF の問題を診断するうえで重要です。
- CF のトレース・ファイル (CAPD.trace.log.<cf_pid>)
 - CF に関するデフォルトの軽量トレースが使用可能です。
 - CF が強制終了または停止した場合、常にこれらのトレース・ファイルが生成されます。
 - これらのトレース・ファイルには CF の問題が示される可能性があります。これらのファイルは他の診断データと組み合わせて使用する場合には、エラーの診断に役立ちます。
- CF プロセスが正常に始動した場合、始動メッセージと初期化メッセージが CF ダンプ・ファイルに書き込まれます。
- 例えば cfdump.out.1548476.host04.128 の内容には、プロセスの始動が成功したことを示す次のようなメッセージが含まれます。

```
CA Server IPC component Initialised: LWD BG buffer count: 16
    Session ID: 1d
CA Server IPC component Acknowledged LWD Startup Message
    Waiting for LWD to Configure Server
Processors: (4:4) PowerPC_POWER5 running at 1498 MHz
```

```
Cluster Accelerator initialized
```

```
Cluster Accelerator Object Information:
OS: AIX 64-bit
Compiler: x1C VRM (900)
SVN Revision: 7584
Built on: Oct 12 2009 at 17:00:54
Executable generated with symbols
Model Components Loaded: CACHE LIST LOCK
Transport: uDAPL
Number of HCAs: 1
Device[0]: hca0
CA Port[0]: 50638
Mgmt Port Type: TCP/IP
Mgmt Port: 50642
IPC Key: 0xe50003d
Total Workers: 4
Conn/Worker: 128
Notify conns: 256
Processor Speed: 1498.0000 MHz
```

- cfdump.out.* ファイルに「クラスター・キャッシング・ファシリティー initialized」または「クラスター・キャッシング・ファシリティー Object Information」という行、および以下の例に示しているその他の行が含まれていない場合、CF プロセスは正常に始動しませんでした。代わりにエラー・メッセージが示される場合があります。詳しくは、IBM サポートに連絡してください。
- この例では、cfdiag-20091109015035000037.128.log の中にプロセス始動の成功が含まれています。CF が正しく始動しなかった場合、このログは空であるか、エラー・メッセージがそれに含まれる可能性があります。

```

2009-11-09-01.50.37.0051837000-300 E123456789A779 LEVEL : Event
PID      : 688182 TID :          1
HOSTNAME : host04
FUNCTION : CA svr_init, mgmnt_castart
MESSAGE  : CA server log has been started.
DATA #1  :
Log Level: Error
Debugging : active
Cluster Accelerator Object Information
  AIX 64-bit
  Compiler: x1C VRM (900)
  SVN Revision: 7584
  Built on Oct 12 2009 at 17:00:59
  Executable generated with symbols.
  Executable generated with asserts.
  Model Components Loaded: CACHE, LIST, LOCK
  Transport: uDAPL
  Number of HCAs: 1
  Device[0]: hca0
  CA Port[0]: 50638
  Total Workers: 4
  Conn/Worker: 128
  Notify conns: 256
  Processor Speed: 1498.000000 Mhz.
  Allocatable Structure memory: 170 MB

```

- CF_DIAGPATH ディレクトリー内のコア・ファイルまたはスタック・トレースバック・ファイルを探します。
- ただし CF が正常に始動しなかった場合には、関連するエラーが db2diag ログ・ファイルに存在する可能性があります。例えば、

```

2009-11-09-02.32.46.967563-300 I261372A332          LEVEL: Severe
PID      : 1282088          TID : 1          KTID : 4751433
PROC     : db2start
INSTANCE: db2inst1          NODE : 000
HOSTNAME: host04
EDUID    : 1
FUNCTION: DB2 UDB, base sys utilities, sqlcIssueStartStop, probe:3973
MESSAGE  : Failed to start any CA.

```

- このメッセージに近い時刻を調べて、関連するエラーを見つけます。例えばクラスター・サービスが CF を始動できない場合、db2diag ログ・ファイルは次のようになります。

```

2009-11-09-02.12.40.882897-300 I256778A398          LEVEL: Error
PID      : 737522          TID : 1          KTID : 2371807
PROC     : db2havend
INSTANCE: db2inst1          NODE : 000
EDUID    : 1
FUNCTION: DB2 UDB, high avail services, db2haOnlineResourceGroup, probe:5982
DATA #1  : <preformatted>
Timeout waiting for resource group ca_db2inst1_0-rg to be online, last known
OpState is 2

```

- CF_DIAGPATH ディレクトリー内のコア・ファイルまたはスタック・トレースバック・ファイルを探します。
- さらに、エラーの原因がまだ分からない場合は、影響を受けたホストのシステム・エラー・ログを調べることもできます。始動していない CF ホストにログオンし、**errpt -a** コマンドを実行してシステム・エラー・ログを表示します (Linux では、/var/log/messages ファイルを調べます)。障害が発生した時点に該当するログ項目を探します。ここに示した例では、CF 128 と CF 129 が常駐しているホスト host04 および host05 にログインします。

- `db2cluster -list -alert` からアラートが示された場合は、問題を解決した後で `db2cluster -clear -alert` を実行し、`db2start` コマンドを再発行します。

db2diag ログ・ファイルと適切な `cfdump.out*` ファイルとのマッピング

診断ログ・ファイルには、`db2start` コマンドが失敗した場合の関連情報を識別するうえで役立つ情報が含まれます。このトピックでは、`db2diag` ログ・ファイルを適切な `cfdump.out*` ファイルにマップする方法について詳しく示します。

このタスクについて

`db2start` コマンドが失敗した場合、`sql1lib/db2dump` ディレクトリーにある診断ログ・ファイルを検査します。データベース・マネージャー構成パラメーター `diagpath` および `cf_diagpath` は、失敗と関連があるのはどの `cfdiag*.log` ファイルであるかを決定します。

手順

`db2diag` ログ・ファイルの情報を適切な `cfdump*.out` ファイルにマップするには、次のようにします。

1. `db2diag` ログ・ファイルの内容を調べて、以下と同じような項目を見つけます。

```
2009-11-09-10.51.11.132220-300 E25081A841          LEVEL: Event
PID      : 463088          TID : 1          KTID : 3551445
PROC     : ca-wdog 129
INSTANCE: svtdbm9        NODE : 129
HOSTNAME: coralpib08
EDUID    : 1              EDUNAME: ca-wdog 129
FUNCTION: DB2 UDB, base sys utilities, sqlSendCFServerConfig, probe:1470
DATA #1 : String, 469 bytes
CF Server 129 Configuration:

    Number of worker threads : 1
    Memory size (MB)         : 3907
    Connections per worker   : 128
    Notify connections       : 256
    Number of HCA            : 1
        port[0]               : 50639
        device name[0]        : hca0
    Transport type           : UDAPL
    Dump file path           :
    Log path                 : /coralpgpfs2/svtdbm9/sql1lib/db2dump/
                             cfdiag-20091109105110000098.129.log
    Log level                : 2
```

2. `db2diag` ログ・ファイルには、関連する `cfdiag*.log` ファイルのパスと名前が `cfdiag-timestamp.cfmember#.log` という形式で表示されます。 `cfdiag*.log` ファイルを開いて確認してください。手順 1 の例を使用すると、`cfdiag-20091109105110000098.129.log` の中に例えば次のような項目が見つかります。

```
2009-11-09-10.53.31.3382320-300 E123456789A269    LEVEL   : Event
PID      : 26981 TID : 1075841344
HOSTNAME : coralpib08
FUNCTION  : CF svr_mgmt, dmon_config_cf
MESSAGE   : LWD Log shmem_id IPC message received by daemon.
DATA #1   :
Shared Memory Segment ID: 0
```

3. 手順 2 の cfdiag*.log 項目の PID および HOSTNAME の値を手順 1 の情報と合わせて使用すると、必要な cfdump*.out ファイルの名前を組み立てるのに必要な情報が得られます。cfdump*.out ファイルの形式は cfdump.out.pid.hostname.cfmember#.log です。この例では、ログは cfdump.out.26981.coralpib08.129.log です。

uDAPL over InfiniBand 通信エラーの識別

診断ログ、オペレーティング・システムのコマンドおよびシステム・トレースを使用すると、uDAPL 通信エラーを識別して解決することができます。

db2start を入力した後、データベース接続が最初にアクティブになったとき、またはメンバーが再始動されたときに、エラーが発生することがあります。**db2diag** ログ・ファイル内のメッセージの例を以下に示します。

```
2009-04-27-15.41.03.299437-240 I9450505A370      LEVEL: Severe
PID      : 651462                TID : 258          KTID : 2674775 PROC : db2sysc 0
INSTANCE: db2inst1             NODE : 000
EDUID    : 258                  EDUNAME: db2sysc 0
FUNCTION: DB2 UDB, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
ca_svr_connect: dat_evd_wait failed: 0xf0000
```

```
2009-04-27-15.41.03.363542-240 I9450876A367      LEVEL: Severe
PID      : 651462                TID : 258          KTID : 2674775 PROC : db2sysc 0
INSTANCE: db2inst1             NODE : 000
EDUID    : 258                  EDUNAME: db2sysc 0
FUNCTION: DB2 UDB, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
CACConnect: cmd_connect failed: 0x80090001
```

```
2009-04-27-15.41.03.421934-240 I9451244A1356      LEVEL: Severe
PID      : 651462                TID : 258          KTID : 2674775 PROC : db2sysc 0
INSTANCE: db2inst1             NODE : 000
EDUID    : 258                  EDUNAME: db2sysc 0
FUNCTION: DB2 UDB, Shared Data Structure Abstraction Layer ,
          SQLE_CA_CONN_ENTRY_DATA::sqlCaCeConnect, probe:622
MESSAGE : CA RC= 2148073473
DATA #1 : String, 17 bytes
CACConnect failed.
DATA #2 : CAToken_t, PD_TYPE_SD_CATOKEN, 8 bytes
0x07000000003EE0B8 : 0000 0001 1064 AF90          .....d..
DATA #3 : CA Retry Position, PD_TYPE_SAL_CA_RETRY, 8 bytes
0
```

```
CALLSTCK:
[0] 0x0900000012FF274C sqlCaCeConnect__23SQLE_CA_CONN_ENTRY_DATAFCP7CATokenC1
    + 0x40C
[1] 0x0900000012FF2CF8 sqlSingleCaCreateAndAddNewConnectionsToPool__
    21SQLE_SINGLE_CA_HANDLEFCU1T1Cb + 0x278
[2] 0x0900000012FF9188 sqlSingleCaInitialize__21SQLE_SINGLE_CA_HANDLEFRC27SQLE_
    CA_CONN_POOL_NODE_INFOFCU1P13SQLO_MEM_POOL + 0x448
[3] 0x0900000013001C50 sqlCaCpAddCa__17SQLE_CA_CONN_POOLFSCPU1 + 0x350
[4] 0x00000001000118AC sqlInitSysCtlr__FPiT1 + 0x140C
[5] 0x0000000100013008 sqlSysCtlr__Fv + 0x4A8
[6] 0x0900000012E15C78 sqloSystemControllerMain__FCUiPFv_iPfi_vPPvCPi + 0xD58
[7] 0x0900000012E177AC sqloRunInstance + 0x20C
[8] 0x0000000100006ECC DB2main + 0xAEC
[9] 0x0900000012C99048 sqloEDUMainEntry__FPcUi + 0xA8
```

また、**db2diag** ログ・ファイルには、次のようなメッセージが示されることもあります。

```

2009-04-27-15.41.04.595936-240 I9453087A387      LEVEL: Severe
PID      : 1249362          TID : 258          KTID : 4395063 PROC : db2sysc 1
INSTANCE: db2inst1        NODE : 001
EDUID    : 258             EDUNAME: db2sysc 1
FUNCTION: DB2 UDB, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
xport_send: dat_ep_post_rdma_write of the MCB failed: 0x70000

2009-04-27-15.42.04.329724-240 I9505628A1358      LEVEL: Severe
PID      : 1249362          TID : 258          KTID : 4395063 PROC : db2sysc 1
INSTANCE: db2inst1        NODE : 001
EDUID    : 258             EDUNAME: db2sysc 1
FUNCTION: DB2 UDB, Shared Data Structure Abstraction Layer ,
          SQLE_CA_CONN_ENTRY_DATA::sqlCaCeConnect, probe:622
MESSAGE  : CA RC= 2148073485
DATA #1  : String, 17 bytes
CAConnect failed.
DATA #2  : CAToken_t, PD_TYPE_SD_CATOKEN, 8 bytes
0x07000000003EE0B8 : 0000 0001 1064 AFD0          .....d..
DATA #3  : CA Retry Position, PD_TYPE_SAL_CA_RETRY, 8 bytes
894
CALLSTCK:
[0] 0x0900000012FF274C sqlCaCeConnect__23SQLE_CA_CONN_ENTRY_DATAFCP7CATokenC1
    + 0x40C
[1] 0x0900000012FF2CF8 sqlSingleCaCreateAndAddNewConnectionsToPool__
    21SQLE_SINGLE_CA_HANDLEFCUIT1Cb + 0x278
[2] 0x0900000012FF9188 sqlSingleCaInitialize__21SQLE_SINGLE_CA_HANDLEFCRC27SQLE_
    CA_CONN_POOL_NODE_INFOCU1P13SQLO_MEM_POOL + 0x448
[3] 0x0900000013001C50 sqlCaCpAddCa__17SQLE_CA_CONN_POOLFSCPU1 + 0x350
[4] 0x000000001000118AC sqlInitSysCtlr__FPiT1 + 0x140C
[5] 0x00000000100013008 sqlSysCtlr__Fv + 0x4A8
[6] 0x0900000012E15C78 sqloSystemControllerMain__FCUiPFv_iPFi_vPPvCPi + 0xD58
[7] 0x0900000012E177AC sqloRunInstance + 0x20C
[8] 0x00000000100006ECC DB2main + 0xAEC
[9] 0x0900000012C99048 sqloEDUMainEntry__FPcUi + 0xA8

```

これらのメッセージは、CF とメンバーの間の通信エラーを示しています。以下の手順に従ってください。

1. pdLogCfPrintf メッセージを見つけて、メッセージ・ストリング CF RC= を検索します。例えば CF RC= 2148073491 などです。
2. このストリングに隣接する数値 (この例では 2148073491) に注目してください。この値は、ネットワークまたは通信層からの理由コードを表しています。
3. このエラーの詳細情報をさらに得るには **db2diag** ツールを実行します。その際、**-cfr** パラメーターの後に値を指定します。例えば db2diag -cfr 2148073491 となります。
4. システムで uDAPL および InfiniBand が最近有効になった場合は、uDAPL 構成を検査します。
5. 前にリストしたエラーを示している各メンバー・ホストから CF IB ホスト名に向けて、および CF ホストからそれらのメンバーの IB ホスト名に向けて、IB ホスト名を ping します。
6. IB ホスト名の ping が失敗した場合、ポートが稼働状態であることを確認してください。ポートが稼働状態かどうか検査するには **ibstat -v** を実行します。以下の例では、Physical Port Physical State (物理ポート物理状態) の値が Link Up で、Logical Port State (論理ポート状態) の値が Active、さらに Physical Port State (物理ポート状態) の値が Active となっているため、リンクは正常と考えられます。

```
$ ibstat -v
```

```
-----  
IB NODE INFORMATION (iba0)  
-----  
Number of Ports: 2  
Globally Unique ID (GUID): 00.02.55.00.02.38.59.00  
Maximum Number of Queue Pairs: 16367  
Maximum Outstanding Work Requests: 32768  
Maximum Scatter Gather per WQE: 252  
Maximum Number of Completion Queues: 16380  
Maximum Multicast Groups: 32  
Maximum Memory Regions: 61382  
Maximum Memory Windows: 61382  
Hw Version info: 0x1000002  
Number of Reliable Datagram Domains: 0  
Total QPs in use: 3  
Total CQs in use: 4  
Total EQs in use: 1  
Total Memory Regions in use: 7  
Total MultiCast Groups in use: 2  
Total QPs in MCast Groups in use: 2  
EQ Event Bus ID: 0x90000300  
EQ Event ISN: 0x1004  
NEQ Event Bus ID: 0x90000300  
NEQ Event ISN: 0x90101  
-----
```

```
-----  
IB PORT 1 INFORMATION (iba0)  
-----  
Global ID Prefix: fe.80.00.00.00.00.00.00  
Local ID (LID): 000e  
Local Mask Control (LMC): 0000  
Logical Port State: Active  
Physical Port State: Active  
Physical Port Physical State: Link Up  
Physical Port Speed: 2.5G  
Physical Port Width: 4X  
Maximum Transmission Unit Capacity: 2048  
Current Number of Partition Keys: 1  
Partition Key List:  
  P_Key[0]: ffff  
Current Number of GUID's: 1  
Globally Unique ID List:  
  GUID[0]: 00.02.55.00.02.38.59.00  
-----
```

7. Galaxy InfiniBand アダプター・カード、InfiniBand スイッチ、およびケーブル接続を検査して、物理サーバーの障害を調べます。
8. また、AIX システム・エラー・ログに関連メッセージが示されることもあります。 **errpt -a** コマンドを実行することによってエラー・ログを確認できます。
9. 以下の例に示すように、InfiniBand ネットワーク・インターフェース、ホスト・チャンネル・アダプター、および icm の値がすべて Available (使用可能) であることを確認してください。

```
$ lsdev -C | grep ib  
fcnet0    Defined    00-08-01 Fibre Channel Network Protocol Device  
fcnet1    Defined    00-09-01 Fibre Channel Network Protocol Device  
ib0       Available   IP over Infiniband Network Interface  
iba0      Available   InfiniBand host channel adapter  
icm       Available   Infiniband Communication Manager
```

- セットアップが正しく実行され、ハードウェアが正しく機能している場合には、3 つの値がすべて「Available」になるはずですが、

- ネットワーク・インターフェースが「Available」でない場合、デバイスの状態を手動で変更できます。デバイスの状態を手動で変更するには、以下を使用できます。

```
$ chdev -l ib0 -a state=up
ib0 changed
```
 - iba0 または icm が Available 状態でない場合、デバイスのエラーを検査します。デバイスのエラーを検査するには、root ユーザーとして

```
/usr/sbin/cfgmgr -vl iba0
```

または

```
/usr/sbin/cfgmgr -vl icm
```

を実行します。
 - iba0 が見つからない場合、または Defined 状態のままである場合には、Host Channel Adapter が HMC 上のホストに現在割り当てられていることを確認します。
10. エラー発生時に CF サーバー・ホスト上で cf-server プロセスが実行中であったことを確認します。その時点で CF ホストが非稼働状態だったか、初期化されていないか、または再始動中だった場合には (その時点で **db2instance -list** を実行すると、1 次 CF は PRIMARY、2 次は PEER)、詳細を調べるために `cfdump.out*`、`cfdiag*.log`、およびコア・ファイルを検査します。しかし、エラー発生時に CF サーバーが稼働中で初期化されていた場合には、uDAPL 通信の問題が存在する可能性があります。
 11. **db2start** コマンドまたは **CONNECT** ステートメントが発行された場合、副次作用としてこれらのエラーを発生させた別の障害があるかどうかを判別するには、688 ページの『インスタンスが開始しない』を参照してください。
 12. そうでない場合、障害発生シナリオのトレースがエラーの原因の判別にしばしば役立ちます。CF トレースが有効になっていた場合は、ダンプを出力してください。CF トレースをダンプするには `db2trc cf dump fileName` というコマンドを実行します (`fileName` パラメーターの値は自分で定義してください)。
 13. CF トレースが無効になっている場合に有効にするには、`db2trc cf on -m "*.CF.xport_udapl.*.*"` というコマンドを実行します。
 14. さらに、問題判別のために IBM サービスから AIX システム・トレースおよび AIX メモリー・トレースを追加的に要請されることもあります。
 15. `xport_udapl` に対する CF トレースおよび AIX システム・トレースが記録された場合には、この情報を集めてください。追加の診断ログを収集するには **db2support** コマンドを実行します。すべてのホストで root として `snap -Y` を実行し、IBM サービスに連絡して支援を受けてください。

uDAPL over InfiniBand 接続のトレースの実行

IBM サービスに問題を報告するとき、uDAPL 通信層およびその下にある通信層の詳細トレースを要請されることがあります。

AIX トレースを取得する一般的な方法として、以下の説明を参考にすることができます。

このタスクについて

AIX システム・トレース機能 (デフォルトでは OFF) を有効にすると、使用方法によってはシステムのパフォーマンスに影響を与える可能性があります。このため、IBM サービスから指示された場合に限り、トレース機能を使って接続の問題を診断

してください。トレース・モード、トレース・バッファ・サイズ、ファイル・サイズ、およびトレース・フックとして使用すべき最適値は、問題の種類によって異なります。IBM サービスが問題を分析して、これらの変数の推奨値をお知らせします。

手順

uDAPL over InfiniBand 接続をトレースするには、次のようにします。

1. 特定のトレース・フックでどんなイベントをトレースするかを適切に設定するために、コンポーネント・トレース・レベルを調整します。例えば、これ以降の AIX システム・トレースで InfiniBand 通信層に関する詳細レベルのイベントを含めるには、以下のコマンドを実行します。

```
ctctrl -t network_ib -r systracedetail
```

2. トレースをオンに設定します。トレースをオンにするには **trace** コマンドを次のように使用します。

```
trace -T bufferize -L filesize -a -j trace_hooks
```

説明:

- デフォルト・トレース・モードが使用されます。トレース・データを継続的に集めるために 2 つのグローバル・バッファが使用されます。1 つのバッファはログ・ファイルに書き込まれ、もう 1 つのバッファでデータが集められます。
- **-T *bufferize*** はトレース・バッファ・サイズを指定します
- **-L *filesize*** は出力トレース・ログ・ファイル・サイズを指定します
- **-a** はバックグラウンドでトレースを実行することを指定します
- **-j *trace_hooks*** はトレース対象となるイベントの種類を指定します。複数のイベントをコンマ区切りリストで指定できます。

注: トレース・モード、バッファ・スコープ、*bufferize*、*filesize*、および *trace_hooks* の値は、発生している問題に応じて異なります。推奨値については IBM サービスにお問い合わせください。

3. 問題を再現させます
4. トレースをオフに設定します。 **trcstop** コマンドを使用します。
5. トレース・バッファをファイルにダンプします。次のように **trcrpt** コマンドを使用します。

```
trcrpt > filename
```

ここで *filename* はトレース・バッファのダンプ先となるファイルを指定します。AIX トレース機能の詳細については、AIX インフォメーション・センター: トレース機能を参照してください。

また AIX では、いくつかのコンポーネントに関してメモリー内だけのトレースを行う機能もあります。これには、デフォルトでオンに設定されたトレースが含まれます。これらはパフォーマンスへの影響を最小限に抑えるよう選択されています。これらのトレースは、IBM サービス担当員が明示的に操作しない限り、ディスク・ログ・ファイルには書き込まれません。InfiniBand メモリー内トレースを詳細レベルに引き上げるには、以下のコマンドを使用します。

```
ctctrl -t network_ib -r memtracedetail
```

ファイル・システム・トレースの取得

IBM サービスに問題を報告するとき、GPFS に基づく DB2 クラスタ・ファイル・システムの詳細トレースを要請されることがあります。GPFS トレースを取得するには、以下の情報を使用します。

このタスクについて

なお、トレース機能を有効にすると (デフォルトでは OFF になっている)、システムのパフォーマンスに影響を与える場合があります。このため、IBM 技術サポートから指示された場合に限り、トレース機能を使用してください。トレース・バッファ・サイズ、ファイル・サイズ、およびトレース・レベルとして使用すべき最適値は、問題の種類によって異なります。IBM 技術サポートが問題を分析して、これらの変数の推奨値をお知らせします。

GPFS トレース・パラメーターを構成するには **mmchconfig** コマンドを使用できます。

構成変更は、クラスタ内のすべてのノードに自動的に適用されます。

手順

GPFS トレースを取得するには、root ユーザーとして以下の手順を実行してください。

1. すべてのノードに /tmp/mmfs ディレクトリが存在することを確認します。トレース・レポートと内部ダンプはこのディレクトリに書き込まれます。
2. 以下のコマンドを入力することにより、各 GPFS クラスタでのトレース・レベルを設定します。

```
mmchconfig trace = trace_hooks
```

ここで *trace_hooks* はトレース対象となるイベントの種類を指定します。

注: トレース・フックの値は、発生している問題に応じて異なります。推奨値については IBM 技術サポートにお問い合わせください。

3. 以下のコマンドを入力することにより、トレース・バッファおよびトレース・ファイルのサイズを増やします。

```
mmchconfig traceFileSize = filesize
```

ここで *filesize* は出力トレース・ログ・ファイル・サイズを指定します。

注: *filesize* の値は、発生している問題に応じて異なります。推奨値については IBM 技術サポートにお問い合わせください。

4. トレースを開始するには、問題の影響を受けているすべてのホストで **mmtrace** コマンドを実行します。
5. トレースが開始済みかどうかを確認するには、**ps -ef | grep trace** を実行します。コマンド /bin/trace が表示される可能性があります。
6. トレース対象となる問題を再現させます。

7. 各 GPFS ノードで `mmtrace` コマンドを入力することにより、トレース・バッファをダンプします。
8. 以下のコマンドを入力することにより、トレースをオフにします。

```
mmtrace stop
```

インストール後の問題に関するよくある質問

この「よくある質問」リストの答えは、IBM DB2 pureScale Feature のインストール後に発生する問題を解決するのに役立ちます。

メンバーまたはクラスター・キャッシング・ファシリティが STARTED、PEER、CATCHUP のいずれの状態でもない場合、どうすればよいでしょうか？

いずれかのメンバーまたはクラスター・キャッシング・ファシリティが STARTED、PEER、CATCHUP のどの状態でもない場合、以下の手順を実行してください。

1. `db2stop` コマンドを発行してインスタンスを停止します。
2. `db2start` コマンドを発行してインスタンスを再始動します。
3. 失敗した `db2start` コマンドからのメッセージを確認して、問題を解決します。
4. 問題が続く場合は、`db2diag` ログ・ファイルを調べると問題についての詳細情報が見つかる可能性があります。
5. ファイルを調べても推定される問題の原因が見つからない場合は、以下の手順を実行してください。
 - a. `db2iupdt -drop` コマンドを発行することにより、メンバーまたはクラスター・キャッシング・ファシリティをドロップします。
 - b. `db2iupdt -add` コマンドを発行することにより、メンバーまたはクラスター・キャッシング・ファシリティを再び追加します。

db2start コマンドが失敗して SQL1517N を受け取った場合、どうすればよいでしょうか？

このエラーは、`db2nodes.cfg` ファイルとクラスター・マネージャーの間に不整合があることを示しています。クラスター・マネージャー・リソース・モデルを修復する方法について、詳しくは『クラスター・マネージャー・リソース・モデルの修復』のトピックを参照してください。

データベースを作成できない場合、どうすればよいでしょうか？

データベース作成コマンドを使ってデータベースを作成しようとしたとき、SQL1032N エラー (START DATABASE MANAGER コマンドが発行されていません) を受け取ることがあります。DB2 メンバーとして機能しているホストに必ずログオンしてください。クラスター・キャッシング・ファシリティとして稼働しているホストでデータベースを作成することはできません。DB2 メンバーにログオンして、`CREATE DATABASE` コマンドを再び発行します。

db2iupdt -drop コマンドが失敗して RSCT ピア・ドメインを縮小できない場合、どうすればよいでしょうか？

失敗したホストを手操作で RSCT ピア・ドメインから除去するには、次のようにします。

1. ピア・ドメインにアタッチされているリソースが依然として存在するかどうかを検査します。そうするには、失敗したホストに root ユーザーとしてログオンして **lssam** コマンドを発行します。 **lssam** コマンドは、特定のホスト上のリソースだけではなく、ドメイン内のすべてのリソースを表示します。リストを調べて、ドロップしようとしているホストにリソースがアタッチされていないことを確認してください。
2. アタッチされているリソースが存在しない場合、インストール開始ホスト (IIH) から **db2cluster -cm -remove -host host_name** を入力します。
3. アタッチされているリソースが依然として存在する場合は、以下の手順を実行します。
 - a. IIH から **su - instance_owner** を入力することにより、インスタンス所有者に切り替えます。
 - b. **db2cluster -cm -delete -resources** を入力することにより、リソースを除去します。この手順によってクラスター内のすべてのリソースが削除されます。
 - c. 元どおり root に切り替えます。
 - d. **db2cluster -cm -remove -host host_name** を入力することにより、失敗したホストを RSCT ピア・ドメインから除去します。
 - e. 再びインスタンス所有者に切り替えます。
 - f. **db2cluster -cm -create -resources** を入力することにより、失敗しなかったホスト上でリソースを再作成します。
 - g. 元どおり root に切り替えます。

ホストまたはメンバーの問題

問題が見つかった場合、問題を診断し、次にトラブルシューティングを行うことが重要です。アラートやエラーを識別し、DB2 pureScale環境で発生する可能性のある症状の診断とトラブルシューティングを行う方法について詳しく示します。

DB2 pureScale環境では、発生する問題症状についての情報を得るために、メンバーまたはクラスター・キャッシング・ファシリティーが配置されている任意のホストから **db2instance -list** コマンドを実行できます。

3 つの別個の状態要素 (メンバー、クラスター・キャッシング・ファシリティー、およびホスト) でアラート、エラー、**WAITING_FOR_FAILBACK** などの症状が見られることがあります。

データベースを使用できない場合でも、ホスト上で DB2 pureScale・インスタンス・マネージャーが引き続き使用可能であれば、データの一部を引き続き報告することができます。現在データベース接続が存在しない場合、またはインスタンスが停止している場合でも **db2instance -list** コマンドを使用できるからです。

追加の基本的な診断手順は次のとおりです。

1. **db2cluster -cm/-cfs -list -alert** を実行してアラートを確認します
2. **db2diag** ログ・ファイルの中にデータ診断情報とエラー・メッセージがあるかどうか確認します
3. **DIAGPATH** の中に、生成された診断ファイルおよび **FODC** ディレクトリーがあるかどうかを確認します
4. **CF_DIAGPATH** の中に、クラスター・キャッシング・ファシリティーの診断データが生成されているかどうか確認します
5. **errpt -a** を実行してシステム・エラー・ログを表示します。障害が発生した時点で該当するログ項目を探します。

ホスト、メンバー、またはクラスター・キャッシング・ファシリティーの状態を検出する詳しい方法については、関連情報セクションにある以下の概念トピックへのリンクを参照してください

- DB2 pureScale環境におけるモニター
- DB2 pureScale インスタンスの状況モニタリング

db2instance -list コマンドからのインスタンス状況情報の要約

このトピックでは、**db2instance -list** コマンドによって戻される状態についての概要を示します。これを使用してトラブルシューティング情報を調べることができます。

db2instance -list コマンドからの出力に関する詳細情報を得るには、以下の表を使用します。

表 88. **db2instance -list** コマンド出力に基づく順列

メンバーの状態	メンバーのアラート	ホストの状態	ホストのアラート	詳細情報
WAITING_FOR_FAILBACK	YES	INACTIVE	YES	703 ページの『メンバーが WAITING_FOR_FAILBACK (アラートあり)、対応するホストは INACTIVE 状態 (アラートあり)』を参照してください。
WAITING_FOR_FAILBACK	NO	INACTIVE	YES	711 ページの『メンバーが WAITING_FOR_FAILBACK (アラートなし)、対応するホストは INACTIVE 状態 (アラートあり)』を参照してください。
WAITING_FOR_FAILBACK	NO	ACTIVE	YES	712 ページの『メンバー が WAITING_FOR FAILBACK (アラートなし)、対応するホストは ACTIVE 状態 (アラートあり)』を参照してください。
STARTED	YES	ACTIVE	NO	704 ページの『STARTED 状態のメンバーにアラートが表示され、すべてのホストは ACTIVE 状態 (アラートなし) になっている』を参照してください。

STARTED	NO	ACTIVE	NO	db2instance -list コマンドの出力で、1 次クラスター・キャッシング・ファシリティーの切り替え状態 (PRIMARY から PEER)、2 次クラスター・キャッシング・ファシリティーの切り替え状態 (PEER から PRIMARY) が示されます。723 ページの『クラスター・キャッシング・ファシリティーのホスト間で 1 次役割が移動する』を参照してください。
ERROR	YES	ACTIVE	NO	709 ページの『メンバーが ERROR 状態』を参照してください。

メンバーまたはホストのアラート

db2instance -list コマンドの出力にメンバーまたはホストのアラートが表示されている場合、インスタンスのアラートに対応する以下のトピックを使用すると問題の診断に役立ちます。

メンバーが **WAITING_FOR_FAILBACK** (アラートあり)、対応するホストは **INACTIVE** 状態 (アラートあり):

db2instance -list コマンドの出力には、少なくとも 1 つのメンバーが **WAITING_FOR_FAILBACK** 状態 (アラートあり)、対応する 1 つ以上のホストが **INACTIVE** 状態 (アラートあり) であることが示されます。

コマンド **db2instance -list** からの以下の出力例は、3 つのメンバーと 2 つのクラスター・キャッシング・ファシリティーを示しています。

```

ID      TYPE      STATE      HOME_HOST  CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
-----
0       MEMBER    WAITING_FOR_FAILBACK  hostA      hostC         YES    0                  1             hostC-ib0
1       MEMBER    STARTED    hostB      hostB         NO     0                  0             hostB-ib0
2       MEMBER    STARTED    hostC      hostC         NO     0                  0             hostC-ib0
128    CF        PRIMARY    hostD      hostD         NO     -                  0             hostD-ib0
129    CF        PEER      hostE      hostE         NO     -                  0             hostE-ib0

HOSTNAME  STATE      INSTANCE_STOPPED  ALERT
-----
hostA     INACTIVE  NO                YES
hostB     ACTIVE    NO                NO
hostC     ACTIVE    NO                NO
hostD     ACTIVE    NO                NO
hostE     ACTIVE    NO                NO

```

メンバー 0 のホーム・ホスト hostA で問題が発生し、hostC 上で light モードで稼働しています。

メンバー 0 は ALERT が発生したことを示しています。何に関する ALERT であるかを確認するには **db2cluster -cm -list -alert** を実行します。

hostA は使用不可であるため、このホストの状態が **INACTIVE** に設定されています。hostA が **INACTIVE** 状態である間は、メンバー 0 はフェイルバックできません。メンバー 0 は **WAITING_FOR_FAILBACK** 状態のままになります。hostB での **restart light** が失敗したため、問題の調査を管理者に促すためにアラートが表示されています。

hostA が再び使用可能になった場合、その状態は **INACTIVE** から **ACTIVE** に変わります。メンバー 0 は hostA にフェイルバックし、その状態が **WAITING_FOR_FAILBACK** から **STARTED** に変わります。これについては、704 ページの『**STARTED** 状態のメンバーにアラートが表示され、すべてのホストは **ACTIVE** 状態 (アラートなし) になっている』で詳しく説明されています。

この症状の診断についての情報は、『hostA で障害が発生、restart light が最初に hostB で失敗、hostC で成功』のリンクを参照してください。

STARTED 状態のメンバーにアラートが表示され、すべてのホストは ACTIVE 状態 (アラートなし) になっている:

観察されるこの症状は、『メンバーが WAITING_FOR_FAILBACK (アラートあり)、対応するホストは INACTIVE 状態 (アラートあり)』のシナリオに続いて発生します。db2instance -list コマンドの出力には、STARTED 状態のメンバーにアラートが存在することが示されます。すべてのホストは ACTIVE 状態 (アラートなし) になっています。

以下は、db2instance -list コマンドからの出力のサンプルで、3 つのメンバーと 2 つのクラスター・キャッシング・ファシリティ環境が示されています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	STARTED	hostA	hostA	YES	0	0	hostA-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	ACTIVE	NO	NO
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

メンバー 0 には引き続きアラートのフラグが立てられています。これは、メンバーでの restart light 実行を妨げるような問題が hostB で発生したことを調査するよう管理者に促す標識です。

注: hostB の問題は、上位トピック『メンバーが WAITING_FOR_FAILBACK (アラートあり)、対応するホストは INACTIVE 状態 (アラートあり)』で詳しく説明されています。

診断についての情報は、『ホスト (A) で障害が発生、restart light がホスト (C) で成功、ただし最初のホスト (B) で失敗、フェイルバック非該当』の関連リンクを参照してください。

restart light が 1 つのホストで失敗し、別のホストで完了する:

restart light の際には、リカバリー・プロセスを完了させるためにメンバーがゲスト・ホストにフェイルオーバーします。このトピックの情報を使用すると、最初のゲスト・ホストで restart light が失敗し、2 番目のゲスト・ホストで正常に完了した理由を診断するのに役立ちます。

症状

コマンド db2instance -list からの以下の出力例は、3 つのメンバーと 2 つのクラスター・キャッシング・ファシリティがある環境を示しています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	WAITING_FOR_FAILBACK	hostA	hostC	YES	0	1	hostC-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	INACTIVE	NO	YES

```

hostB    ACTIVE    NO          NO
hostC    ACTIVE    NO          NO
hostD    ACTIVE    NO          NO
hostE    ACTIVE    NO          NO

```

メンバー 0 のホーム・ホスト hostA に問題が発生し、hostB での restart light が試行されました。しかし hostB での restart light は失敗しました。次に、このメンバーは hostC での restart light を試行し、それが成功しました。

hostA が再び使用可能になった場合、その状態は INACTIVE から ACTIVE に変わります。メンバー 0 は hostA にフェイルバックし、メンバーの状態が WAITING_FOR_FAILBACK から STARTED に変わります。

```

ID      TYPE      STATE      HOME_HOST  CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
---      -
0      MEMBER    STARTED    hostA      hostA         YES    0                  0              hostA-ib0
1      MEMBER    STARTED    hostB      hostB         NO     0                  0              hostB-ib0
2      MEMBER    STARTED    hostC      hostC         NO     0                  0              hostC-ib0
128    CF        PRIMARY    hostD      hostD         NO     -                  0              hostD-ib0
129    CF        PEER      hostE      hostE         NO     -                  0              hostE-ib0

```

```

HOSTNAME  STATE  INSTANCE_STOPPED  ALERT
-----
hostA     ACTIVE NO                 NO
hostB     ACTIVE NO                 NO
hostC     ACTIVE NO                 NO
hostD     ACTIVE NO                 NO
hostE     ACTIVE NO                 NO

```

トラブルシューティングの手順

hostB での restart light 失敗のトラブルシューティングを行うには、以下のいずれか 1 つ、または両方の手順を実行してください。

- db2diag ログ・ファイルで失敗に関する情報を確認した後、それについて調査します。

以下の出力例は、hostB での restart light 試行を示しています。

```

2009-08-27-23.37.52.416270-240 I6733A457          LEVEL: Event
PID      : 1093874          TID : 1          KTID : 2461779
PROC     : db2star2
INSTANCE:          NODE : 000
HOSTNAME: hostB
EDUID   : 1
FUNCTION: DB2 UDB, base sys utilities, DB2StartMain, probe:3368
MESSAGE : Idle process taken over by member
DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
0

```

diag メッセージを調べて、hostB での restart light 失敗に対応するエラーを分析します。

- hostA でのホスト障害を診断する手順については、727 ページの『restart light を伴うホスト・リブートの診断』を参照してください。
- このシナリオのトラブルシューティングの方法を示す例については、『restart light 中に発生したクラスター・ファイル・システム障害の診断』を参照してください。
- 問題を診断した後、メンバーに関するアラートをクリアします。

restart light 中に発生したクラスター・ファイル・システム障害の診断:

メンバーが restart light の実行を試みますが、クラスター・ファイル・システムのエラーが発生して、それが原因で restart light が失敗します。

症状

このトピックの目的は、障害の原因を診断することです。以下は、**db2instance -list** コマンドからの出力のサンプルで、3 つのメンバーと 2 つのクラスター・キャッシング・ファシリティ環境が示されています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	RESTARTING	hostA	hostB	No	0	1	hostB-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	INACTIVE	NO	YES
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

hostB で発生するメンバー 0 の RESTARTING 状態が存在します。メンバー 0 のホーム・ホストは hostA です。この出力は、メンバー 0 が hostB で restart light をアクティブに実行していることを示しています。

HostA は INACTIVE 状態で、それに対応するアラートがあります。これは、ホストの異常シャットダウン (電源障害、ネットワーク通信障害によるホスト・アクセス失敗など) を示しています。

その後の **db2instance -list** 出力は、hostB での restart light が失敗したことを示しています。次にメンバーは hostC での restart light を試行し、それが成功しました。hostA がオフラインのままであるため、メンバー 0 の状態は引き続き WAITING_FOR_FAILBACK です。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	WAITING_FOR_FAILBACK	hostA	hostC	YES	0	1	hostC-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	INACTIVE	NO	YES
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

hostA がオンラインになった場合、ホストの状態は ACTIVE に戻り、メンバー 0 は hostA にフェイルバックします。以下の **db2instance -list** 出力はそれを示しています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	STARTED	hostA	hostA	YES	0	0	hostA-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	ACTIVE	NO	NO
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

診断と解決

restart light のエラーを診断するには、以下のいずれか 1 つまたは複数の手順を実行してください。

- \$INSTHOME/sql1lib/db2dump/db2diag.log ファイルを確認します。以下の出力に示されるように、hostB での restart light 試行の時刻に対応するメッセージが見つかります。

```
2009-08-27-23.37.52.416270-240 I6733A457          LEVEL: Event
PID      : 1093874          TID : 1          KTID : 2461779
PROC     : db2star2
INSTANCE:                  NODE : 000
HOSTNAME: hostB
EDUID    : 1
FUNCTION: DB2 UDB, base sys utilities, DB2StartMain, probe:3368
MESSAGE  : Idle process taken over by member
DATA #1  : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2  : Database Partition Number, PD_TYPE_NODE, 2 bytes
0
```

- db2diag ログ・ファイルを調べて、hostB での restart light 失敗に対応するエラー・メッセージを分析します。restart light の試行時に以下の出力が生成されます。

```
2009-08-27-23.37.53.260582-240 E596165A1624      LEVEL: Error (OS)
PID      : 667736          TID : 1          PROC : db2rocm
INSTANCE:                  NODE : 000
HOSTNAME: hostB
EDUID    : 1
FUNCTION: DB2 UDB, oper system services, sqloflock, probe:100
MESSAGE  : ZRC=0x83000037=-2097151945
CALLED  : OS, -, fcntl
OSERR    : EINPROGRESS (55) "Operation now in progress"
DATA #1  : File handle, PD_TYPE_SQO_FILE_HDL, 8 bytes
File Handle          = 3
File System Block Size = 32768 bytes
File System Type    = mmfs
File Handle Flags :
  Require Sector Align = No
  DIO/CIO Mode         = No
  Raw Block Device     = No
  Reserved Handle      = No
  Flush On Close       = No
  Thread-Level Lock    = No
  Write-through Mode   = No
  File Not Tracked     = No
```

出力に示されているとおり、オペレーティング・システム関数 fcntl() が呼び出されて、エラー・メッセージおよびコード EINPROGRESS (55) "Operation now in progress" を生成しました。「File System Type」は mmfs、つまり General Parallel File System (GPFS) です。

- AIX errpt システム・ログの内容を表示するためにオペレーティング・システム・コマンド **errpt -a** を実行します。このシナリオでは、以下の出力例に示されるように、前述の Restart Light メッセージとほぼ同じ時刻に生成された MMFS_* メッセージ (例えば MMFS_MOREINFO) が hostB の AIX errpt ログに含まれています。これらの MMFS* メッセージは、問題の発生源が GPFS サブシステムであることを示しています。

```
LABEL:          MMFS_MOREINFO
IDENTIFIER:     E7AA68A1

Date/Time:      Thu Aug 27 23:37:53 EDT 2009
Sequence Number: 356562
Machine Id:     0006DA8AD700
Node Id:        hostB
Class:          S
```

```
Type:          PERM
WPAR:         Global
Resource Name: mmfs
```

```
Description
SOFTWARE PROGRAM ERROR
```

restart light を開始させたホスト障害の原因を診断するには、727 ページの『restart light を伴うホスト・リブートの診断』のトピックを参照してください。

ホストおよび restart light の問題についての役立つ追加情報が IBM 技術サポート Web サイト (DB2 for Linux UNIX and Windows サポート) にあります。ここでは症状やエラー・ログ・メッセージに基づいて既知の問題に関する情報を得ることができます。さらに、診断データを IBM 技術サポートで分析できるようにするには、クラスタの各ノードに対して **db2support output directory -d database name -s** を実行することにより、**db2support** パッケージを取得してください。IBM 技術サポートにデータを提出するには、Submitting diagnostic information to IBM Technical Support for problem determination を参照してください。

メンバーが **WAITING_FOR_FAILBACK** (アラートあり)、対応するホストは **ACTIVE** 状態 (アラートなし):

db2instance -list コマンドの出力に、少なくとも 1 つのメンバーが **WAITING_FOR_FAILBACK** 状態 (アラートあり) であり、対応する 1 つ以上のホストが **ACTIVE** 状態 (アラートなし) であることが示されます。

コマンド **db2instance -list** からの以下の出力例は、3 つのメンバーと 2 つのクラスタ・キャッシング・ファシリティがあることを示しています。

```
ID      TYPE      STATE      HOME_HOST      CURRENT_HOST      ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
----      ---      -
0      MEMBER    STOPPED    hostA          hostA             NO     0                 0             -
1      MEMBER    WAITING_FOR_FAILBACK  hostB          hostC             YES     0                 1             -
2      MEMBER    STARTED    hostC          hostC             NO     0                 0             -
128    CF        CATCHUP    hostB          hostB             NO     -                 0             -
129    CF        PRIMARY    hostC          hostC             NO     -                 0             -

HOSTNAME      STATE      INSTANCE_STOPPED  ALERT
-----      -
hostC         ACTIVE    NO                NO
hostB         ACTIVE    NO                NO
hostA         ACTIVE    NO                NO
```

メンバー 1 のホーム・ホスト hostB で問題が発生し、hostC 上で light モードで稼働しています。

メンバー 1 は **ALERT** が発生したことを示しています。**db2cluster -cm -list -alert** を実行すると、以下の出力例のような、アラートに関する情報が得られません。

1.

Alert: DB2 member '1' is currently awaiting failback to its home host 'coral18'. The cluster manager has restarted the DB2 member in restart light mode on another host. The home host for the member is available however automatic failback to home has been disabled.

Action: In order to move the member back its home host, the alert must be manually cleared with the command: 'db2cluster -cm -clear -alert'.

Impact: DB2 member '1' will not be able to service requests until this alert has been cleared and the DB2 member returns to its home host.

hostB が使用可能なとき、このホストの状態は ACTIVE に設定されています。自動フェイルバックが使用不可になっているため、メンバー 1 は hostB にフェイルバックできません。管理者がアラートをクリアして、次のコマンドを実行して自動フェイルバックを手動で使用可能にするまで、メンバー 1 は WAITING_FOR_FAILBACK 状態のままになります。

```
db2cluster -cm -option autofailback -value on
```

前述のコマンドの設定を有効にするには、インスタンスを再始動する必要があります。

メンバーのアラート状態

db2instance -list コマンドの出力には、メンバーが ALERT 状態で、ホストにアラートが存在することが示されます。

メンバーが ERROR 状態:

db2instance -list コマンドの出力に ERROR 状態のメンバーが含まれるシナリオを識別および診断する手法について、要約を示します。

以下は、**db2instance -list** コマンドからの出力のサンプルで、3 つのメンバーと 2 つのクラスター・キャッシング・ファシリティ環境が示されています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	ERROR	hostA	hostA	YES	0	0	hostA-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	ACTIVE	NO	NO
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

この出力に示されているとおり、DB2 クラスター・サービスは、障害の発生したメンバーをホーム・ホストまたはクラスター内の他のホストで再始動できませんでした。STATE 列に示されているとおり、メンバー 0 の状態は ERROR です。メンバー 0 に関するアラート・フラグが示すように、手操作による介入が必要です。

この症状の診断についての情報は、『ホストに障害が発生し、restart light がどのホストでも機能せず、フェイルバックが発生しない』を参照してください。

ホストに障害が発生し、restart light がどのホストでも機能しない:

メンバーおよびホストに障害が発生します。restart light がどのホストでも成功せず、フェイルバックが発生しません。

メンバーに障害が発生した理由、および restart light が失敗した理由を識別します。

- ホーム・ホストで、db2diag ログ・ファイルおよび errpt ファイルの FODC ディレクトリーを確認します。
- 以下のように **db2cluster** コマンドを発行して、クラスター・キャッシング・ファシリティの状態を検査します。

```
db2cluster -cm -list -host -state
db2cluster -cfs -list -host -state
```

- 727 ページの『restart light を伴うホスト・リブートの診断』を参照してください。

アイドル・リソースでの「オフラインに失敗」状態のクリア:

db2cluster -cm -clear -alert コマンドを実行してもアイドル・リソースでの「オフラインに失敗」状態をクリアできない場合には、このトピックの情報を使用して、「オフラインに失敗」状態をクリアしてください。

インスタンス・ユーザーまたは DB2クラスター・サービス管理者でなければなりません。

症状

アイドル・リソースでの「オフラインに失敗」状態は、**db2cluster -cm -clear -alert** コマンドを実行してもクリアできない場合があります。

問題の解決

1. **lssam** コマンドを発行し、「オフラインに失敗」状態になっているアイドル・リソースを検出します。クラスターに「オフラインに失敗」状態のアイドル・リソースがある場合は、以下と同じような出力が表示されます。

```
Failed offline IBM.ResourceGroup:idle_rmsinst_999_pub21-rg Nominal=Offline
'- Failed offline IBM.Application:idle_rmsinst_999_pub21-rs
'- Failed offline IBM.Application:idle_rmsinst_999_pub21-rs:pub21
```

この例では、「オフラインに失敗」状態のアイドル・リソースのインスタンス名は **rmsinst**、アイドル番号は **999**、ホスト名は **pub21** です。

2. **db2diag.log** ファイルの内容を調べて、以下と同じような項目を見つけます。このログ・ファイルは **DIAGPATH** 構成パラメーターによって指定されたディレクトリにあります。

```
2011-09-14-02.36.43.179141+540 I19433062A1157      LEVEL: Error
PID      : 12582956          TID   : 1          PROC  :db2rocm 999[rmsinst]
INSTANCE: rmsinst          NODE  : 999
HOSTNAME: pub21
EDUID   : 1                EDUNAME: db2rocm 999 [rmsinst]
FUNCTION: DB2 UDB,high avail services,sqlhaMonitorPartitionProcess,probe:647
MESSAGE  : ZRC=0x827301BF=-2106392129=HA_ZRC_MONITOR_FAULT_FORCED
          "The monitor fault is being forced."
DATA #1 : String, 54 bytes
Monitor override is detected, returning failed offline
DATA #2 : PGRP File Contents, PD_TYPE_SQLO_PGRP_FILE_CONTENTS, 2512 bytes
pgrpFile->iPgrpFileVersion : 2225
pgrpFile->iPgrpID : NOT_INITIALIZED
pgrpFile->iWdogPgrpId : NOT_INITIALIZED
pgrpFile->iSubPgrpId : NOT_INITIALIZED
pgrpFile->iIndex : NOT_INITIALIZED
pgrpFile->iNumber : NOT_INITIALIZED
pgrpFile->iMonitorOverride : 2
pgrpFile->crashCounter : 0
pgrpFile->firstCrashTime : 1970-01-01 09:00:00.000000
pgrpFile->monitorTimeoutCounter : 0
pgrpFile->firstMonitorTimeoutTime : 1970-01-01 09:00:00.000000
pgrpFile->lastMonitorTimeoutTime: 1970-01-01 09:00:00.000000
pgrpFile->hostname : pub21
pgrpFile->iNumHCAs : 0
```

- 以下のとおりになっていることを確認します。
 - db2diag.log ファイルのタイム・スタンプが現在時刻になっている。
 - lssam コマンド出力内のアイドル番号が、db2diag.log ファイル内の NODE 番号と一致している。
 - lssam コマンド出力内のホスト名が、db2diag.log ファイル内のホスト名と一致している。
- アラートをクリアするために、rm コマンドを使用して以下のファイルを同時に削除します。

```
rm sqllib/ctrlha/.idle.idle_number.hostname.pgrp sqllib/ctrlhamirror/.idle.idle_number.hostname.pgrp.mirror
```

ここで、idle_number はアイドル・リソースの番号、hostname は現在のホストの名前です。

この例では、次のコマンドを発行してアラートをクリアする必要があります。

```
rm sqllib/ctrlha/.idle.999.pub21.pgrp sqllib/ctrlhamirror/.idle.999.pub21.pgrp.mirror
```

注: アイドル・リソースが自動的に再始動するまで、最大で 10 分待機してください。アイドル・リソースが「オフラインに失敗」状態のままになっている場合は、IBM サービスにお問い合わせください。

ホストのアラート状態

db2instance -list の出力には、少なくとも 1 つのホストが ALERT 状態であることが示されていますが、ALERT 状態のメンバーはありません。

メンバーが WAITING_FOR_FAILBACK (アラートなし)、対応するホストは INACTIVE 状態 (アラートあり):

db2instance -list コマンドの出力には、少なくとも 1 つのメンバーが WAITING_FOR_FAILBACK 状態 (アラートなし) で、対応するホストが INACTIVE 状態 (アラートあり) であることが示されます。

以下は、db2instance -list コマンドからの出力のサンプルで、3 つのメンバーと 2 つの クラスター・キャッシング・ファシリティ環境が示されています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	WAITING_FOR_FAILBACK	hostA	hostB	NO	0	1	hostB-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	INACTIVE	NO	YES
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

メンバー 0 のホーム・ホスト hostA で問題が発生し、hostB での restart light が正常に実行されました。

この時点で **db2instance -list** を実行すると、restart light 実行後のメンバー 0 に関してはアラートが表示されません。これは、メンバー 0 の ALERT 列で示されています。つまり、メンバー 0 が hostB で restart light を実行する最初の試みが成功しました。

hostA は使用不可であるため、このホストの状態が INACTIVE に設定されています。hostA が INACTIVE 状態である間は、メンバー 0 はフェイルバックできません。このためメンバー 0 は WAITING_FOR_FAILBACK 状態のままになります。

hostA が再び使用可能になった場合、その状態は INACTIVE から ACTIVE に変わります。メンバー 0 は hostA にフェイルバックし、その状態が WAITING_FOR_FAILBACK から STARTED に変わります。この症状を診断するには、restart light を伴うホスト・リブートの診断を参照してください。

メンバー が WAITING_FOR FAILBACK (アラートなし)、対応するホストは ACTIVE 状態 (アラートあり):

db2instance -list コマンドの出力には、少なくとも 1 つのメンバーが WAITING_FOR_FAILBACK 状態 (アラートなし) で、対応するホストが ACTIVE 状態 (アラートあり) であることが示されます。

以下は、**db2instance -list** コマンドからの出力のサンプルで、3 つのメンバーと 2 つの クラスター・キャッシング・ファシリティ環境が示されています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	WAITING_FOR_FAILBACK	hostA	hostB	NO	0	1	hostB-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	ACTIVE	NO	YES
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

この **db2instance -list** 出力は、メンバー 0 のホーム・ホスト hostA で問題が発生し、hostB での restart light が正常に実行されたことを示しています。

この時点で **db2instance -list** を実行すると、restart light 実行後のメンバー 0 に関してはアラートが表示されません。これは、メンバーの ALERT 列で示されています。つまり、メンバー 0 が hostB で実行した最初の restart light 試行が成功しました。

hostA に関するアラートのフラグが立てられています。これは、hostA の問題を調査するよう管理者に促す標識です。hostA は ACTIVE ですが、クラスターでの使用を妨げている問題が存在する可能性があります (例えばファイル・システムやネットワークの問題)。

この症状の診断に関する情報は、関連リンク『ホストで障害が発生、restart light が成功、フェイルバックが失敗またはホーム・ホストで再始動できない』を参照してください。

restart light が成功した後、ホーム・ホストでメンバーを再始動できない:

ホストに障害が発生した後、restart light モードでメンバーがゲスト・ホストに正常にフェイルオーバーしますが、その後、フェイルバックできません。ホーム・ホスト上でメンバーを手動で再始動できません。

フェイルバックが失敗する原因となる状況

ホストが依然としてダウンしている場合

ハードウェアの障害または電源切れを検査してください。

/var が満杯である場合

DB2 は現在のホスト上で始動できず、別のホストで restart light モードでの再始動を試みます。

注: この状況では、/var がまだ満杯である可能性があるため、フェイルバックが発生しないことがあります。十分なスペースを確保してください。少なくとも 3GB を推奨します。

- クラスタ・ファイル・システムのログ・ファイルは /var/adm/ras に保管されます。(コア・ダンプが有効になっている場合) クラスタ・マネージャに関連するコア・ファイルはすべて /var/ct/<domain>/run/mc/*/* に入ります。このパスを調べて、コア・ファイルがそこに存在するかどうか確認してください。
 - これらのパスにある古いファイルは、古いシステム・ログと共にクリーンアップされる可能性があります
 - /var ファイル・システムのディスク・スペースを増やして、少なくとも 3 GB の空き領域を確保してください
- GPFS を始動し、**db2cluster -cfs -start host <failedHost>** を実行します
- **db2cluster -cm -list -alert** を実行してアラートのリストを表示します
- **db2cluster -cm -clear -alert** を実行してアラートをクリアします
- 引き続き問題が発生する場合、**db2support <output directory> -d <database name> -s** を実行して、IBM 技術サポートに連絡してください。

ホストは稼働しているが、通信障害 (uDAPL を介した通信不能) のためにアタッチまたは再始動できない場合

以下の診断手順を実行することにより、フェイルバック失敗の理由が uDAPL の通信障害であるかどうかを判別します。

- 障害が発生したホストとクラスタ・キャッシング・ファシリティーのホスト間の ping を実行します。
- InfiniBand コンポーネントが使用可能な状態であることを検証するために、**lsdev -C | grep ib** を実行します。使用可能な状態であることが表示されるはずですが。
- InfiniBand 状態を検査するために **ibstat -v** を使用します。ポートがアクティブで、リンクが機能していることを確認してください。
- クラスタ・キャッシング・ファシリティーが始動しないことによる障害が存在するかどうか確認するために、**cfdump.out***、**core files**、**cfdiag*.log**、**mgmnt_lwd_log** を検査します。障害が存在する場合は

db2instance -list を実行します。その結果、*primary* が **STARTED** 以外の状態、*secondary* が **PEER** 以外の状態であることが示されます。

- 初期化情報やオブジェクト情報が *cfdump.out* に示されない場合、クラスター・キャッシング・ファシリティが正常に始動しなかったと考えられます。
- *cfdump.out* に情報が含まれる場合、いずれかの時点でクラスター・キャッシング・ファシリティが正常に始動していたことになります。
- 物理的な IB ネットワーク・ケーブル接続を確認します
 - 716 ページの『uDAPL 通信障害によるメンバーのタイムアウトおよび障害』を参照してください。
- 通信できない場合は、**db2support <output directory> -d <database name> -s** を実行して、IBM 技術サポートに連絡してください。

ホストは稼働しているが、ファイバー・チャネル・カードの問題、ディスク・エラー、SAN ケーブル接続の問題 (これが原因で GPFS ディスクへの接続がホスト上で失敗する) が **sys** ログに示されている場合

718 ページの『DB2 クラスター・ファイル・システムの障害の原因となるディスク障害』を参照してください。または、GPFS が再マウントされないかどうか確認してください。

- **db2cluster -cfs -list -host -state** を実行します。
- `mount | grep mmfs` で `filesystem type=mmfs` が結果に示されているかどうかを確認します。
- 接続、カード、ディスクを検査し、**db2cluster -cfs -start host <failedHost>** を使って GPFS を再始動します。

ホストに障害が発生する原因

ホストに障害が発生する原因を識別するには

- ハードウェアの障害または電源切れを検査してください。
- *hostA* でのホスト障害を診断する手順については、727 ページの『restart light を伴うホスト・リブートの診断』を参照してください。

ディスク使用量の増大 (または */var* ファイル・パスが満杯)、または **SQL30108N** エラー:

このシナリオでは、IBM Tivoli System Automation for Multiplatforms リカバリー・リソース・マネージャー・デーモンの障害について説明します。このデーモンは Tivoli SA MP の意思決定エンジンとして機能し、システムでは **IBM.RecoveryRM** として識別されます。このデーモンに障害が発生した場合の症状として、ディスク使用量の増大 (または */var* ファイル・パスが満杯)、あるいは **SQL30108N** があります。

このシナリオには 2 つのケースがあります。

ケース 1: ディスク使用量の増大 (または満杯の) '*/var*' ファイル・システム

このエラーの症状は次のシナリオのとおりです。

- 実行時に IBM Tivoli System Automation for Multiplatforms リカバリー・リソース・マネージャー・デーモンに障害が発生した場合、エンド・ユーザーやアプリケーションが影響を受けないよう、またデータベース処理が継続するように、Tivoli SA MP は自動的にデーモンを再始動しようとします。特に顕著な症状はありません。Tivoli SA MP は /var/ct/db2domain/log/mc/ (エラー・ログ)、/var/ct/db2domain/run/mc/ (コア・ダンプ)、および /tmp/db2_cluster_manager_spooling (デフォルト・トレース・ディレクトリー) の中に診断情報を書き込みます。診断データが蓄積しているかどうか、var/... ディレクトリーを検査してください。

診断および解決の詳しい手順は、次のとおりです。

- 上記のディレクトリーに書き込まれる診断データが蓄積し続けている場合、古い診断データをアーカイブ場所に保存するのが安全です。例えば、
 - mv /var/ct/db2domain/run/mc/IBM.RecoveryRM/trace.6.sp /archivepath (ここで /var/ct/db2domain/run/mc/IBM.RecoveryRM/trace.6.sp は Tivoli SA MP 診断宛先パス)

注: /archivepath は任意のファイル・システムです
- /var/ct/db2domain/log/mc/ および /var/ct/db2domain/run/mc/ ディレクトリーを定期的にモニターして、/var ファイル・システムの空き領域を少なくとも 3 GB に保つことが重要です。
- IBM サービス・チームおよび開発チームは、トラブルシューティングのためにトレース・ファイルとコア・ファイルを使用します。診断データを IBM 技術サポートで分析できるようにするには、クラスタの各メンバーに対して以下のコマンドを実行することにより、**db2support** パッケージを取得します。
 - **db2support** output_directory -d database_name -s
- 以下の説明に従って、IBM Technical Support にデータをアップロードしてください。
 - Submitting diagnostic information to IBM Technical Support for problem determination
- IBM 技術サポートの Web サイトには役立つ情報が提供されており、症状やエラー・ログ・メッセージに基づいて既知の問題を識別できます。
 - DB2 for Linux UNIX and Windows サポート

ケース 2: SQL30108N

エラーのためにデーモンの再始動が失敗した結果として、クラスタ内の別のホストに restart light が実行された場合に、このケースが発生します。

このエラーの症状は次のシナリオのとおりです。

- アプリケーションが SQL30108N エラー・メッセージを戻します。
- /var ファイル・システムの下で診断データが蓄積しているかどうか検査してください。Tivoli SA MP は /var/ct/db2domain/log/mc/ (エラー・ログ)、/var/ct/db2domain/run/mc/ (トレースおよびコア・ダンプ)、および /tmp/db2_cluster_manager_spooling (デフォルト・トレース・ディレクトリー) の中に診断情報を書き込みます。 .

診断および解決の詳しい手順は、次のとおりです。

- db2diag ログ・ファイルの中に次のようなメッセージがあるかどうか確認します。

- DATA #6 : String, 48 bytes
Line # : 6884---2610-403 The resource is stale.

または

- DATA #6 : String, 142 bytes
Line # : 9578---2610-422 Cannot execute the command on node coralxib38.torolab.ibm.com. The resource manager IBM.RecoveryRM is not available.

上記のエラーがある場合、これは Tivoli SA MP リカバリー・リソース・マネージャー・デーモンで問題が発生したことを示します。問題を診断するための診断データが Tivoli SA MP によって書き込まれます。

- /var/ct/db2domain/log/mc/ および /var/ct/db2domain/run/mc/ ディレクトリーに書き込まれる診断データが蓄積し続けている場合、古い診断データをアーカイブ場所に保存するのが安全です。例えば、

- mv /var/ct/db2domain/run/mc/IBM.RecoveryRM/trace.6.sp /archivepath (ここで /var/ct/db2domain/run/mc/IBM.RecoveryRM/trace.6.sp は Tivoli SA MP 診断宛先パス)

注: /archivepath は任意のアーカイブ・ファイル・システムです

- /var/ct/db2domain/log/mc/ および /var/ct/db2domain/run/mc/ ディレクトリーを定期的にモニターして、/var ファイル・システムの空き領域を少なくとも 3 GB に保つことが重要です。

- IBM サービス・チームおよび開発チームは、トラブルシューティングのためにトレース・ファイルとコア・ファイルを使用します。診断データを IBM 技術サポートで分析できるようにするには、クラスタの各ノードに対して以下のコマンドを実行することにより、**db2support** パッケージを取得します。

- **db2support** output_directory -d database_name -s

- 以下の説明に従って、IBM Technical Support にデータをアップロードしてください。

- Submitting diagnostic information to IBM Technical Support for problem determination

- IBM 技術サポートの Web サイトには役立つ情報が提供されており、症状やエラー・ログ・メッセージに基づいて既知の問題を識別できます。

- DB2 for Linux UNIX and Windows サポート

uDAPL 通信障害によるメンバーのタイムアウトおよび障害:

メンバーの散発的な障害と再始動を引き起こす uDAPL 通信タイムアウトの診断および解決。

症状

始動時、または日常の OLTP 処理において、メンバーの障害が散発的に発生します。その結果、メンバーがホーム・ホスト上で正常に再始動されるか、(ホーム・ホストで再始動できない場合には) 別のホストに restart light されます。情報スタッ

ク・トレースバック・ファイル、ダンプ・ファイル、および通常ダンプされるその他のデータが **diagpath** および **cf_diagpath** に書き込まれます。

症状として、このような診断データの書き込みによって db2dump ファイル・システム内のディスク・スペース使用のペースが速くなる可能性があります。

診断と解決

この問題を診断して解決するには、以下の手順を実行します。

- `~/sqlllib_shared/db2dump/ $m` を確認して、メンバーの **db2diag** ログ・ファイルが異常に大きくないか、または多くの診断ダンプ・ディレクトリーがその中に存在していないか確認します。 **db2diag** ログ・ファイルを調べると、関数 `pdLogCaPrintf` からの次のようなメッセージが含まれる可能性があります。

```
2009-01-23-17.33.23.976179-300 I5463632A503      LEVEL: Severe
PID      : 602310          TID   : 15422          PROC  : db2sysc 0
INSTANCE:                NODE  : 000           DB   :
APPHDL  : 0-53           APPID:
AUTHID  :
EDUID   : 15422          EDUNAME: db2agent (    ) 0
FUNCTION: DB2 UDB, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
xport_send: Timed-out waiting for completion of dat_ep_post_rdma_write of an MCB
```

```
2009-01-23-17.33.24.473642-300 I5464136A467      LEVEL: Severe
PID      : 602310          TID   : 15422          PROC  : db2sysc 0
INSTANCE:                NODE  : 000           DB   :
APPHDL  : 0-53           APPID:
AUTHID  :
EDUID   : 15422          EDUNAME: db2agent (    ) 0
FUNCTION: DB2 UDB, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
ClientXport.send (CARAR:) failed: 0x80090013
```

- 上記の診断メッセージの後に **Restart Light** メッセージがあるかどうか、**db2diag** ログ・ファイルを調べます。 **Restart Light** メッセージを含むさまざまな再始動メッセージについて、詳しくは、719 ページの『DB2 pureScale環境で発生する可能性のある再始動イベント』を参照してください。

```
2009-08-27-23.37.52.416270-240 I6733A457      LEVEL: Event
PID      : 1093874        TID   : 1             KTID  : 2461779
PROC    : db2star2
INSTANCE:                NODE  : 001
HOSTNAME: hostC
EDUID   : 1
FUNCTION: DB2 UDB, base sys utilities, DB2StartMain, probe:3368
MESSAGE : Idle process taken over by member
DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
1
```

- `pdLogCfPrintf` メッセージを見つけた後、診断メッセージ・ストリング `CF RC=` を検索します。例えば `CF RC= 2148073491` となります。
- このストリングに隣接する数値 (この例では `2148073491`) に注目してください。これは、ネットワークまたは通信層からの理由コードを表しています。
- このエラーの詳細情報をさらに得るには、**db2diag** ツールを使用します。例えば `db2diag -cfrc 2148073491` です。
- クラスタ・キャッシング・ファシリティがオンライン状態になっているかどうか確認するために `ping` します。 `ping` が成功した場合、各クラスタに対して

db2supportoutput_directory-ddatabase_name-s を実行して **db2support** パッケージを収集した後、IBM 技術サポートに連絡してください。

- このような問題を診断するために IBM サービスから uDAPL トレースを要請される場合があります。697 ページの『uDAPL over InfiniBand 接続のトレースの実行』を参照してください。

DB2 クラスタ・ファイル・システムの障害の原因となるディスク障害:

GPFS に基づく DB2 クラスタ・ファイル・システム の、1 つのメンバーでの障害。これによって、すべての共有ファイル・システム (db2dump を含む) がそのメンバーについてアンマウントされます。GPFS はダウンしたままです。

症状

注: db2dump ディレクトリは、GPFS ファイル・システム上に存在する場合のみ、失われます。別のもの (例えば **diagpath**) に設定されている場合は、この GPFS 障害によって影響を受けません。

以下は **db2instance -list** コマンドからの出力サンプルで、ホスト・アラートが出された 3 つのメンバーと 2 つのクラスタ・キャッシング・ファシリティ環境が示されています。

```
db2instance -list
ID  TYPE  STATE                HOME_HOST  CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
---  ---  -----                -          -             -      -                 -             -
0   MEMBER  WAITING_FOR_FAILBACK hostA      hostB         NO     0                 1             hostB-ib0
1   MEMBER  STARTED             hostB      hostB         NO     0                 0             hostB-ib0
2   MEMBER  STARTED             hostC      hostC         NO     0                 0             hostC-ib0
128 CF      PRIMARY            hostD      hostD         NO     -                 0             hostD-ib0
129 CF      PEER               hostE      hostE         NO     -                 0             hostE-ib0

HOSTNAME                STATE  INSTANCE_STOPPED  ALERT
-----                -
hostA                   ACTIVE  NO                 YES
hostB                   ACTIVE  NO                 NO
hostC                   ACTIVE  NO                 NO
hostD                   ACTIVE  NO                 NO
hostE                   ACTIVE  NO                 NO
```

診断と解決

GPFS エラー・メッセージを介してこのシナリオを識別できます。

- **db2cluster -cm -list -alert** を実行することにより、アラート・メッセージを確認します。例えば、

```
db2cluster -cm -list -alert
The host "hostA.torolab.ibm.com" is not able to access the following
file systems:
"/db2cfs/db2inst1/sql1lib/db2dump".
Check the disk connections and mount the file system.
See the DB2 Information Center for more details.
This alert must be cleared manually via the command
db2cluster -clear -alert -host hostA.torolab.ibm.com
While the file system is offline, the db2 メンバー on this
host will be in restart light mode on other systems and will
be WAITING_FOR_FAILBACK
```

- オペレーティング・システム・コマンド **ls** または **cd** を使用することにより、該当するファイル・システムにアクセスできることを確認します。

```
ls /db2cfs/db2inst1/sql1lib/db2dump
cd /db2cfs/db2inst1/sql1lib/db2dump
```

ファイル・システムがアクセス不能またはオフラインである場合、ディレクトリが存在しない (または使用不可である) ことを示すメッセージがコマンドから戻されます。

- **ls** または **cd** コマンドを実行した結果、ファイル・システムがアクセス不能である場合には、問題のホスト上にファイル・システムがマウント済みと見なされているかどうか確認します。

– この例のシナリオでは、hostA に対して以下を実行します。

```
mount |grep sqllib
```

マウントされていない場合、/db2cfs/db2inst1/sqllib ファイル・システムは結果セットに表示されません。

- 問題のファイル・システムをマウントするには、**db2cluster -cfs -mount -filesystem fs_name** というコマンドを実行します。

- **diagpath** の場所にある **db2diag** ログ・ファイルを確認します。障害についての詳細情報が必要な場合は、**restart light** の原因である問題を確認するために、該当するメッセージを検索します。再始動の時刻に対応する **db2diag** ログ・レコードが存在する可能性があります。例えば、

```
2009-08-27-23.37.52.416270-240 I6733A457          LEVEL: Event
PID      : 1093874          TID   : 1          KTID  : 2461779
PROC     : db2star2
INSTANCE:                  NODE   : 000
HOSTNAME: hostB
EDUID    : 1
FUNCTION: DB2 UDB, base sys utilities, DB2StartMain, probe:3368
MESSAGE  : Idle process taken over by member
DATA #1  : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2  : Database Partition Number, PD_TYPE_NODE, 2 bytes
0
```

Restart Light メッセージを含むさまざまな再始動メッセージについて、詳しくは、『DB2 pureScale環境で発生する可能性のある再始動イベント』を参照してください。

- さらに、問題を診断するためにシステム・ログを参照することもできます。AIX **errpt** システム・ログの内容を表示するにはオペレーティング・システム・コマンド **errpt -a** を実行します。この例のシナリオでは、上記の Restart Light メッセージの時刻の直前における hostA からの AIX **errpt** ログを見ると、MMFS_* というメッセージ (例えば **errpt** の中の MMFS_GENERIC、MMFS_PHOENIX) に次のようなテキストが表示されます。

```
message: "GPFS: 6027-752 Lost membership in cluster hostA. Unmounting file systems.
```

- ディスク、ディスク接続、およびファイバー・チャネル・カードを検査します。この例のシナリオでは、ホストと SAN の間の相互接続に障害があったことが根本原因です。
- 診断データを IBM 技術サポートで分析できるようにするために、クラスタの各メンバーに対して **db2support output_directory -d database_name** を実行することにより、**db2support** パッケージを取得します。Submitting diagnostic information to IBM Technical Support for problem determination の説明に従って、IBM 技術サポートにデータをアップロードしてください。

DB2 pureScale環境で発生する可能性のある再始動イベント:

DB2 pureScaleで発生する可能性のある再始動イベントの種類に関する情報です。
Restart Light 診断メッセージと、他の種類の再始動メッセージを混同しないでください。

DB2 pureScale環境で発生する可能性のある再始動イベントの種類は、次のとおりです。

1. メンバーのローカル再始動

```
2009-11-08-18.56.29.767906-300 I151976017E340 LEVEL: Event
PID : 26319 TID : 46989525792064 KTID : 26319
PROC : db2rocm 1
INSTANCE: inst1 NODE : 001
HOSTNAME: hostA
FUNCTION: DB2 UDB, high avail services, sqlhaStartPartition, probe:1636
DATA #1 : <preformatted>
Successful start of member
```

```
2009-11-08-18.56.29.769311-300 I151976358E373 LEVEL: Event
PID : 26319 TID : 46989525792064 KTID : 26319
PROC : db2rocm 1
INSTANCE: inst1 NODE : 001
HOSTNAME: hostA
FUNCTION: DB2 UDB, high avail services, db2rocm_main, probe:1381
DATA #1 : String, 30 bytes
db2rocm 1 DB2 inst1 1 START
DATA #2 : String, 7 bytes
SUCCESS
```

関数 sqlhaStartPartition からの「Successful start of member」というストリングに注目してください。これはローカル再始動が開始したことを示します。

2. メンバーの Restart Light

```
2009-08-27-23.37.52.416270-240 I6733A457 LEVEL: Event
PID : 1093874 TID : 1 KTID : 2461779
PROC : db2star2
INSTANCE: NODE : 001
HOSTNAME: hostC
EDUID : 1
FUNCTION: DB2 UDB, base sys utilities, DB2StartMain, probe:3368
MESSAGE : Idle process taken over by member
DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
1
```

「Idle process taken over by member」に注目してください。これは、Restart Light を実行するためにリカバリー・アイドル・プロセスがアクティブ化され、メンバー 1 が hostC にフェイルオーバーしたことを示しています。

3. メンバーのクラッシュ・リカバリー

```
2009-11-09-13.55.08.330120-300 I338293E831 LEVEL: Info
PID : 24616 TID : 47881260099904 KTID : 24731
PROC : db2sysc 0
INSTANCE: NODE : 000 DB :
APPHDL : 0-52 APPID: *N0.DB2.091109185417
EDUID : 24 EDUNAME: db2agent ( ) 0
FUNCTION: DB2 UDB, data protection services, sqlpgint, probe:430
DATA #1 : <preformatted>
Crash recovery decision:
Is this member consistent? No
Is the database marked consistent in the GLFH? No
Is the database restore pending? No
Is the database rollforward pending? No
```

Were the CF structures valid on startup? No
Are we performing an offline restore? No
Are we performing group crash recovery? No
Are we performing member crash recovery? Yes
Are we initializing the LFS for the group? No

.....

2009-11-09-13.55.09.002292-300 I50967394E495 LEVEL: Info
PID : 24616 TID : 47881260099904 KTID : 24731
PROC : db2sysc 0
INSTANCE: NODE : 000 DB :
APPHDL : 0-52 APPID: *N0.DB2.091109185417
AUTHID :
EDUID : 24 EDUNAME: db2agent () 0
FUNCTION: DB2 UDB, recovery manager, sqlpresr, probe:3170
DATA #1 : <preformatted>
Crash recovery completed. Next LSN is 00000000006DB3D1

「Are we performing member crash recovery? Yes」というストリングは、メンバーのクラッシュ・リカバリー・イベントが発生したことを示しています。

4. グループ再始動

5. グループ・クラッシュ・リカバリー

2009-11-09-22.24.41.330120-300 I338293E831 LEVEL: Info
PID : 10900 TID : 46949294139712 KTID : 18929
PROC : db2sysc 2
INSTANCE: NODE : 002 DB :
APPHDL : 2-52 APPID: *N2.DB2.091110032438
EDUID : 24 EDUNAME: db2agnti () 2
FUNCTION: DB2 UDB, data protection services, sqlpgint, probe:430
DATA #1 : <preformatted>
Crash recovery decision:
Is this member consistent? No
Is the database marked consistent in the GLFH? No
Is the database restore pending? No
Is the database rollforward pending? No
Were the CF structures valid on startup? No
Are we performing an offline restore? No
Are we performing group crash recovery? Yes
Are we performing member crash recovery? No
Are we initializing the LFS for the group? Yes

2009-11-09-22.24.41.540562-300 E365262E436 LEVEL: Info
PID : 10900 TID : 46949294139712 KTID : 18929
PROC : db2sysc 2
INSTANCE: NODE : 002 DB :
APPHDL : 2-52 APPID: *N2.DB2.091110032438
EDUID : 24 EDUNAME: db2agnti () 2
FUNCTION: DB2 UDB, base sys utilities, sqledint, probe:3559
MESSAGE : Crash Recovery is needed.

2009-11-09-22.24.53.177348-300 E431539E458 LEVEL: Info
PID : 10900 TID : 46949294139712 KTID : 18929
PROC : db2sysc 2
INSTANCE: NODE : 002 DB :
APPHDL : 2-52 APPID: *N2.DB2.091110032438
EDUID : 24 EDUNAME: db2agnti () 2
FUNCTION: DB2 UDB, recovery manager, sqlpresr, probe:210
MESSAGE : ADM1527I **Group crash recovery has been initiated.**

.....

2009-11-09-22.25.14.432113-300 E552559E467 LEVEL: Info
PID : 10900 TID : 46949294139712 KTID : 18929

```

PROC      : db2sysc 2
INSTANCE:                               NODE : 002          DB      :
APPHDL   : 2-52                          APPID: *N2.DB2.091110032438
EDUID    : 24                             EDUNAME: db2agnti (      ) 2
FUNCTION: DB2 UDB, recovery manager, sqlpresr, probe:3110
MESSAGE  : ADM1528I  Group crash recovery has completed successfully.

```

- メッセージ・ログの以下の行は、グループ・クラッシュ・リカバリー・イベントが発生したことを示しています。

```

Are we performing group crash recovery? Yes
Group crash recovery has been initiated.
Group crash recovery has completed successfully.

```

メンバーが **ACTIVE** だがホストに `/var` ファイル・システムに関連するアラートがある:

db2instance -list コマンドの出力には、すべてのメンバーが **ACTIVE** 状態であるが、少なくとも 1 つのホストに `/var` ファイル・システムに関連するアラートがあることが示されています。

以下は、**db2instance -list** コマンドからの出力のサンプルで、3 つのメンバーと 2 つの クラスター・キャッシング・ファシリティ環境が示されています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	STARTED	hostA	hostA	NO	0	0	-
1	MEMBER	STARTED	hostB	hostB	NO	0	0	-
2	MEMBER	STARTED	hostC	hostC	NO	0	0	-
128	CF	PRIMARY	hostD	hostD	NO	-	0	-
129	CF	PEER	hostE	hostE	NO	-	0	-

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	ACTIVE	NO	YES
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

この出力には、hostA にアラートが設定されていることが示されています。アラートについてさらに情報を収集するには、**db2cluster** コマンドに `-cm -list -alert` オプションを指定して使用します。 `/var` ファイル・システムのフリー・スペースがいずれかのホストで 25 MB 未満であった場合、そのホストについて次のアラート・メッセージが表示されます。

```

Alert: Host name hostA has 10 MB free space available on /var file system. Failure
to write to /var due to no
space will result in DB2 cluster failure.

```

```

Action: The file system requires a minimum of 25 MB free disk space is available.
Free up space on /var file system.
The alert will be cleared automatically when a
sufficient amount of space becomes available on the filesystem.

```

```

Impact: DB2 Cluster Services may not function properly on the specified host and
may eventually lead to a DB2 instance failure.

```

このアラートは情報提供のみを目的としていますが、この状態をモニターして影響を受けているホストのファイル・システムのスペースを増やす処置を行うことがきわめて重要です。

別の可能性として、DB2 がホストの `/var` ファイル・システムの使用量に関する情報を収集できないという状態です。この場合、そのホストについて次のアラート・メッセージが表示されます。

Alert: There was a failure to retrieve /var file system usage on host name db2dev07. Check the db2diag.log for messages concerning failures on host 'hostA' for /var file system.

Action: The failure needs to be resolved to clear the alert.

Impact: DB2 cluster will fail to write information about important errors and events. This will cause DB2 cluster failure.

ホスト・ファイル・システムの使用量がモニターされることが重要であるため、できるだけ早く処置する必要があります。

メンバーとホストにアラートが存在しない

db2instance -list コマンドの出力には、どのメンバーおよびホストにもアラートが存在しないことが示されていますが、他の問題が発生した可能性があります。

クラスター・キャッシング・ファシリティーのホスト間で 1 次役割が移動する:

db2instance -list コマンドを最後に実行したときと同じホストに 1 次役割がありません。つまり、別のクラスター・キャッシング・ファシリティー・ホストに 1 次役割が移動しています。このホスト変更は、過去のいずれかの時点でクラスター・キャッシング・ファシリティーの問題が発生したことを示しており、調査が必要になる場合があります。

以下は、**db2instance -list** コマンドからの出力のサンプルで、3 つのメンバーと 2 つのクラスター・キャッシング・ファシリティー環境が示されています。

ID	TYPE	STATE	HOME_HOST
0	MEMBER	STARTED	hostA
1	MEMBER	STARTED	hostB
2	MEMBER	STARTED	hostC
128	CF	PEER	hostD
129	CF	PRIMARY	hostE

CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
hostA	NO	0	0	hostA-ib0
hostB	NO	0	0	hostB-ib0
hostC	NO	0	0	hostC-ib0
hostD	NO	-	0	hostD-ib0
hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	ACTIVE	NO	NO
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

メンバーと同様に、それぞれのクラスター・キャッシング・ファシリティーは `cfdiag*.log` に情報を記録し、必要に応じてより詳しい診断データをダンプします。これらのファイルの格納場所は、データベース・マネージャー構成パラメーター **cf_diagpath** で設定されたディレクトリーです。設定されていない場合は **diagpath** の場所、または `$INSTHOME/sqlllib_shared/db2dump/ $m` (デフォルト) です。

- クラスタ・キャッシング・ファシリティーの診断ログ・ファイル
(*cfdiag-timestamp.cf_id.log*)
 - これらの各ファイルには、クラスタ・キャッシング・ファシリティーに関連するアクティビティーのログが保持されます。イベント、エラー、警告、または追加のデバッグ情報がこのログに記録されます。このログの外観は **db2diag** ログ・ファイルと同様です。クラスタ・キャッシング・ファシリティー が始動するたびに、新しいログが作成されます。
 - それぞれのクラスタ・キャッシング・ファシリティーについて常に最新の診断ロギング・ファイルを指し示し、*cfdiag.cf_id.log* という形式を持つ、単一の静的クラスタ・キャッシング・ファシリティー診断ログ名があることに注意してください。
- クラスタ・キャッシング・ファシリティーの出力ダンプ診断ファイル
(*cfdump.out.cf_pid.hostname.cf_id*)
 - これらのファイルには、クラスタ・キャッシング・ファシリティーの始動と停止に関する情報が含まれます。追加の出力が表示される場合があります。
- 管理 LWD 診断ログ・ファイル (*mgmnt_lwd_log.cf_pid*)
 - このログ・ファイルは、特定のクラスタ・キャッシング・ファシリティーの LightWeight Daemon (LWD) プロセスに関するログ項目を示します。このログ・ファイルに提示されるエラーは、LWD が正しく始動しなかったことを示しています。正常に始動した場合、ログには **ERROR** メッセージが含まれません。
- クラスタ・キャッシング・ファシリティーのスタック・ファイル
(*CAPD.cf_pid.tid.thrstk*)
 - これらは、クラスタ・キャッシング・ファシリティーがシグナルを検出したときに生成されるスタック・ファイルです。これらのファイルは、クラスタ・キャッシング・ファシリティーの問題を診断するうえで重要です。
- クラスタ・キャッシング・ファシリティーのトレース・ファイル
(*CAPD.tracelog.cf_pid*)
 - クラスタ・キャッシング・ファシリティーに関するデフォルトの軽量トレースが使用可能です。クラスタ・キャッシング・ファシリティーが強制終了または停止した場合、常にこれらのトレース・ファイルが生成されます。これらのファイルはクラスタ・キャッシング・ファシリティーの問題を示す可能性があります。他の診断データと組み合わせて使用する場合にのみ、エラーの診断に役立ちます。

始動および初期化のメッセージがクラスタ・キャッシング・ファシリティーのダンプ・ファイルに示されます。例えば *cfdump.out.1548476.host04.128* に関するメッセージには、プロセスの始動が成功したことを示す次のようなメッセージが含まれます。

```
CA Server IPC component Initialised: LWD BG buffer count: 16
      Session ID: 1d
CA Server IPC component Acknowledged LWD Startup Message
      Waiting for LWD to Configure Server
Processors: (4:4) PowerPC_POWER5 running at 1498 MHz

Cluster Accelerator initialized

Cluster Accelerator Object Information:
      OS: AIX 64-bit
```



```
Compiler: x1C VRM (900)
SVN Revision: 7584
Built on: Oct 12 2009 at 17:00:54
Executable generated with symbols
Model Components Loaded: CACHE LIST LOCK
Transport: uDAPL
Number of HCAs: 1
Device[0]: hca0
CF Port[0]: 50638
Mgmt Port Type: TCP/IP
Mgmt Port: 50642
IPC Key: 0xe50003d
Total Workers: 4
Conn/Worker: 128
Notify conns: 256
Processor Speed: 1498.0000 MHz
```

この例では、`cfdiag-20091109015035000037.128.log` の中にプロセス始動の成功が含まれています。クラスター・キャッシング・ファシリティーが正しく始動しなかった場合、このログは空であるか、エラー・メッセージが含まれます。例えば、

```
2009-11-09-01.50.37.0051837000-300 E123456789A779 LEVEL : Event
PID      : 688182 TID :          1
HOSTNAME : host04
FUNCTION : CA svr_init, mgmnt_cfstart
MESSAGE  : CA server log has been started.
DATA #1  :
Log Level: Error
Debugging : active
Cluster Accelerator Object Information
  AIX 64-bit
  Compiler: x1C VRM (900)
  SVN Revision: 7584
  Built on Oct 12 2009 at 17:00:59
  Executable generated with symbols.
  Executable generated with asserts.
  Model Components Loaded: CACHE, LIST, LOCK
  Transport: uDAPL
  Number of HCAs: 1
  Device[0]: hca0
  CF Port[0]: 50638
  Total Workers: 4
  Conn/Worker: 128
  Notify conns: 256
  Processor Speed: 1498.000000 Mhz.
  Allocatable Structure memory: 170 MB
```

障害の発生したクラスター・キャッシング・ファシリティーと同じ CF ID を持つ `cfdiag log` を探すことにより、関連のクラスター・キャッシング・ファシリティー診断ログ・ファイルを探します。例えば (前の `db2instance -list` コマンドと同様に) CF ID 128 に障害が発生した場合、次のコマンドを使用します。

```
$ ls cfdiag*.128.log
```

```
cfdiag.128.log -> cfdiag-20091109015035000215.128.log
cfdiag-20091110023022000037.128.log
cfdiag-20091109015035000215.128.log
```

`cfdiag.128.log` が常に CF 128 に対する最新の `cfdiag log` を指し示すことに注意してください。 `cfdiag-20091109015035000037.128.log` (直前の `cfdiag log`)、および `2009-11-10-02.30.22.000215` に対応する時刻の `db2diag` ログ・ファイルを調べてエラーを確認します。

さらに、エラーの原因がまだ分からない場合は、影響を受けたホストのシステム・エラー・ログを調べることもできます。始動していないクラスター・キャッシング・ファシリティー・ホストにログオンし、**errpt -a** コマンドを実行してシステム・エラー・ログを表示します (Linux では、/var/log/messages ファイルを調べます)。ここに示す例では、CF 128 で障害が発生したため hostD にログインします。

ホーム・ホストでの複数のメンバー再始動:

db2instance -list コマンドの出力には問題がないように見え、メンバーとホストの ALERT 列にはアラートのフラグが立てられていません。しかし、存在する FODC ディレクトリーの数が非常に多い (または増加している) か、**diagpath** 診断ダンプ・ディレクトリー (デフォルトでは *INSTHOME/sqllib/db2dump/ \$m*) のスペース使用量が異常なペースで増え続けていることが、過去の動作に基づいて観察される場合があります。

以下は、**db2instance -list** コマンドからの出力のサンプルで、3 つのメンバーと 2 つのクラスター・キャッシング・ファシリティー環境が示されています。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	STARTED	hostA	hostA	NO	0	0	hostA-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	ACTIVE	NO	NO
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

この症状は、ホーム・ホストでのメンバー再始動によって発生する可能性があります。継続する (または繰り返される) 問題のためにホーム・ホスト上で複数のメンバー再始動が発生すると、**diagpath** 診断ダンプ・ディレクトリー (デフォルトでは *\$INSTHOME/sqllib/db2dump/ \$m*) のスペース使用量が増加する可能性があります。

- *instance_owner.nfy* ログを調べて、障害がいつ発生したかについての情報を探します。
- 障害が発生した理由の詳細について、このメンバーの **db2diag** ログ・ファイルでこのタイム・スタンプの周辺の項目を確認します。

注: **db2diag** ログ・ファイルで db2rstar に関連するエラー・メッセージを検査します。

- **diagpath** の場所 (またはデフォルトでは *sqllib/db2dump/ \$m* ディレクトリー) にある FODC ディレクトリーを探します。FODC ディレクトリーが存在する場合は、その後の手順について、『First Occurrence Data Capture 情報』の関連リンクを参照してください。
- FODC ディレクトリーが存在せず、エラーの原因が依然として分からない場合は、影響を受けたホストのシステム・エラー・ログを調べてください。
 - 各ホストにログインします
 - AIX では、**errpt -a** コマンドを実行してシステム・エラー・ログを表示します。
 - Linux では、/var/log/messages を確認します。

- ホスト・リブート・イベントの証拠を探します。『restart light を伴うホスト・リブートの診断』を参照してください。

restart light を伴うホスト・リブートの診断:

実行時にホストがリブートされ、restart light が発生しています。ここでは、発生した可能性のあるさまざまな状況を識別し、診断する方法について詳しく説明します。

診断

このセクションでは、ホスト・リブートによって発生した restart light を識別する方法について説明します。

例えば次のような、restart light イベントを示すメッセージが db2diag ログ・ファイルに表示されます。

```
2009-11-02-22.56.30.416270-240 I6733A457 LEVEL: Event
PID      : 1093874          TID : 1    KTID : 2461779
PROC     : db2star2
INSTANCE:                  NODE : 001
HOSTNAME: hostC
EDUID    : 1
FUNCTION: DB2 UDB, base sys utilities, DB2StartMain, probe:3368
MESSAGE : Idle process taken over by member
DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
1
```

前述の db2diag ログ・ファイル項目の時刻に基づいて、リブートを示すメッセージが AIX エラー・ログに表示されることがあります。AIX エラー・ログにアクセスするには **errpt -a** を実行します。これが発生した理由として、以下のシナリオには 3 つの可能性ががあります。

- **ユーザーによって開始されるホスト・シャットダウンとリブートが発生した。**

この状態が発生したかどうかを判別するには、AIX errpt ログの中で、以下のような項目を探します。

```
LABEL:          REBOOT_ID
IDENTIFIER:     2BFA76F6

Date/Time:      Mon Nov  2 22:56:28 EST 2009
Sequence Number: 11878
Machine Id:     00057742D900
Node Id:        coralpib17
Class:          S
Type:           TEMP
WPAR:           Global
Resource Name:  SYSPROC
```

```
Description
SYSTEM SHUTDOWN BY USER
```

```
Probable Causes
SYSTEM SHUTDOWN
```

```
Detail Data
USER ID
```

0

```
0=SOFT IPL 1=HALT 2=TIME REBOOT
0
TIME TO REBOOT (FOR TIMED REBOOT ONLY)
0
```

この例では、ユーザーがリブートを開始しました。

- **Tivoli SA MP/RSCT がリブートを開始した。**

この状態が発生したかどうかを判別するには、AIX errpt ログの中で、以下のような項目を探します。

```
LABEL:          TS_CRITICAL_CLNT_ER
IDENTIFIER:     75FA8C75
```

```
Date/Time:      Tue Mar 31 10:58:49 EDT 2009
Sequence Number: 358837
Machine Id:     0006DA8AD700
Node Id:        coralp08
Class:          S
Type:           PERM
WPAR:           Global
Resource Name:  cthats
```

```
Description
Critical client blocked/exited
```

```
Probable Causes
Group Services daemon was blocked too long or exited
```

```
Failure Causes
Group Services daemon blocked: resource contention
Group Services daemon blocked: protocol problems
Group Services daemon exited: internal failure
Group Services daemon exited: critical client failure
```

```
Recommended Actions
Group Services daemon blocked: reduce system load
Group Services daemon exited: diagnose Group Services
```

```
Detail Data
DETECTING MODULE
rsct,monitor.C,1.124.1.3,5520
ERROR ID
6plcyp/dyWo7/1Sx/p3k37.....
REFERENCE CODE
```

```
Critical client - program name
hagsd
Failure Code
BLOCKED
アクション
NODE REBOOT
```

この例では、クラスター内のクリティカル・リソースを保護するために RSCT がホストをリポートしました。

- **カーネル・パニックのためにリポートが発生した。**

AIX errpt ログ内の KERNEL_PANIC メッセージ、またはその前に書き込まれたメッセージを確認すると、カーネル・パニックを誘発させた根本的な原因を識別できる可能性があります。メッセージの LABEL、PANIC STRING、または Detail Data フィールドに MMFS_* が含まれる場合 (例えば

MMFS_GENERIC、MMFS_PHOENIX)、これは GPFS が誘因である可能性を示しています。同様に、いずれかのフィールドに TS_* または RSCT* が含まれる場合、Tivoli SA MP が誘因である可能性を示しています。この状態が発生したかどうかを判別するには、AIX errpt ログの中で、以下のような項目を探します。

```

LABEL:          KERNEL_PANIC
IDENTIFIER:     225E3B63

Date/Time:      Mon May 26 08:02:03 EDT 2008
Sequence Number: 976
Machine Id:     0006DA8AD700
Node Id:        coralpib08
Class:          S
Type:           TEMP
Resource Name:  PANIC
  
```

Description
SOFTWARE PROGRAM ABNORMALLY TERMINATED

Recommended Actions
PERFORM PROBLEM DETERMINATION PROCEDURES

詳しくは関連情報を参照してください。

トラブルシューティング

影響を受けたホストがオンラインである場合、**db2instance -list** コマンドを実行します。**db2instance -list** によってメンバーが **WAITING_FOR_FAILBACK** であると報告される場合、出力の中でアラートを探します。アラートを確認してください。場合によっては、メンバーをホーム・ホストにフェイルバックさせるには、その前にアラートをクリアする必要があります。依然としてフェイルバックが発生しない場合、712 ページの『restart light が成功した後、ホーム・ホストでメンバーを再始動できない』を参照してください。

カーネル・パニックを伴う GPFS ファイル・システム障害:

IBM General Parallel File System (GPFS) トリガーが原因で、メンバーホスト上でカーネル・パニックが発生しました。トリガーが、散発的ではありますが繰り返し発生します。

症状

db2instance -list コマンドの出力に、以下の例のような保留中のフェイルバック操作が含まれます。

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	WAITING_FOR_FAILBACK	hostA	hostB	NO	0	1	hostB-ib0
1	MEMBER	STARTED	hostB	hostB	NO	0	0	hostB-ib0
2	MEMBER	STARTED	hostC	hostC	NO	0	0	hostC-ib0
128	CF	PRIMARY	hostD	hostD	NO	-	0	hostD-ib0
129	CF	PEER	hostE	hostE	NO	-	0	hostE-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
hostA	INACTIVE	NO	YES
hostB	ACTIVE	NO	NO
hostC	ACTIVE	NO	NO
hostD	ACTIVE	NO	NO
hostE	ACTIVE	NO	NO

上記の例で、hostA の状態は **INACTIVE** であり、ALERT フィールドが **YES** となっています。hostA がオフライン状態またはリブート中である場合に、この

db2instance -list コマンド出力が示されます。メンバー 0 のホーム・ホスト hostA がオフラインであるため、メンバー 0 が hostB にフェイルオーバーしました。WAITING_FOR_FAILBACK 状態が示すように、メンバー 0 はホーム・ホストへのフェイルバックを待機しています。hostA がパニックからリブートした後、メンバー 1 が hostA にフェイルバックします。

診断

db2diag ログ・ファイルを確認すると、以下の例のように、restart light 操作が発生したことを示すログ項目が多数含まれているのを確認できます。

```
2009-08-27-23.37.52.416270-240 I6733A457          LEVEL: Event
PID      : 1093874          TID   : 1          KTID  : 2461779
PROC     : db2star2
INSTANCE:                  NODE   : 000
HOSTNAME: hostB
EDUID    : 1
FUNCTION: DB2 UDB, base sys utilities, DB2StartMain, probe:3368
MESSAGE  : Idle process taken over by member
DATA #1  : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2  : Database Partition Number, PD_TYPE_NODE, 2 bytes
0
```

この種類の問題を診断する別の方法として、システム・ログを確認できます。AIX errpt システム・ログの内容を表示するために OS コマンド **errpt -a** を実行します。AIX errpt ログの中で、hostA に関して以下の例のようなログ項目が見つかる可能性があります。

```
LABEL:          KERNEL_PANIC
IDENTIFIER:     225E3B63

Date/Time:      Mon May 26 08:02:03 EDT 2008
Sequence Number: 976
Machine Id:     0006DA8AD700
Node Id:        hostA
Class:          S
Type:           TEMP
Resource Name:  PANIC

Description
SOFTWARE PROGRAM ABNORMALLY TERMINATED

Recommended Actions
PERFORM PROBLEM DETERMINATION PROCEDURES

Detail Data
ASSERT STRING
5.1: xmemout succeeded rc=d

PANIC STRING
kx.C:2024:0:0:04A53FA8::adv0bjP == ofP->advLk0bjP
```

上記の例のような KERNEL_PANIC ログ項目が表示される場合、システム・リブートの原因は、GPFS サブシステムの問題によってトリガーされたオペレーティング・システム・カーネル・パニックである可能性があります。カーネル・パニックとシステム・リブートが発生するのは、GPFS デーモンが重要なタスクを実行するための十分なシステム・リソースを受け取っていないときに、システムで過剰なプロセッサ使用または負荷が高いページングが発生した結果である可能性があります。カーネル・パニックに関連する GPFS ファイル・システムの障害が発生した場合

は、プロセッサ使用量またはページングについての内在する問題を最初に解決する必要があります。そのような内在する問題を解決できない場合は、データベースに対して **-s** パラメーターを指定して **db2support** コマンドを実行して診断情報を収集し、IBM 技術サポートにお問い合わせください。

db2cluster コマンドのトラブルシューティング・オプション

db2cluster コマンドの拡張ヘルプ・メニューでは、クラスター・マネージャーまたは共有ファイル・システム・クラスターに関する問題を診断し、トラブルシューティングを行う上で役立ついくつかのオプションがリストされます。これらのオプションは、サービスによる指示の下でのみ使用することをお勧めします。

クラスター・マネージャー・リソース・モデルの修復

クラスター・マネージャー・リソース・モデルと `db2nodes.cfg` ファイルの間に不整合が存在する場合、DB2 pureScale インスタンスまたは新しいリソースの開始試行は失敗します。その場合、クラスター・マネージャー・リソース・モデルを修復し、**db2start** コマンドを再発行する必要があります。

始める前に

このコマンドを実行する前に、DB2 インスタンスを完全に停止する必要があります。

このタスクについて

このタスクは、`db2nodes.cfg` ファイルとクラスター・マネージャーの間に不整合が存在することが検出された場合（通常は **db2start** が失敗したときに生じる）にクラスター・マネージャー・リソース・モデルを修復する方法を示します。これらの不整合は、以下のいずれかの状況が生じると発生します。

- `db2nodes.cfg` ファイルが手動で編集された場合
- リソース・モデルが手動で編集された場合
- ノードの追加またはドロップ試行に失敗した場合

db2cluster コマンドを使用し、リソース・モデルを削除して内部ファイルに保管されている情報を使ってリソース・モデルを再作成します。

この操作を実行するには、SYSADM、SYSCTL、または SYSMAINT グループのユーザーである必要があります。

手順

クラスター・マネージャー・リソース・モデルを修復するには、オンライン・ホストから以下のコマンドを発行します。

```
db2cluster -cm -repair -resources
```

タスクの結果

クラスター・リソース・モデルを修復できない場合は、この問題を解決する方法について、IBM サービス担当員にお問い合わせください。

クラスター・マネージャー・ドメインの修復

DB2 pureScaleインスタンスで、クラスター・マネージャー・ドメインの再作成が必要となる障害状態が発生した場合は、**db2cluster** コマンドを使用して再作成します。

始める前に

このタスクを実行する前に、DB2 インスタンスを停止する必要があります。クラスター内のすべてのノードはオンラインでなければなりません。

このタスクについて

db2cluster コマンドを使用して、インスタンスのドメインおよびリソース・モデルを削除して再作成します。

ドメインは、既存のドメインと同じトポロジーおよび構成 (クラスター・サービス・タイブレーカーやホスト障害検出時間など) を使用して再作成されます。

制約事項

このタスクで使用するコマンドは、DB2 クラスター・サービス管理者としてのみ実行できます。

手順

1. **DB2INSTANCE** 環境変数を使用してターゲット・インスタンスを指定します。

```
export DB2INSTANCE=<inst-name>
```
2. **db2cluster** コマンドに **-repair -cluster** オプションを指定して発行します。

```
db2cluster -cm -repair -domain <domain-name>
```

まだオンラインのリソースが存在する場合は、**db2cluster** は失敗し、**-force** オプションを使用してコマンドを再実行する必要があることを示すメッセージが出されます。

タスクの結果

クラスター・マネージャー・ドメインが正常に再作成されたら、**db2start** コマンドを使用してインスタンスをオンラインに戻してください。

クラスター・マネージャー・ドメインを正常に再作成できない場合は、この問題からの復旧方法について、IBM サービス担当員にお問い合わせください。このとき、元のクラスター・マネージャーの構成情報を提供するように求められる可能性があります。この情報は、**db2cluster** により、クラスター・マネージャー・ドメインの解体前にテキスト・ファイル (/tmp/ibm.db2.db2cluster_cm_repar_domain_config_<TIMESTAMP>) に保存されています。

例

DB2 クラスター・サービス権限を持つ DBA が、DB2 インスタンス MYINST 内のクラスター・マネージャー・ドメイン MYDOMAIN を再作成する必要があるとします。

```
export DB2INSTANCE=myinst1
```



```
db2cluster -cm -repair -domain mydomain
```

ドメインを解体して再作成するとき、**db2cluster** は、以下のような操作の進行および正常終了についての通知メッセージを出します。

```
Deleting the domain 'mydomain' from the cluster ...  
Deleting the domain 'mydomain' from the cluster was successful.  
Creating domain 'mydomain' in the cluster ...  
Creating domain 'mydomain' in the cluster was successful.  
Configuring quorum device for domain 'mydomain' ...  
Configuring quorum device for domain 'mydomain' was successful.  
The host failure detection time has been set to 8 seconds.  
The resource model for the instance 'myinst1' has been re-created.  
The cluster manager domain has been successfully repaired.
```

正常でないホストに対応するための処理の設定

まれな状態ではありますが、クラスター内の 1 つのホストで過度な負荷またはページングが発生して、そのクラスターの他の部分のスループットまたは可用性に影響を及ぼすことがあります。

このような状態への標準的な対応は、構成変更またはハードウェア変更を行ってメモリーを増やすことですが、これができない場合は、クラスターへの影響を防ぐための自動修正処置である、**正常でないホストに対応するための処理** を行うよう指定することもできます。

始める前に

このタスクを実行するには、DB2 クラスター・サービス管理者である必要があります。

このタスクについて

デフォルトでは、この非正常ホストの問題が発生した場合、DB2 クラスター・サービスは何の処置も行いません。この正常でないホストのフィーチャーに関連する情報はログに書き込まれず、他のツールを使用して、正常でないホストに関連するシステムの詳細をモニターする必要があります。このタスクでは、このような問題が検出されたときに以下の 2 つの処置のうちいずれかを自動的に行うように指定する方法について説明します。

ホストをリポートする

これは、ホストをリポートして、強制的に、そのホスト上のメンバーを別のホスト上で **light** モードで再始動し、1 次クラスター・キャッシング・ファシリティは 1 次役割を 2 次クラスター・キャッシング・ファシリティにフェイルオーバーします。リポートが完了すると、常駐メンバーがホスト上で再始動します (ただし自動フェイルバックが無効になっている場合を除く)。さらに、クラスター・キャッシング・ファシリティは、すべて 2 次クラスター・キャッシング・ファシリティとして再始動します。これは、正常でないホストの問題がクラスターの他の部分に与える影響を最小限に抑えるための最も確実な方法であるため、推奨されるオプションです。

メンバーまたはクラスター・キャッシング・ファシリティをオフラインにする

これは、ホスト上のすべてのメンバーとクラスター・キャッシング・ファシリティをオフラインにします。メンバーのすべての処理は停止され、別のホストで **light** モードで再始動します。このホスト上に 1 次クラスター・

キャッシング・ファシリティが存在する場合は、2次クラスター・キャッシング・ファシリティが1次役割をテークオーバーします。アラートが出され、それを手動でクリアしなければ、メンバーがフェイルバックするようにも、クラスター・キャッシング・ファシリティが再始動するようにもなりません。このオプションは、影響を与えたくない他のものがホスト上に存在する場合や、問題を診断するためにホストを使用できる状態で維持する場合に適しています。出されたアラートを手動でクリアすることが目的で、このオプションを選択する場合があります。

繰り返しになりますが、このオプションを設定するのは、予期しないクラスターの問題を防止するための、まさに最後の手段としての対応です。このような非正常ホストの状態の発生が懸念される場合、最善の対応は、メモリー・リソースをホストに追加することです。

手順

正常でないホストに対応するための処理をセットアップするには、以下のようになります。

- ホストを自動的にリポートするには、以下のコマンドを実行します。

```
db2cluster -cm -create -unhealthy_host_response -option -reboot_host -option -apply_to_current_host
```

- ホスト上のメンバーまたはクラスター・キャッシング・ファシリティを自動的にオフラインにするには、以下のコマンドを実行します。

```
db2cluster -cm -create -unhealthy_host_response -option -offline_member -option -apply_to_current_host
```

タスクの結果

自動応答としてホストのリポートを指定した場合、ホストが正常にリポートすると、すべてのメンバーは、そのホスト上で再始動するはずですが、リポート後は、自動フェイルバックを無効にした場合を除き、メンバーをホスト上で再始動する必要があります。このホスト上のすべてのクラスター・キャッシング・ファシリティはそのホストで再始動しますが、その直前まで1次クラスター・キャッシング・ファシリティとしてホストに存在していた場合は、そのホストの2次クラスター・キャッシング・ファシリティになります。リポート・イベントに関する情報は、**syslog** ファイルに書き込まれます。詳しくは、『[関連リンク](#)』セクションを参照してください。

メンバーまたはクラスター・キャッシング・ファシリティをオフラインにするように指定していて、それが発生した場合、メンバーまたはクラスター・キャッシング・ファシリティでアラートが生成されます。オフラインになったメンバーは、このアラートが削除されるまで、ホーム・ホストでの再始動やホーム・ホストへのフェイルバックを行いません。オフラインになったクラスター・キャッシング・ファシリティは、このアラートが削除されるまで再始動しません。

アンインストール

このセクションには、IBM DB2 pureScale Feature のアンインストール中に発生する可能性がある問題の理解および解決に役立つ情報が含まれています。

不完全な DB2 pureScaleインスタンス・ドロップのクリーンアップ

db2idrop コマンドが、インスタンスの完全ドロップに失敗した場合は、DB2 pureScaleインスタンスのクリーンアップを完了しなければなりません。

手順

不完全な DB2 pureScaleインスタンス・ドロップをクリーンアップするには、以下のようになります。

1. 次のようにして RSCT ピア・ドメイン (PD) を強制的に除去します。
 - `rmrpdomain -f domain_name` の実行を試みます。

ドメイン・名前が分からない場合は、**lsrpdomain** を実行して Name 列の下でドメイン・名前を探します。

```
$ lsrpdomain
Name      OpState RSCTActiveVersion MixedVersions TSPort GSPort
db2domain Online  2.5.3.5           No           12347  12348
```

2. DB2 クラスター・ファイル・システムをクリーンアップします。
3. すべてのホストでインスタンス・ユーザーの `$HOME/sql1lib` をクリーンアップします。
4. DB2 pureScale インスタンスのユーザーに関する `/etc/services` 内の項目をクリーンアップします。
5. インスタンス・シグニチャーを除去します
`installed_path/instance/db2iset -d instance_name`
6. すべてのホストで以下のグローバル・レジストリーを除去します。
 - INSTPROF
 - PEER_DOMAIN

例えば、

```
installed_path/bin/db2greg -delvarrec
service=GPFS_CLUSTER,variable=NAME,installpath=-
```

```
installed_path/bin/db2greg -delvarrec
service=PEER_DOMAIN,variable=NAME,installpath=-
```

```
installed_path/bin/db2greg -delvarrec
service=DEFAULT_INSTPROF,variable=DEFAULT,installpath=-
```

```
installed_path/bin/db2greg -delvarrec
service=INSTPROF,variable=<instance name>,installpath=-
```

IBM Reliable Scalable Cluster Technology ピア・ドメインのクリーンアップ

このトピックでは、Reliable Scalable Cluster Technology (RSCT) ピア・ドメインを手操作でクリーンアップするのに必要な手順を示します。 IBM DB2 pureScale Feature と共に DB2 Enterprise Server Edition をアンインストールする際、通常は

RSCT が除去されます。予期しないエラーが発生して手操作によるクリーンアップが必要になった場合は、以下の手順に従ってください。

このタスクについて

db2idrop -g コマンドを実行すると、通常は IBM Reliable Scalable Cluster Technology (RSCT) ピア・ドメインが自動的に除去されます。以下の手操作の手順は、問題が発生した場合にのみ必要となります。

手順

1. **lsrpdomain** コマンドを実行することにより、既存のすべてのピア・ドメインをリストします。このコマンドの出力は、例えば次のようになります。

```
lsrpdomain
```

何もリストされない場合、アクティブなピア・ドメインはありません。任意の時点でアクティブ (オンライン) 状態になることができる RSCT ピア・ドメインの数は 1 つだけです。停止、除去、ノードの追加などの操作はこのオンライン・ドメインにのみ影響を与えます。

2. **db2cluster -cm -list -host -state** コマンドを実行することにより、このアクティブ・ピア・ドメイン内のすべてのノードをリストします。このコマンドの出力は、例えば次のようになります。

```
HOSTNAME  STATE
-----  -
coralpi135  ONLINE
coralpi136  ONLINE
```

3. すべてのノードを含むピア・ドメイン全体を除去します。以下のコマンドを正常に実行するには、ピア・ドメインがオンラインでなければなりません。

db2cluster -cm -delete -domain db2domain コマンドを実行します。このコマンドの出力は、例えば次のようになります。

```
Deleting the domain db2domain from the cluster ...
Deleting the domain db2domain from the cluster was successful.
```

4. ピア・ドメインが適切に除去されたことを確認するには、**db2cluster -cm -list -domain** コマンドを実行します。

DB2 pureScale Feature のアンインストールの問題に関するよくある質問

このトピックの情報を使用すると、IBM DB2 pureScale Feature のアンインストールに関する問題の解決策を判別できます。

db2idrop コマンドで GPFS クラスタからホストを除去しようとして失敗した場合、どうすればよいでしょうか?

*DB2DIR/tmp/ibm.db2.cluster.** ディレクトリー内の **db2cluster** コマンド・ログ・ファイルを確認すると、失敗の原因を判別するのに役立ちます。問題を解決した後、**db2idrop** コマンドを再発行します。

インスタンスが使用可能でないため db2idrop コマンドが失敗した場合、どうすればよいでしょうか？

DB2DIR/tmp ディレクトリー内の db2idrop コマンド・ログを確認すると、失敗の原因を判別するのに役立ちます。 instance_user/sqlllib/db2nodes.cfg が有効かどうか確認してください。

特定のホスト上で GPFS CLUSTER グローバル・レジストリー (または PEER DOMAIN グローバル・レジストリー) を除去しようとして db2idrop コマンドが失敗した場合、どうすればよいでしょうか？

この失敗が発生した場合、IBM ソフトウェア・サポートに連絡してください。

db2idrop コマンドで RSCT ピア・ドメインの除去が失敗した場合、どうすればよいでしょうか？

db2idrop コマンドで RSCT ピア・ドメインの除去に失敗した場合、以下の手順に従って手操作でこれを除去する必要があります。

1. **lssam** コマンドを実行して、ピア・ドメインに引き続きアタッチされているリソースがあるかどうか確認します。
2. アタッチされているリソースが依然として存在する場合は、以下の手順を実行します。
 - a. `su - instance_owner` を入力することにより、インスタンス所有者に切り替えます。
 - b. `db2cluster -cm -delete -resources` を入力することにより、リソースを除去します。
 - c. 元どおり `root` に切り替えます。
3. DB2DIR/bin ディレクトリーから `db2cluster -cm -delete -domain domain_name` コマンドを実行することによって RSCT ピア・ドメインを除去します。指定すべきドメイン・ネームを判別するには `lsrpdomain` コマンドを実行します。

詳しくは、『手操作による IBM Reliable Scalable Cluster Technology ピア・ドメインのクリーンアップ』を参照してください。

db2_deinstall コマンドが失敗した場合、どうすればよいでしょうか？

db2_deinstall コマンドを使用すると、DB2 データベース製品、フィーチャー、または言語を db2_deinstall コマンドの実行場所のホストからアンインストールできます。ホストが依然として GPFS クラスタに含まれる (またはホストが引き続き RSCT ピア・ドメインに属する) ために db2_deinstall コマンドが失敗することがあります。ホストが依然として GPFS クラスタに含まれる場合は、db2_deinstall -s GPFS コマンドを実行してそれを除去してください。

DB2 pureScale インスタンスが引き続き存在する場合、db2iupdt -drop または db2idrop -g コマンドを使って DB2 pureScale インスタンスを除去できます。その後、db2_deinstall コマンドを再実行してください。

ピア・ドメインを保持したい場合には、db2_deinstall -a -s TSAMP を実行することにより IBM Tivoli System Automation for Multiplatforms バイナリー・ファイルの

除去操作を迂回できます。このホスト上でピア・ドメインを使用するリソースが存在しない場合は、コマンドを再発行する前に、手操作で Tivoli SA MP を除去してください。

第 8 章 DB2 Text Search のトラブルシューティング

DB2 Text Search のトラブルシューティングとメンテナンスのタスクには、孤立したテキスト検索コレクションの削除やログとトレースのプロパティの調整などが含まれます。

DB2 Text Search の問題判別を行うために役立つリソースが 3 つあります。つまり、イベント表、**db2trc** コマンド、および DB2 Text Search サーバーのログ機能とトレース機能です。

テキスト検索操作での DB2 トレース機能の使用

IBM 担当員にエラーを報告する必要がある場合、エラーの特定に使用されるファイルに情報を書き込めるようにするため、トレースをオンにするように依頼されることがあります。

このタスクについて

DB2 のトレース機能 **db2trc** を使用して、テキスト検索やテキスト検索に関連する管理操作を実行する際に発生する可能性のある問題について、情報を収集することができます。

トレースをオンにするとシステムのパフォーマンスに影響が出るため、トレース機能は、IBM サポートの担当者やご使用のシステムの技術サポート担当者から指示があった場合にのみ使用してください。

トレースをオンにして DB2 Text Search 特有の情報を受け取るには、CIE (155) のコンポーネント・コードにマスクを使用して **db2trc** コマンドを実行します。次のようにします。

```
db2trc on -m '.*.CIE.*'
```

加えて、**db2diag** ログ・ファイルも確認するなら、重大エラーの診断に役立ちます。

Text Search サーバーでは追加のトレース機能も利用できます。詳しくは、DB2 Text Search サーバーのロギングとトレースに関するトピックを参照してください。

DB2 Text Search サーバーのロギングとトレース

DB2 Text Search サーバーは、システム・メッセージとトレース・メッセージを記録し、問題が発生した場合の原因の判別を助けます。収集する診断データ、およびそのデータの収集元となるソースは、現在調査している問題のタイプによって異なります。

特定の症状では、その問題に関係する追加のデータを次の場所から収集しなければならないことがあります。

- `<install-path>`

- `<DBPATH>/db2tss` - 統合テキスト検索サーバーの場合
- `<ECMTS_HOME>` - スタンドアロン・テキスト検索サーバーの場合
- `<absolute-path-to-config-folder>` - テキスト検索サーバーの構成ファイルの場所
- `<collection-path>` - CREATE INDEX 操作で COLLECTION DIRECTORY として指定されるディレクトリー (統合テキスト検索サーバーの場合のみ)、またはデフォルトが使用された場合は、defaultDataDirectory の指定 (統合テキスト検索サーバーおよびスタンドアロン・テキスト検索サーバーの場合)
-

スタンドアロンの DB2 Text Search サーバーの診断データを収集するには、次のようにします。

1. `version` コマンドを使用して管理ツールを実行します。

```
adminTool version -configPath <absolute-path-to-config-folder>
```
2. テキスト検索サーバーに対して構成された `logpath` に保管されているログ・ファイルを収集します。
3. 場合によっては、より詳細な情報を収集することをお勧めします。
 - `<absolute_path_to_config_folder>` ディレクトリーの `ecmts_config_logging.properties` ファイルを編集することによって、コマンド行ツールのロギング・プロパティを構成します。
 - `<absolute_path_to_config_folder>` ディレクトリーの `ecmts_logging.properties` ファイルを編集することによって、トレース・ロギング・プロパティを構成します。
 - `adminTool` コマンドを使用してトレースを使用可能にします。

```
adminTool configureTrace -configPath <absolute-path-to-config-folder> -trace on
```
4. `<absolute-path-to-config>` ディレクトリーから次の構成ファイルを収集します。
 - a. `config.xml`
 - b. `jetty.xml`
 - c. `key.txt`
 - d. `authentication.xml`
 - e. `constructors.xml`
5. 収集の作成、削除、索引付け、または検索を実行するときに収集関連の問題が発生した場合は、`<collection-path>` ディレクトリーで収集構成ファイルを探し、必要に応じて `<collection-path>/data` ディレクトリーから収集データを見つめます。
6. ストップワードに関連する問題が発生した場合は、`<install-path>/resource/uima` フォルダーからストップワード XML ファイルを収集します。

ストップワード XML ファイルの名前は、`<language_code>-Stw.xml` です。
7. シノニムに関連する問題が発生した場合は、`<collection-path>/data/synonym` ディレクトリーからシノニムを収集します。

DB2 Text Search 索引更新のキューのモニター

索引付けのパフォーマンスの問題が発生した場合、DB2 Text Search サーバー構成の調整に使用できるモニター情報を収集できます。

-monitorQueues フラグを設定して DB2 Text Search サーバーを始動することによって、入力キューと出力キューをモニターできます。-monitorQueues フラグを指定してサーバーを始動すると、入力キューと出力キューの現在の状態に関する情報が InputQueueSizes.csv ファイルと OutputQueueSizes.csv ファイルに出力されます。これらのファイルは、ECMTS_HOME¥logs ディレクトリーに保管されます。各ファイルには、次の情報が記載されます。

- 時刻。
- プリプロセスおよび索引付けされるのを待機している文書の現在の数。
- 現在プリプロセスおよび索引付けされている文書の数。
- 現行のキュー・サイズ。最大キュー・サイズは、構成可能なパラメーターです。
- キューを移動した文書の合計数。つまり、最後のサーバーの始動以降にプリプロセスおよび索引付けされた文書の数。

重要: CSV ファイルはサーバーが始動するたびに再作成されます。これらのファイルに情報を保管する場合、サーバーを再始動する前にバックアップしてください。

モニター・キュー・モードでサーバーを始動するには、次のようにします。

1. サーバーを停止します。
2. ECMTS_HOME¥bin ディレクトリーの始動スクリプトを編集します。
3. -monitorQueues フラグを始動コマンドの 1 つに追加します。例えば、

```
%JAVA_HOME%\bin\java" -classpath "%CLASSPATH%"  
com.ibm.es.nuvo.launcher.Launcher "%CONFIG_XML%" -monitorQueues  
-Xmx%HEAP_SIZE% %1 %2 %3 %4 %5 %6 %7 %8 %9 %IPv6Flag% %JVM_OPTIONS%
```
4. 始動スクリプトを保存して、サーバーを再始動します。

次の表の情報をを使用して、キュー状況に基づいて索引付けのパフォーマンスのトラブルシューティングを行います。

表 89. キュー状況に基づいた、索引付けのパフォーマンスのトラブルシューティング

キュー状況	トラブルシューティング戦略
入力キューが空である	クライアントから文書を受け取っていません。クライアントが十分な速さで文書をサーバーにプッシュしていることを確認します。

表 89. キュー状況に基づいた、索引付けのパフォーマンスのトラブルシューティング (続き)

キュー状況	トラブルシューティング戦略
入力キューが満杯で、出力キューが空である	<p>ログでプリプロセス・エラーを確認して、プリプロセス・スレッドの数を増やすことを検討します。大規模なマルチプロセッサ・システムを構成するサーバーにおいて CPU レベルが低い (60% 未満) 状態は、プリプロセス・スレッドの数を増やす必要があることも示している可能性があります。プリプロセス・スレッドの数は、プロセッサの数を超えないようにする必要があります。分散アーキテクチャーでは、それぞれのプリプロセス・サーバーが稼働していて正しく構成されていることを確認します。例えば、接続パラメーター (索引付けサーバーのホスト名、ポート番号、およびトークン) がそれぞれのプリプロセス・サーバーにおいて正しく構成されていることを確認します。それぞれのプリプロセス・サーバーの入力キューをモニターすることによって接続を確認し、文書を受け取れることを確認します。また、それぞれのプリプロセス・サーバーでプリプロセス・スレッドの数を増やしてみることもできます。</p>
入力キューと出力キューの両方が満杯である	<p>次の方法を 1 つ以上試してみます。</p> <ul style="list-style-type: none"> • 索引スレッド数を増やします。 • <code>ECMTS_HOME¥config¥collections¥collection_name¥collection.xml</code> ファイルの MaxMergeDocs パラメーターの値を減らすことによって、短いマージを高い頻度で行うように構成します。 • バッチ・サイズを大きくします。小さいバッチで文書の索引付けを行うと、索引付けされるバッチごとに索引がフラッシュされてディスクに保存されるため、パフォーマンスが低下する可能性があります。 • <code>ECMTS_HOME¥config¥collections¥collection_name¥collection.xml</code> ファイルの BufferSize パラメーターを増やします。 • 1 回に 1 つのコレクションを索引付けします。 • ディスクの速度とネットワーク接続が十分速いか確認します。

DB2 Text Search のトラブルシューティングのヒント

次の表の情報を使用すると、問題のトラブルシューティングに役立ちます。

表 90. 問題および解決策

課題	解決方法
DB2 接続セッションがアクティブであるときにテキスト索引をドロップして再作成する場合、以前に使用された照会 (Net Search Extender または DB2 Text Search 用) のプランがキャッシュに入っていて再使用される可能性があるため、結果が不正確になることがあります。DB2 Text Search エンジンがこの問題を検出した場合、検索に不正確な集合名が使用されたことを示す IQQG0037W が返されます。	FLUSH PACKAGE CACHE DYNAMIC ステートメントを使用して、SQL ステートメントの実行プランをステートメント・キャッシュから削除します。
db2idrop コマンド、 db2iupdt コマンド、または db2iupgrade コマンドを使用して DB2 インスタンスをドロップまたは更新する場合、インスタンス内の <code>sqllib</code> ディレクトリー全体 (<code>sqllib/db2tss/config</code> ディレクトリーを含む) が削除されます。これにより、テキスト検索インスタンス・データベースの再利用が困難になります。	db2idrop コマンド、 db2iupdt コマンド、または db2iupgrade コマンドを発行する前に、 <code>sqllib/db2tss/config</code> ディレクトリーをバックアップしてください。また、 configTool コマンドの printAll オプションを使用して、新しいインスタンスの構成に役立つ、構成パラメーターのレコードを保存します。
ディスク・フル・エラーにより、テキスト索引のメタデータに不整合が生じることがあります。	DB2 Text Search の資料の『孤立したテキスト検索コレクションの削除』で概要を示しているように、管理ツールを使用して孤立したコレクションがあるか確認して削除します。ディスク・フル・エラーを検出したときに使用されていたテキスト索引名は、再利用できないことがあるので注意してください。
db2ts START FOR TEXT コマンドを発行した後に、メッセージ CIE00301「権限が十分ではありません」または CIE00377「このコマンドを使用できるのはインスタンス所有者 <instance-owner> だけです」を受け取ります。	Windows ユーザーの場合、十分な権限を持つユーザーが db2ts START FOR TEXT を発行するようにします。Linux および UNIX の場合、インスタンス所有者がこのコマンドを発行するようにします。
Text Search サーバーでディスク・フル・エラーが発生した場合、次のエラー・メッセージが返されることがあります。 IQQD0085E The requested operation cannot be done. The server is running in safe mode due to index has an IO error.	DB2 Text Search サーバーは、制限モードで実行されています。Text Search サーバーをシャットダウンして、ディスク・スペースの問題を解決した後で再始動します。
内部ファイルを開くことができないことを示すメッセージ CIE00311 を受け取ります。このメッセージは、構成ディレクトリーが欠落していることを示している可能性があります。また、ファイルが失われたか破損したことを示している場合もあります。この障害は、db2tss/config のあるファイル・システム上のディスク・フル・エラーや、db2tss/config ディレクトリーのバックアップ中の問題などが原因で発生します。	インスタンスに、テキスト検索用の <code>config</code> ディレクトリーがあるかどうかを確認します。 <ul style="list-style-type: none"> • <code>config</code> ディレクトリーがない場合、DB2 Text Search がインストール済みで構成されていることを確認します。 • <code>config</code> ディレクトリーが別の場所にある場合、そこへのシンボリック・リンクを追加します (UNIX)。 <p>このエラーがファイルの欠落または破損が原因で発生している場合は、IBM サポートにお問い合わせください。</p>

表 90. 問題および解決策 (続き)

課題	解決方法
<p>要求された操作を実行できないことを示すメッセージ CIE00445N を受け取ります。 REBIND を実行します。このメッセージは、以前の DB2 コマンドのいずれかが DB2 Text Search で必要なパッケージ NULLID.SYSSH100、NULLID.SYSSH200、NULLID.SYSSN100、NULLID.SYSSN200、または NULLID.SYSLH202 を無効にしたことを示している可能性があります。</p>	<p>ユーザーは、db2ts コマンドを連続して実行する前に、NULLID.SYSSH100、NULLID.SYSSH200、NULLID.SYSSN100、NULLID.SYSSN200、または NULLID.SYSLH202 のパッケージを手動で再バインドする必要があります。パッケージの無効化が通常発生するのは、REVOKE コマンドまたは GRANT コマンドが実行されるときです。</p> <p>以下の SQL ステートメントを使ってパッケージの状態を確認することにより、このエラーの発生を防ぐことができます。</p> <pre>select 1 from syscat.packages where VALID = 'N' AND ((PKGSCHEMA='NULLID' AND PKGNAME= 'SYSSH100') OR (PKGSCHEMA='NULLID' AND PKGNAME= 'SYSSH200') OR (PKGSCHEMA='NULLID' AND PKGNAME= 'SYSSN100') OR (PKGSCHEMA='NULLID' AND PKGNAME= 'SYSSN200') OR (PKGSCHEMA='NULLID' AND PKGNAME= 'SYSLH202')) FETCH FIRST 1 ROWS ONLY FOR READ ONLY;</pre> <p>この SQL ステートメントの戻り値が 1 の場合、DB2 Text Search コマンドを実行する前に、パッケージ NULLID.SYSSH100、NULLID.SYSSH200、NULLID.SYSSN100、NULLID.SYSSN200 または NULLID.SYSLH202 を再バインドする必要があります。</p>
<p>テキスト索引の更新時に処理文書が大量にある場合は、db2diag.log ファイルに「メモリ不足です」のメッセージが記録されて失敗します。</p>	<p>使用可能メモリーを増やすことができない場合は、sysibmts.tsdefaults 管理ビューで documentqueueresultsize の値を減らし、もう一度試します。</p>
<p>データの再配分後、欠落したコレクションに関する照会に IQQG0037W メッセージが表示されます。</p>	<p>次にテキスト検索の UPDATE INDEX コマンドが発行されるときに、必ず FOR DATA REDISTRIBUTION オプションが使用されるようにします。</p>

第 9 章 トラブルシューティングについて

DB2 データベース製品で作業中の何らかの時点で、問題が発生する場合があります。こうした問題は、データベース・マネージャーから報告されることもありますし、データベースに対して実行中のアプリケーションから報告されることもあります。また、データベースが「正常ではない」というユーザーからのフィードバックとして報告されることもあります。

ここで述べる概念とツールは、データベースの操作で発生する実際の問題や問題と見なされるものに対するトラブルシューティング作業の概要を説明し、読者のトラブルシューティング作業を支援することを目的としています。適切なときに適切なデータを収集することが重要であることを強調する意味で、ツールの説明は First Occurrence Data Capture から始めます。データベース・マネージャーがデータベースの操作に関するデータをキャプチャーするために使用する他のログおよびファイルについても、オペレーティング・システムの診断ツールを含めて説明します。

詳細情報

以下のトピックは、DB2 製品での問題を効果的にトラブルシューティングするために必要な概念情報を取得するのに役立ちます。

- [トラブルシューティングについて](#)

トラブルシューティングとは、問題を解決するための体系的なアプローチのことです。その目標は、何かが期待したとおりに機能しない理由と問題の解決方法を判別することです。

- [診断データ・ディレクトリー・パスについて](#)

ユーザーのプラットフォームによっては、ダンプ・ファイル、トラップ・ファイル、診断ログ・ファイル、管理通知ログ・ファイル、アラート・ログ・ファイル、および FODC (First Occurrence Data Collection) パッケージに含まれる DB2 診断情報を、**diagpath** データベース・マネージャー構成パラメーターの指定する診断データ・ディレクトリーで見つけることができます。

- [管理通知ログ・ファイルについて](#)

DB2 データベース・マネージャーが管理通知ログに書き込むのは、DB2 ユーティリティーの状況 (**REORG**、**BACKUP** など)、クライアント・アプリケーション・エラー、サービス・クラスの変更、ライセンス交付アクティビティー、ログ・ファイル・パスおよびストレージの問題、モニターおよび索引付けアクティビティー、表スペースの問題などの情報です。データベース管理者はこの情報を使用することにより、問題の診断やデータベースの調整を行ったり、データベースのモニターを行ったりすることもできます。

- [DB2 診断 \(db2diag\) ログ・ファイルについて](#)

標準化されたメッセージ・フォーマットで管理通知ログ・メッセージの追加部分が **db2diag** ログ・ファイルに記録された場合は、データベースで何が起きているのかを理解するために、**db2diag** ログ・ファイルを最初に表示することをお勧めします。

- プラットフォーム固有のエラー・ログについて

DB2 以外のファイルやユーティリティの中にも、問題の分析に役立つものが多数あります。これらのファイルやユーティリティは、根本原因を判別するうえで、DB2 ファイルに用意される情報と同じほど重要な役割を果たすことがよくあります。

- メッセージについて

メッセージについて詳しく知ると、エラーまたは問題を識別し、適切なりカバリー・アクションを使用して問題を解決するのに役立ちます。この情報は、メッセージが生成されてログに記録される場所を理解するためにも使用できます。

- 内部戻りコードについて

内部戻りコードには、ZRC 値と ECF 値という 2 つのタイプがあります。例えば、これらは、DB2 トレース出力および **db2diag** ログ・ファイルに出現します。ZRC 値と ECF 値は通常、負の数値で、エラー状態を表すために使用されません。

- ダンプ・ファイルについて

ダンプ・ファイルは、エラー発生時に問題を診断するのに役立つと思われる追加情報 (内部制御ブロックなど) がある場合に作成されます。ダンプ・ファイルに書き込まれるデータ項目はそれぞれ、問題の判別に役立つようにタイム・スタンプが関連付けられています。ダンプ・ファイルはバイナリー・フォーマットで、DB2 お客様サポート担当者用です。

- トラップ・ファイルについて

DB2 は、トラップやセグメンテーション違反、例外などにより処理が続行できない時、トラップ・ファイルを生成します。DB2 によって予約されたすべてのシグナルまたは例外は、トラップ・ファイルに記録されます。トラップ・ファイルには、エラーが発生したときに実行されていた関数シーケンスも含まれています。このシーケンスは、「関数呼び出しスタック」または「スタック・トレース」と呼ばれることもあります。トラップ・ファイルには、シグナルまたは例外がキャッチされた時のプロセスの状態に関する追加情報も含まれています。

- First Occurrence Data Capture (FODC) について

First Occurrence Data Capture (FODC) は、DB2 インスタンスに関するシナリオ・ベースのデータをキャプチャーするためのプロセスです。FODC は、DB2 ユーザーが特定の症状に基づいて手動で呼び出すこともできれば、事前定義のシナリオまたは症状が検出されたときに自動的に呼び出すこともできます。この情報があれば、診断情報を取得するためにエラーを再現する必要性は少なくなります。

- コールアウト・スクリプト (db2cos) 出力ファイルについて

データベース・マネージャーが、パニック、トラップ、セグメンテーション違反、例外のために処理を続けられなくなると、デフォルトで db2cos スクリプトが呼び出されます。

- DB2 と OS 診断の結合について

メモリー、スワップ・ファイル、CPU、ディスク・ストレージ、その他のリソースに関連した問題を診断するには、所定のオペレーティング・システムがこれらのリソースをどのように管理するかについて十分に理解しておく必要があります。リソース関連の問題を定義するには、最低でも、そのリソースがどれだけ存在するか、ユーザーごとにどのようなリソース制限があるかを知っておく必要があります。

診断データ・ディレクトリー・パス

ユーザーのプラットフォームによっては、ダンプ・ファイル、トラップ・ファイル、診断ログ・ファイル、管理通知ログ・ファイル、アラート・ログ・ファイル、および First Occurrence Data Collection (FODC) パッケージに含まれる DB2 診断情報を、**diagpath** または **alt_diagpath** データベース・マネージャー構成パラメーターの指定する診断データ・ディレクトリーで見つけることができます。

概説

以下のディレクトリー・パス・メソッドのうちどれを使って診断データを保管できるかは、**diagpath** データベース・マネージャー構成パラメーターを使って診断データ・ディレクトリー・パスを指定したり、**alt_diagpath** データベース・マネージャー構成パラメーターを使って代替診断データ・ディレクトリー・パスを指定したりして決めることができます。

1 次診断データ・ディレクトリー・パス

DB2 インスタンスの診断データはすべて、データベースがパーティション化されているかどうかに関係なく、単一のディレクトリーに保管されます。パーティション・データベース環境では、ホスト内の各パーティションからの診断データはすべて、この単一の診断データ・ディレクトリー・パスにダンプされます。この単一の診断データ・ディレクトリー・パスは、**diagpath** 値が NULL であるか、または \$h と \$n のどちらのパターン ID も使用しない任意の有効なパス名に設定されたときのデフォルトの条件です。

代替診断データ・ディレクトリー・パス

alt_diagpath データベース・マネージャー構成パラメーターは、診断情報を保管するための 2 次パスを提供する代替診断データ・ディレクトリー・パスです。**alt_diagpath** パラメーターに指定されるパスは、**diagpath** に指定されるパスにデータベース・マネージャーが書き込みを行えない場合にのみ使用され、重要な診断情報が失われないためのものです。代替診断データ・ディレクトリー・パスを使用できるようにするには、**alt_diagpath** 構成パラメーターを設定しなければなりません。回復力を上げるために、このパラメーターを、**diagpath** の設定値とは別のファイル・システム上のパスに設定することをお勧めします。

分割された診断データ・ディレクトリー・パス

パーティション・データベース環境では、ホスト、データベース・パーティション、またはその両方に従って名前が付けられたディレクトリーに診断データを別々に保管することができます。そのため、特定の診断ディレクトリー内の各種診断ファイルには、1 つのホストからのみの診断情報、または 1 つのデータベース・パーティションからのみの診断情報、あるいは 1 つの

ホストと 1 つのデータベース・パーティション両方からの診断情報が入ります。診断データ・ディレクトリー・パスと代替診断データ・ディレクトリー・パスの両方を分割するには、**diagpath** で一度、その後 **alt_diagpath** でもう一度、分割の手順を実行する必要があります。

利点

診断データ・ディレクトリー・パスの指定には、次のような利点があります。

- 複数のデータベース・パーティションおよびホストからの診断情報をセントラル・ロケーションに統合することで、単一の診断データ・ディレクトリー・パスを設定することによって簡単にアクセスできるようになります。
- 診断データ・ディレクトリー・パスをホストごとまたはデータベース・パーティションごとに分割した場合、**db2diag** ログ・ファイルにおける競合が減るため、診断のロギング・パフォーマンスを向上させることができます。

2 次診断データ・パス **alt_diagpath** を指定することの利点として、以下があります。

- 重要な診断情報の消失に対する回復力の向上。
- **diagpath** で使用するいくつかのツールとの互換性 (分割など)。

ファイルのマージおよびレコードのソート

分割された診断データ・ディレクトリー・パスの場合、**db2diag -merge** コマンドを使用することにより、同じタイプの複数の診断ファイルのレコードをマージし、タイム・スタンプに基づいてソートすることができます。詳しくは、「コマンド・リファレンス」の『**db2diag - db2diag** ログ分析ツール・コマンド』を参照してください。

診断データのスペース所要量

diagpath パラメーターで指定されたパスに診断データを収集すると、大量の診断情報が生成されることがあります。特に、コア・ファイル・ダンプや First Occurrence Data Capture (FODC) データを別のディレクトリー・パスにリダイレクトしていない場合や、サイズが無限に大きくなる単一の **db2diag.log** ファイルを使用する場合は、これに該当します。診断データを保管できるだけの十分なスペースがなければならぬので、診断パス内のハウスキーピングを定期的に行って、十分なスペースを確実に使用できるようにしておく必要があります。

データ・サーバー上での診断データ・ロギングを構成する際に、以下の推奨事項に従うことで、診断データのスペース所要量の要件が確実に満たされるようにすることができます。

診断用の最小スペース所要量を満たす

診断ディレクトリー・パス内の使用可能なフリー・スペースの最小量は、マシンにインストールされている物理メモリー量の 2 倍にする必要があります (最小フリー・スペース = 2 x 物理メモリー)。例えば、マシンに 64 GB の物理メモリーがインストールされている場合は、ファイル・システム内の 128 GB 以上のスペースを診断データに使用できるようにする必要があります。

コア・ファイル・ダンプと FODC データを別のディレクトリー・パスにリダイレクトする

コア・ファイル・ダンプと FODC データは両方とも、短時間で大きなディスク・スペースを消費し、デフォルトでは **diagpath** データベース・マネージャー構成パラメーターで指定されたディレクトリー・パスにデータを送信します。診断ディレクトリー・パス内に使用可能なスペースをより多く確保するために、コア・ファイル・ダンプと FODC データを別のディレクトリー・パスかファイル・システムにリダイレクトできます。**DB2FODC** レジストリー変数を使用して、**diagpath** 以外のディレクトリー・パスを指すように **DUMPDIR** 変数を設定することによって、コア・ファイルが生成される場所を制御することができます。同様に、別のディレクトリー・パスを指すように **FODCPATH** 変数を設定して、FODC パッケージ・ディレクトリーの作成場所を制御することもできます。

必要なくなったファイルを移動するか除去する

diagpath 値以外の出力パスを指定しないで **db2support** コマンドを実行すると、実行結果の圧縮アーカイブは診断ディレクトリー・パスに保管されます。ファイルを IBM にアップロードし終わったら、診断ディレクトリー・パスから圧縮アーカイブを移動しないと、引き続き使用可能なディスク・スペースが消費されることに注意してください。

循環診断ログ・ファイルとアーカイブ・ログ・ファイルを構成する

デフォルトでは、単一の **db2diag.log** ファイルを使用すると、DB2 診断ログ・ファイルのサイズは無限に大きくなります。**diagsize** データベース・マネージャー構成パラメーターを設定して循環診断ログを構成すると、**diagsize** で定義されているサイズに収まる一連の循環診断ログ・ファイルと一連の循環管理通知ログ・ファイルが使用されます。ログ・ファイルが満杯になるたびに、最も古いファイルが削除され、新しいログ・ファイルが作成されます。ファイル循環 (最も古いログ・ファイルの削除) のために情報が早く失われすぎるのを避けるには、**diagsize** を 50 MB より大きい値に設定します。ただし、**diagpath** パラメーターや **alt_diagpath** パラメーターを使用して指定するディレクトリー・パスのフリー・スペースの 80% を超えないようにしてください。**db2diag -archive** コマンドを使用して **diagpath** からアーカイブし、循環診断ログ・ファイルを保存することもできます。

代替診断パスを構成する

重要な診断情報が失われないようにするフェイルセーフ動作として、**alt_diagpath** データベース・マネージャー構成パラメーターは、診断情報を保管するための代替診断データ・ディレクトリー・パスを提供します。**diagpath** によって指定されるパスにデータベース・マネージャーが書き込みを行えない場合、再度 **diagpath** が使用可能になるまで、**alt_diagpath** で指定されるパスを使用して診断情報が保管されます。回復力を上げるために、**alt_diagpath** パラメーターが指すファイル・システムが **diagpath** パラメーターと異なるようにしてください。

データベース・パーティション・サーバーごと、データベース・パーティションごと、またはその両方での診断データ・ディレクトリー・パスの分割

diagpath 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の DB2 診断データ・ディレクトリー・パス設定は、単一の診断データ・ディレクトリー内にすべての診断情報を収集します。診断データ・ディレクトリー・パスを分割することにより、データベース・パーティション・サーバー、データベース・パーティション、またはその両方に従って別々のディレクトリーが作成され、名前が付けられます。このように、以前は単一のディレクトリーに保管されていた診断ダンプ・ファイルを、診断データ・ダンプの発生元のデータベース・パーティション・サーバーまたはデータベース・パーティションに従って、別々のディレクトリーに保管できるようになりました。

始める前に

DB2 バージョン 9.7 フィックスパック 1 以降のフィックスパックが必要です。

このタスクについて

診断データ・ディレクトリー・パスを分割することにより、診断データ・ダンプの発生元のデータベース・パーティション・サーバーまたはデータベース・パーティションに従って診断情報を別々に保管できるようになります。

制約事項

パーティション・データベース環境ではほとんどの場合、診断情報の複数のソースを分けておくための診断データ・ディレクトリー・パスの分割は有用です。

手順

・ 物理データベース・パーティション・サーバーごとの診断データ・ディレクトリー・パスの分割

- デフォルトの診断データ・ディレクトリー・パスを分割するには、以下のステップを実行します。
 - 以下のコマンドを発行して、デフォルトの診断データ・ディレクトリー・パスが物理データベース・パーティション・サーバーごとに分割されるように **diagpath** 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。

```
db2 update dbm cfg using diagpath "$h"
```

このコマンドにより、以下のようにデフォルトの診断データ・ディレクトリーの下にサブディレクトリーが作成され、コンピューター名が付けられます。

Default_diagpath/HOST_db-partition-server-name

- ユーザー指定の診断データ・ディレクトリー・パス (例えば /home/usr1/db2dump/) を分割するには、以下のステップを実行します。
 - 以下のコマンドを発行して、/home/usr1/db2dump/ 診断データ・ディレクトリー・パスがデータベース・パーティション・サーバーごとに分割されるように **diagpath** 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。

```
db2 update dbm cfg using diagpath "/home/usr1/db2dump/ $h"
```

注: ブランク・スペースによって /home/usr1/db2dump/ と \$h を分離する必要があります。

このコマンドにより、以下のように /home/usr1/db2dump/ 診断データ・ディレクトリーの下にサブディレクトリーが作成され、データベース・パーティション・サーバー名が付けられます。

/home/usr1/db2dump/HOST_db-partition-server-name

• データベース・パーティションごとの診断データ・ディレクトリー・パスの分割

- デフォルトの診断データ・ディレクトリー・パスを分割するには、以下のステップを実行します。

- 以下のコマンドを発行して、**diagpath** データベース・マネージャー構成パラメーターを設定し、デフォルトの診断データ・ディレクトリー・パスをデータベース・パーティションごとに分割します。

```
db2 update dbm cfg using diagpath "$n"
```

このコマンドにより、以下のように、デフォルトの診断データ・ディレクトリーの下に各パーティションのサブディレクトリーがパーティション番号で作成されます。

Default_diagpath/NODENumber

- ユーザー指定の診断データ・ディレクトリー・パス (例えば /home/usr1/db2dump/) を分割するには、以下のステップを実行します。

- 以下のコマンドを発行して、**diagpath** データベース・マネージャー構成パラメーターを設定し、/home/usr1/db2dump/ 診断データ・ディレクトリー・パスをデータベース・パーティションごとに分割します。

```
db2 update dbm cfg using diagpath "/home/usr1/db2dump/ $n"
```

注: ブランク・スペースによって /home/usr1/db2dump/ と \$n を分離する必要があります。

このコマンドにより、以下のように、/home/usr1/db2dump/ 診断データ・ディレクトリーの下に各パーティションのサブディレクトリーがパーティション番号で作成されます。

/home/usr1/db2dump/NODENumber

• 物理データベース・パーティション・サーバーごとおよびデータベース・パーティションごとの診断データ・ディレクトリー・パスの分割

- デフォルトの診断データ・ディレクトリー・パスを分割するには、以下のステップを実行します。

- 以下のコマンドを発行して、デフォルトの診断データ・ディレクトリー・パスが物理データベース・パーティション・サーバーごとおよびデータベース・パーティションごとに分割されるように **diagpath** 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。

```
db2 update dbm cfg using diagpath "$h$n"
```

このコマンドにより、以下のようにデフォルトの診断データ・ディレクトリーの下に、データベース・パーティション・サーバー上の論理パーティションごとにサブディレクトリーが作成され、データベース・パーティション・サーバー名とパーティション番号が付けられます。

Default_diagpath/HOST_db-partition-server-name/NODENumber

- ユーザー指定の診断データ・ディレクトリー・パス (例えば /home/usr1/db2dump/) を分割するには、以下のステップを実行します。
 - 以下のコマンドを発行して、/home/usr1/db2dump/ 診断データ・ディレクトリー・パスがデータベース・パーティション・サーバーごとおよびデータベース・パーティションごとに分割されるように **diagpath** 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。

```
db2 update dbm cfg using diagpath '/home/usr1/db2dump/ $h$n'
```

注: ブランク・スペースによって /home/usr1/db2dump/ と \$h\$n を分離する必要があります。

このコマンドにより、以下のように /home/usr1/db2dump/ 診断データ・ディレクトリーの下に、データベース・パーティション・サーバー上の論理パーティションごとにサブディレクトリーが作成され、データベース・パーティション・サーバー名とパーティション番号が付けられます。

```
/home/usr1/db2dump/HOST_db-partition-server-name/NODENumber
```

例えば、boson という名前の AIX データベース・パーティション・サーバーに 3 つのデータベース・パーティションがあり、ノード番号はそれぞれ 0、1、2 であるとしてます。ディレクトリーのリスト出力は、例えば次のようになります。

```
usr1@boson /home/user1/db2dump->ls -R *
HOST_boson:

HOST_boson:
NODE0000 NODE0001 NODE0002

HOST_boson/NODE0000:
db2diag.log db2eventlog.000 db2resync.log db2samp1_Import.msg events usr1.nfy

HOST_boson/NODE0000/events:
db2optstats.0.log

HOST_boson/NODE0001:
db2diag.log db2eventlog.001 db2resync.log usr1.nfy stmmlog

HOST_boson/NODE0001/stmmlog:
stmm.0.log

HOST_boson/NODE0002:
db2diag.log db2eventlog.002 db2resync.log usr1.nfy
```

次のタスク

注:

- データベース・パーティションごとに分割された診断データ・ディレクトリー・パスを指定した場合 (\$n または \$h\$n) は常に、データベース・パーティション・サーバーごとに NODE0000 ディレクトリーが作成されます。NODE0000 ディレクトリーが作成されたデータベース・パーティション・サーバーにデータベース・パーティション 0 が存在しない場合は、NODE0000 ディレクトリーを無視することができます。
- 診断データ・ディレクトリー・パスの設定が正常に分割されたかどうかを調べるには、次のコマンドを実行します。

```
db2 get dbm cfg | grep DIAGPATH
```

診断データ・ディレクトリー・パスが正常に分割されている場合、先行ブランク・スペースを持つ値 \$h、\$n、または \$h\$n が返されます。例えば、次のような出力が返されます。

```
Diagnostic data directory path          (DIAGPATH) = /home/usr1/db2dump/ $h$n
```

分析とトラブルシューティングをより簡単にするために個々の **db2diag** ログ・ファイルをマージするには、**db2diag-merge** コマンドを使用します。追加情報については、「コマンド・リファレンス」の『db2diag - db2diag ログ分析ツール・コマンド』および 541 ページの『db2diag ツールを使用した db2diag ログ・ファイルの分析』を参照してください。

管理通知ログ

管理通知ログ (*instance_name.nfy*) は、多数のデータベース管理および保守アクティビティに関する情報を取得できるリポジトリーです。データベース管理者はこの情報を使用することにより、問題の診断やデータベースの調整を行ったり、単にデータベースのモニターを行ったりすることもできます。

UNIX および Linux オペレーティング・システム・プラットフォームにおいて、DB2 データベース・マネージャーは以下の種類の情報を管理通知ログに書き込みます (Windows オペレーティング・システム・プラットフォームの場合、管理通知イベントの記録にイベント・ログが使用されます)。

- DB2 ユーティリティー (**REORG**、**BACKUP** など) の状況
- クライアント・アプリケーションのエラー
- サービス・クラスの変更
- ライセンス交付アクティビティ
- ファイル・パス
- ストレージの問題
- モニター・アクティビティ
- 索引付けアクティビティ
- 表スペースの問題

管理通知ログ・メッセージも、標準化されたメッセージ・フォーマットを使用して **db2diag** ログ・ファイルに記録されます。

通知メッセージには、提供されている **SQLCODE** を補足する追加情報が備えられています。

管理通知ログ・ファイルには、以下の 2 種類の形式があります。

単一管理通知ログ・ファイル

instance_name.nfy という名前の 1 つのアクティブな管理通知ログ・ファイルがあり、そのサイズは無制限に大きくなります。これはデフォルトの形式であり、**diagsize** データベース・マネージャー構成パラメーターの値が 0 (このパラメーターのデフォルト値は 0 です) である場合に存在します。

循環管理通知ログ・ファイル

単一のアクティブ・ログ・ファイル (名前は *instance_name.N.nfy* で、*N* は 0 から順番に大きくなるファイル名インデックス)。ただし、**diagpath** 構成

パラメーターで定義された場所に一連の管理通知ログ・ファイルが存在します。それぞれのログ・ファイルは大きくなって制限サイズに達した時点で閉じられ、ファイル名インデックスが 1 つ大きくされた新しいログ・ファイル (*instance_name.N+1.nfy*) が作成されて開かれます。これは、**diagsize** データベース・マネージャー構成パラメーターがゼロ以外の値である場合に存在します。

注: Windows オペレーティング・システム・プラットフォームでは、単一管理通知ログ・ファイルも循環管理通知ログ・ファイルも使用できません。

diagsize データベース・マネージャー構成パラメーターを適切に設定することにより、これらの 2 種類の形式のうちどちらがシステム上に存在するかを選択できます。

構成

管理通知ログ・ファイルは、以下のデータベース・マネージャー構成パラメーターを設定することにより、そのサイズ、場所、および記録されるイベントの種類と詳細のレベルを構成できます。

diagsize

diagsize の値は、採用される管理通知ログ・ファイルの形式を決定します。値が 0 の場合、単一管理通知ログ・ファイルが採用されます。値が 0 でない場合、循環管理通知ログ・ファイルが採用され、そのゼロ以外の値によってすべての循環診断ログ・ファイルとすべての循環管理通知ログ・ファイルの合計サイズが指定されます。**diagsize** パラメーターの新規値を有効にするには、インスタンスを再始動する必要があります。詳細については、『**diagsize** - 診断ログ・ファイル・サイズ構成パラメーター』のトピックを参照してください。

diagpath

診断情報は、**diagpath** 構成パラメーターで定義された場所の管理通知ログ・ファイルに書き込まれるように指定することができます。詳細については、『**diagpath** - 診断データ・ディレクトリー・パス構成パラメーター』のトピックを参照してください。

notifylevel

notifylevel 構成パラメーターを使用して、管理通知ログ・ファイルに書き込むイベントの種類と詳細のレベルを指定することができます。詳細については、『**notifylevel** - 通知レベル構成パラメーター』のトピックを参照してください。

注: **diagsize** 構成パラメーターがゼロ以外の値に設定された場合、その値は診断データ・ディレクトリーの中に含まれるすべての循環管理通知ログ・ファイルとすべての循環診断ログ・ファイルを合わせた合計サイズを指定します。例えば、データベース・パーティションが 4 つあるシステムで **diagsize** が 1 GB に設定されている場合、通知ログと診断ログを合わせた最大合計サイズは、4 GB (4 x 1 GB) になります。

管理通知ログ・ファイル項目の解釈

テキスト・エディターを使って、問題が発生した疑いのあるマシンの管理通知ログ・ファイルを表示することができます。最新のイベントは、ファイルの一番下に記録されています。

一般的に、各項目には以下のような部分があります。

- タイム・スタンプ
- エラーが報告された場所。アプリケーション ID を使って、サーバーとクライアントのログにあるアプリケーションに関係した項目を見つけることができます。
- エラーを説明する診断メッセージ (通常「DIA」または「ADM」で始まる)。
- 利用可能なサポート・データ (SQLCA データ構造、追加のダンプ・ファイルまたはトラップ・ファイルの場所を指すポインターなど)。

以下の例はサンプル・ログ項目のヘッダー情報で、ログのすべての部分が示されています。

注: すべてのログ項目にこれらの部分がすべて含まれるとは限りません。

```
2006-02-15-19.33.37.630000 1 Instance:DB2 2 Node:000 3  
PID:940(db2syscs.exe) TID: 660 4 Appid:*LOCAL.DB2.020205091435 5  
recovery manager 6 sqlpresr 7 Probe:1 8 Database:SAMPLE 9  
ADM1530E 10 Crash recovery has been initiated. 11
```

凡例:

1. メッセージのタイム・スタンプ。
2. メッセージを生成したインスタンスの名前。
3. マルチパーティション・システムの場合、メッセージを生成したデータベース・パーティション。(非パーティション・データベースでは、この値は "000" です。)
4. メッセージが生成される原因になったプロセスのプロセス ID (PID) (プロセスの名前、スレッド ID (TID) が後に続きます)。
- 5.

プロセスが作動しているアプリケーションの識別番号。この例では、メッセージを生成したプロセスは、ID が *LOCAL.DB2.020205091435 であるアプリケーションのプロセスとして作動しています。

この値は **appl_id** モニター・エレメント・データと同じになります。この値を解釈する方法の詳細については、**appl_id** モニター・エレメントの資料を参照してください。

特定のアプリケーション ID の詳細情報を識別するには、以下のいずれかを行ってください。

- DB2 サーバー上で **LIST APPLICATIONS** コマンドを使用するか、DB2 Connect ゲートウェイ上で **LIST DCS APPLICATIONS** を使用して、アプリケーション ID のリストを表示します。このリストから、エラーが起きたクライアントに関する情報を判別できます (ノード名、TCP/IP アドレスなど)。

- **GET SNAPSHOT FOR APPLICATION** コマンドを使用して、アプリケーション ID のリストを表示します。
- 6. メッセージを書き込んでいる DB2 コンポーネント。
db2AdminMsgWrite API を使ってユーザー・アプリケーションによって書き込まれたメッセージの場合、このコンポーネントは「User Application」となります。
- 7. メッセージを出している関数の名前。この関数は、メッセージを書き込んでいる DB2 コンポーネントの中で機能します。
db2AdminMsgWrite API を使ってユーザー・アプリケーションによって書き込まれたメッセージの場合、この関数は「User Function」となります。
- 8. ユニークな内部 ID。この番号を使って、DB2 お客様サポートや開発部門は DB2 ソース・コード内のメッセージを報告した場所を見つけることができます。
- 9. エラーが発生したデータベース。
- 10. エラーのタイプと番号を 16 進数コードで示すメッセージ (存在する場合)。
- 11. ログに記録されたイベントを説明するメッセージ・テキスト (存在する場合)。

管理通知ログ・ファイルのエラー・キャプチャー・レベルの設定

このタスクでは、管理通知ログ・ファイルのエラー・キャプチャー・レベルを設定する方法について説明します。

このタスクについて

DB2 がどのような情報を管理通知ログに記録するかは、**notifylevel** 設定によって決まります。

手順

- 現在の設定値を確認するには、**GET DBM CFG** コマンドを発行します。

以下のような変数を見つけてください。

```
Notify Level          (NOTIFYLEVEL) = 3
```

- 設定値を変更するには、**UPDATE DBM CFG** コマンドを使用します。例えば、

```
DB2 UPDATE DBM CFG USING NOTIFYLEVEL X
```

ここで、X はご希望の通知レベルです。

DB2 診断 (db2diag) ログ・ファイル

DB2 診断 **db2diag** ログ・ファイルは、主に IBM ソフトウェア・サポートがトラブルシューティングの目的で使用するためのものです。管理通知ログは、主にデータベースおよびシステム管理者がトラブルシューティングに使用するためのものです。管理通知ログ・メッセージも、標準化されたメッセージ・フォーマットを使用して **db2diag** ログ・ファイルに記録されます。

概説

db2diag ログ・ファイルには DB2 診断および管理通知メッセージの両方が記録されるため、**db2diag** ログ・ファイルはしばしば、データベースの操作に関する情報を取得するための最初の場所になります。診断ログ・ファイルの内容の解釈に役立つ情報は、『関連リンク』セクションでリストされているトピックに記載されています。トラブルシューティングを行っても問題を解決できず、支援が必要と感じる場合は、IBM ソフトウェア・サポートに連絡できます (詳細については、『IBM ソフトウェア・サポートへの連絡』を参照してください)。IBM ソフトウェア・サポートに送信するよう要求される関連診断情報を収集する際は、他の関連ログ、ストレージ・ダンプ、およびトレースを含む他の情報ソースとともに **db2diag** ログ・ファイルを含めることができます。

db2diag ログ・ファイルには、以下の 2 種類の形式があります。

単一診断ログ・ファイル

サイズが無限に増大する、**db2diag.log** という名前のアクティブな診断ログ・ファイル。これはデフォルト書式であり、**diagsize** データベース・マネージャー構成パラメーターの値が 0 (このパラメーターのデフォルト値は 0 です) である場合に存在します。

循環診断ログ・ファイル

単一のアクティブ・ログ・ファイル (名前は **db2diag.N.log** であり、*N* は 0 から継続的に増大するファイル名標識です)。ただし、**diagpath** 構成パラメーターで定義された場所に一連の診断ログ・ファイルが存在することがあります。その場合、それぞれのログ・ファイルは増大して制限サイズに達した時点で閉じられ、ファイル名の標識が増加された新しいログ・ファイル (**db2diag.N+1.log**) が作成されて開かれます。これは、**diagsize** データベース・マネージャー構成パラメーターがゼロ以外の値である場合に存在します。

diagsize データベース・マネージャー構成パラメーターを適切に設定することにより、これらの 2 種類の形式のうちどちらがシステム上に存在するかを選択できます。

構成

db2diag ログ・ファイルは、以下のデータベース・マネージャー構成パラメーターを設定することにより、そのサイズ、場所、および記録される診断エラーのタイプを構成できます。

diagsize

diagsize の値は、採用される診断ログ・ファイルの形式を決定します。値が 0 の場合は、単一診断ログ・ファイルが採用されます。値が 0 でない場合は、循環診断ログ・ファイルが採用されるとともに、その非ゼロ値によってすべての循環診断ログ・ファイルとすべての循環管理通知ログ・ファイルの合計サイズが指定されます。**diagsize** パラメーターの新規値を有効にするには、インスタンスを再始動する必要があります。詳細については、『**diagsize** - 診断ログ・ファイル・サイズ構成パラメーター』トピックを参照してください。

diagpath

診断情報は、**db2diag** ログ・ファイル内の **diagpath** 構成パラメーターで定義された場所書き込まれるように指定できます。詳細については、『diagpath - 診断データ・ディレクトリー・パス構成パラメーター』トピックを参照してください。

alt_diagpath

alt_diagpath データベース・マネージャー構成パラメーターは、診断情報を保管するための代替診断データ・ディレクトリー・パスを提供します。**diagpath** によって指定されるパスにデータベース・マネージャーが書き込みを行えない場合、**alt_diagpath** で指定されるパスを使用して診断情報が保管されます。

diaglevel

db2diag ログ・ファイルに書き込まれる診断エラーのタイプは、**diaglevel** 構成パラメーターによって指定できます。詳細については、『diaglevel - 診断エラー・キャプチャー・レベル構成パラメーター』トピックを参照してください。

注: **diagsize** 構成パラメーターがゼロ以外の値に設定された場合、その値は診断データ・ディレクトリーの中に含まれるすべての循環管理通知ログ・ファイルとすべての循環診断ログ・ファイルを合わせた合計サイズを指定します。例えば、データベース・パーティションが 4 つあるシステムで **diagsize** が 1 GB に設定されている場合、通知ログと診断ログを合わせた最大合計サイズは、4 GB (4 x 1 GB) になります。

診断ログ・ファイル項目の解釈

db2diag ログ・ファイル分析ツール (**db2diag**) は、**db2diag** ログ・ファイルのフィルター処理とフォーマットに使用します。標準化されたメッセージ・フォーマットで管理通知ログ・メッセージの追加部分が **db2diag** ログ・ファイルに記録された場合は、データベースで何が起こったのかを理解するために、**db2diag** ログ・ファイルを最初に表示することをお勧めします。

db2diag を使用する代わりに、テキスト・エディターを使用して、問題が発生した可能性のあるマシンの診断ログ・ファイルを表示できます。最新のイベントは、ファイルの一番下に記録されています。

注: 管理通知 (*instance_name.nfy*) および診断 (**db2diag**) ログは、単一ログ・ファイルとして継続的に増加します。**diagsize** データベース・マネージャーの構成パラメーターがゼロ以外の値に設定されている場合、管理通知および **db2diag** ログ・ファイルの両方は、**diagsize** 構成パラメーターの値により決定される限定された合計サイズを持つ、連続した循環ログ・ファイル (*instance_name.N.nfy* and *db2diag.N.log*) になります。

以下の例はサンプル・ログ項目のヘッダー情報で、ログのすべての部分が示されています。

注: すべてのログ項目にこれらの部分がすべて含まれるとは限りません。すべての **db2diag** ログ・ファイルレコードに必ず含まれるのは、最初のいくつかのフィールド (タイム・スタンプから TID まで) および FUNCTION だけです。

```
2007-05-18-14.20.46.973000-240 1 I27204F655 2 LEVEL: Info 3
PID : 3228 4 TID : 8796 5 PROC : db2syscs.exe 6
INSTANCE: DB2MPP 7 NODE : 002 8 DB : WIN3DB1 9
APPHDL : 0-51 10 APPID: 9.26.54.62.45837.070518182042 11
AUTHID : UDBADM 12
EDUID : 8796 13 EDUNAME: db2agntp 14 (WIN3DB1) 2
FUNCTION: 15 DB2 UDB, data management, sqldInitDBCBC, probe:4820
DATA #1 : 16 String, 26 bytes
Setting ADC Threshold to:
DATA #2 : unsigned integer, 8 bytes
1048576
```

凡例:

1. メッセージのタイム・スタンプとタイム・ゾーン。

注: **db2diag** ログ・ファイルのタイム・スタンプには、タイム・ゾーンが含まれます。例えば、2006-02-13-14.34.35.965000-300 の場合、"-300" は UTC (協定世界時、以前は GMT と呼ばれていた) とアプリケーション・サーバーの現地時間との差異です (単位は分)。つまり -300 とは UTC マイナス 5 時間という意味で、米国の東部標準時 (EST) がこれに該当します。

2. レコード ID フィールド。**db2diag** ログ・ファイルのレコード ID は、DB2 診断ログが作成されたプラットフォームで現在のメッセージが記録されている場所のファイル・オフセット (例えば「27204」) とメッセージの長さ (例えば「655」) を示します。
3. エラー・メッセージに関連した診断レベル。例えば、Info (通知)、Warning (警告)、Error (エラー)、Severe (重大)、または Event (イベント)。
4. プロセス ID
5. スレッド ID
6. プロセス名
7. メッセージを生成したインスタンスの名前。
8. マルチパーティション・システムの場合、メッセージを生成したデータベース・パーティション。(非パーティション・データベースでは、この値は "000" です。)
9. データベース名
10. アプリケーション・ハンドル。この値は、**db2pd** 出力ファイルおよびロック・ダンプ・ファイルの値と同じになります。コーディネーター・パーティション番号の後にダッシュ (-)、その後にコーディネーター索引番号が示されます。
11. プロセスが作動しているアプリケーションの識別番号。この例では、メッセージを生成したプロセスは、ID 9.26.54.62.45837.070518182042 のアプリケーションのために作動しています。

TCP/IP によって生成されるアプリケーション ID は、3 つのセクションから構成されます。

1. **IP アドレス:** 最大 8 個の 16 進文字を使用する 32 ビットの数値として表されます。
2. **ポート番号:** 4 個の 16 進文字として表されます。
3. このアプリケーションのインスタンスを表す**ユニーク ID**。

注: 16 進数の IP アドレスまたはポート番号が 0 から 9 で始まる場合、それらは G から P に変更されます。例えば、"0" は "G" に、"1" は "H" にとりかかるといように、それぞれマップされます。IP アドレス AC10150C.NA04.006D07064947 は次のように解釈されます: IP アドレスは AC10150C のままで、172.16.21.12 に変換されます。ポート番号は NA04 です。最初の文字 "N" は "7" にマップされます。したがって、16 進形式のポート番号は 7A04 となり、これは 10 進形式で 31236 に変換されます。

この値は *appl_id* モニター・エレメント・データと同じになります。この値を解釈する方法の詳細については、*appl_id* モニター・エレメントの資料を参照してください。

特定のアプリケーション ID の詳細情報を識別するには、以下のいずれかを行ってください。

- DB2 サーバー上で **LIST APPLICATIONS** コマンドを使用するか、DB2 Connect ゲートウェイ上で **LIST DCS APPLICATIONS** を使用して、アプリケーション ID のリストを表示します。このリストから、エラーが起きたクライアントに関する情報を判別できます (データベース・パーティション名、TCP/IP アドレスなど)。
- **GET SNAPSHOT FOR APPLICATION** コマンドを使用して、アプリケーション ID のリストを表示します。
- **db2pd -applications -db <dbname>** コマンドを使用します。

- 12 許可 ID。
- 13 エンジン・ディスパッチ可能単位 ID。
- 14 エンジン・ディスパッチ可能単位の名前。
15. メッセージを書き込んだ製品の製品名 (「DB2」)、コンポーネント名 (「data management」)、関数名 (「sqlInitDBCB」)、関数内のプローブ点 (「4820」)。
16. 呼び出した関数から返された情報。複数のデータ・フィールドが返される可能性があります。

サンプル **db2diag** ログ・ファイルの項目をすでに見ましたので、可能なすべてのフィールドのリストをご覧ください。

```
<timestamp><timezone>          <recordID>          LEVEL: <level> (<source>)
PID      : <pid>                TID   : <tid>        PROC  : <procName>
INSTANCE: <instance>          NODE  : <node>       DB    : <database>
APPHDL   : <appHandle>        APPID : <appID>
AUTHID   : <authID>
EDUID    : <eduID>            EDUNAME: <engine dispatchable unit name>
FUNCTION: <prodName>, <compName>, <funcName>, probe:<probeNum>
MESSAGE  : <messageID> <msgText>
CALLED   : <prodName>, <compName>, <funcName>  OSERR: <errorName> (<errno>)
RETCODE  : <type>=<retCode> <errorDesc>
ARG #N   : <typeTitle>, <typeName>, <size> bytes
... argument ...
DATA #N  : <typeTitle>, <typeName>, <size> bytes
... data ...
```

上記の例で説明されなかったフィールドは以下のとおりです。

•

<source> は、ログに記録されたエラーの発生源を示します。(サンプルの最初の行の最後にあります。)可能な値は以下のとおりです。

- origin - メッセージは、エラーが発生した場所 (開始点) である関数によって記録されました
- OS - エラーはオペレーティング・システムによって生成されました
- received - エラーは別のプロセス (クライアント/サーバー) から受信されました
- sent - エラーは別のプロセス (クライアント/サーバー) に送信されました

MESSAGE は記録されたメッセージです。以下のもので構成されます。

- <messageID> - メッセージ番号 (例えば ECF=0x9000004A または DIA8604C)
- <msgText> - エラーの説明

CALLED フィールドが存在する場合、<msgText> は、CALLED 関数から戻されたエラーがメッセージを記録した関数 (FUNCTION フィールドで識別される) に与えた影響を表します。

CALLED はエラーを戻した関数です。以下のもので構成されます。

- <prodName> - 製品名 ("OS", "DB2", "DB2 Tools", または "DB2 Common")
- <compName> - コンポーネント名 (システム呼び出しの場合は '-')
- <funcName> - 呼び出された関数名

- OSERR は、CALLED システム呼び出しによって戻されたオペレーティング・システム・エラーです。(CALLED と同じ行の最後にあります。)以下のもので構成されます。

- <errorName> - システム固有のエラー名
- <errno> - オペレーティング・システムのエラー番号

- ARG セクションは、エラーを戻した関数呼び出しの引数をリストします。以下のもので構成されます。

- <N> - 関数の呼び出しにおける引数の位置
- <typeTitle> - N 番目の引数のタイプ名に関連したラベル
- <typeName> - ログに記録された引数のタイプ名
- <size> - ログに記録された引数のサイズ

- DATA には、ロギング機能によってダンプされた追加のデータが示されます。以下のもので構成されます。

- <N> - ダンプ対象のデータ・オブジェクトの順序番号
- <typeTitle> - ダンプ対象のデータのラベル
- <typeName> - ログに記録されたデータ・フィールドのタイプ名 (例えば PD_TYPE_UINT32、PD_TYPE_STRING)
- <size> - データ・オブジェクトのサイズ

db2diag ログ・ファイルの情報レコードの解釈

db2diag ログ・ファイル内の最初のメッセージとして、常に情報レコードが記録されます。

以下は、情報レコードの例です。

```
2006-02-09-18.07.31.059000-300 I1H917          LEVEL: Event
PID      : 3140                               TID   : 2864          PROC  : db2start.exe
INSTANCE: DB2                                NODE  : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START    : New Diagnostic Log file
DATA #1 : Build Level, 124 bytes
Instance "DB2" uses "32" bits and DB2 code release "SQL09010"
with level identifier "01010107".
Informational tokens are "DB2 v9.1.0.190", "s060121", "", Fix Pack "0".
DATA #2 : System Info, 1564 bytes
System: WIN32_NT MYSRVR Service Pack 2 5.1 x86 Family 15, model 2, stepping 4
CPU: total:1 online:1 Cores per socket:1 Threading degree per core:1
Physical Memory(MB): total:1024 free:617 available:617
Virtual Memory(MB): total:2462 free:2830
Swap Memory(MB): total:1438 free:2213
Information in this record is only valid at the time when this file was created
(see this record's time stamp)
```

情報レコードは、それぞれの論理パーティション上の **db2start** ごとに出力されます。このため、論理パーティションごとに 1 つ、合わせて複数の情報レコードが存在します。情報レコードには各パーティションごとに異なるメモリー値が含まれるため、この情報が役立つことがあります。

診断ログ・ファイルのエラー・キャプチャー・レベルの設定

DB2 診断 (**db2diag**) ログ・ファイルは、DB2 データベース・システムによってログ記録されるテキスト情報を含むファイルです。この情報はトラブルシューティングに使用され、そのほとんどは主として IBM ソフトウェア・サポートのためのものです。

このタスクについて

db2diag ログ・ファイルに記録される診断エラーのタイプは、**diaglevel** データベース・マネージャーの構成パラメーター設定によって決定されます。

手順

- 現在の設定値を確認するには、コマンド **GET DBM CFG** を発行します。

以下のような変数を見つけてください。

```
Diagnostic error capture level          (DIAGLEVEL) = 3
```

- 値を動的に変更するには、**UPDATE DBM CFG** コマンドを使用します。

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
```

例えば、

```
DB2 UPDATE DBM CFG USING DIAGLEVEL X
```

ここで、*X* はご希望の通知レベルです。再現可能な問題を診断する場合には、IBM ソフトウェア・サポート担当員はトラブルシューティング実行時に `diaglevel 4` を使用するよう勧めるでしょう。

DB2 データベースの診断と OS の診断の結合

メモリー、スワップ・ファイル、CPU、ディスク・ストレージ、その他のリソースに関連した問題を診断するには、所定のオペレーティング・システムがこれらのリソースをどのように管理するかについて十分に理解しておく必要があります。リソース関連の問題を定義するには、最低でも、そのリソースがどれだけ存在するか、ユーザーごとにどのようなリソース制限があるかを知っておく必要があります。(関係のある制限は、DB2 インスタンス所有者のユーザー ID に対するものであるのが一般的です。)

以下は、入手する必要がある重要な構成情報の一部です。

- オペレーティング・システムのパッチ・レベル、インストール済みソフトウェア、およびアップグレード履歴
- CPU の数
- RAM の容量
- スワップおよびファイル・キャッシュの設定
- ユーザー・データとファイル・リソースの制限、およびユーザーごとの処理限界
- IPC リソースの制限 (メッセージ・キュー、共有メモリー・セグメント、セマフォ)
- ディスク・ストレージのタイプ
- マシンがその他の用途に使用されているかどうか。ご使用の DB2 サーバーは、リソースを獲得するために競合しますか。
- 認証が行われる場所

ほとんどのプラットフォームには、リソースの情報を取得するための簡単なコマンドがあります。しかし、こうした情報を手動で取得する必要があることはまれです。`db2support` ユーティリティーが、このデータ、およびその他多くのデータを収集するからです。`db2support` (オプション `-s` および `-m` を指定した場合) によって生成される `detailed_system_info.html` ファイルには、この情報を収集するために使用される多数のオペレーティング・システム・コマンドの構文が記載されています。

以下の演習の目的は、各種の DB2 診断ファイルからシステム構成とユーザー環境の情報を発見するのを助けることです。最初の演習では、`db2support` ユーティリティーの実行に関係するステップを示します。以下の演習では、トラップ・ファイルを取り上げます。トラップ・ファイルは、ユーザー環境とリソース制限の理解に役立つ、DB2 サーバーに関するより多くのデータを生成します。

演習 1: `db2support` コマンドの実行

1. `db2start` コマンドを使って DB2 インスタンスを開始します。
2. `db2support` からの出力を格納するためのディレクトリーを作成します (既に `SAMPLE` データベースがあるものと仮定しています)。
3. そのディレクトリーに移動して、以下を実行します。

```
db2support <directory> -d sample -s -m
```

4. コンソール出力を確認します。特に、収集されている情報の種類に注目します。

以下のような出力が表示されるはずです (Windows で実行した場合)。

```
...
Collecting "System files"
  "db2cache.prf"
  "db2cos9402136.0"
  "db2cos9402840.0"
  "db2dbamr.prf"
  "db2diag.bak"
  "db2eventlog.000"
  "db2misc.prf"
  "db2nodes.cfg"
  "db2profile.bat"
  "db2system"
  "db2tools.prf"
  "HealthRulesV82.reg"
  "db2dasdiag.log"
...
Collecting "Detailed operating system and hardware information"
Collecting "System resource info (disk, CPU, memory)"
Collecting "Operating system and level"
Collecting "JDK Level"
Collecting "DB2 Release Info"
Collecting "DB2 install path info"
Collecting "Registry info"
...
Creating final output archive
  "db2support.html"
  "db2_sqllib_directory.txt"
  "detailed_system_info.html"
  "db2supp_system.zip"
  "dbm_detailed.supp_cfg"
  "db2diag.log"
db2support is now complete.
An archive file has been produced: "db2support.zip"
```

5. 次に、Web ブラウザーを使って `detailed_system_info.html` ファイルを表示します。ご使用の各システムで、以下の情報を確認してください。

- CPU の数
- オペレーティング・システムのレベル
- ユーザー環境
- ユーザー・リソースの制限 (UNIX `ulimit` コマンド)

演習 2: DB2 トラップ・ファイルから環境情報を見付ける

1. DB2 インスタンスが開始されていることを確認し、その後以下を発行します。
`db2pd -stack all`

呼び出しスタックは、診断ディレクトリー (データベース・マネージャー構成パラメーター `diagpath` により定義されている) 内のファイルに置かれます。

2. いずれかのトラップ・ファイルで、以下を見付けます。
 - DB2 のコード・レベル
 - Data seg top (要求された専用アドレス・スペースの最大サイズ)
 - Cur data size (専用アドレス・スペースの上限)

- Cur core size (コア・ファイル・サイズの上限)
- Signal Handlers (この情報は、必ずしもすべてのトラップ・ファイルに表示されるとは限らない)
- Environment variables (この情報は、必ずしもすべてのトラップ・ファイルに表示されるとは限らない)
- map output (ロードされたライブラリー)

Windows からのトラップ・ファイルの例 (短縮版)

```

...
<DB2TrapFile version="1.0">
<Trap>
<Header>
DB2 build information: DB2 v9.1.0.190 s060121 SQL09010
timestamp: 2006-02-17-14.03.43.846000
uname: S:Windows
comment:
process id: 940
thread id: 3592
</Header>
<SystemInformation>
Number of Processors: 1
Processor Type: x86 Family 15 Model 2 Stepping 4
OS Version: Microsoft Windows XP, Service Pack 2 (5.1)
Current Build: 2600
</SystemInformation>
<MemoryInformation>
<Usage>
Physical Memory:      1023 total,      568 free.
Virtual Memory :      2047 total,      1882 free.
Paging File   :      2461 total,      2011 free.
Ext. Virtual   :           0 free.
</Usage>
</MemoryInformation>
<EnvironmentVariables>
<![CDATA[
[e] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2_EXTSECURITY=YES
[g] DB2SYSTEM=MYSRVR
[g] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2INSTDEF=DB2
[g] DB2ADMINSERVER=DB2DAS00
]]></EnvironmentVariables>

```

DB2 とシステム・イベントまたはエラーとの相関

システム・メッセージとエラー・ログは見過ごされることが多くあります。問題の定義と調査の最初の段階で時間を取って 1 つの簡単な作業さえ行っておけば、問題解決までに要する時間を、時間単位、日単位、さらには週単位で減らすことができます。その作業とは、さまざまなログの項目同士を比較して、時間と、参照しているリソースの両面で関連がありそうな項目がないか注意することです。

必ずしも常に問題診断に関係があるわけではありませんが、システム・ログで最良の手掛かりが簡単に見つかることは少なくありません。報告されたシステム問題と DB2 エラーとの相関を見出すことができれば、多くの場合、DB2 の症状の直接的な原因をすでに識別できていることになります。分かりやすい例としては、ディスク・エラー、ネットワーク・エラー、およびハードウェア・エラーがあります。分かりにくいのは、別のマシンで報告される問題です。例えば、ドメイン・コントローラーが接続時間や認証に影響を与える可能性がある場合などです。

特に、新しいシステムで問題が報告される場合には、システム・ログを調べて安定度を分析できます。共通アプリケーションで断続的にトラップが生じているなら、それは基盤となるハードウェアに障害があることを示すしるしかもしれません。

以下に、システム・ログが提供する他の情報をいくつか挙げます。

- システムのリポートなどの重大なイベント
- システム上で生じた DB2 のトラップ (および障害が起こった他のソフトウェアからのエラー、トラップ、または例外) の時間的順序
- カーネルのパニック、out-of-filesystem-space、および out-of-swap-space などのエラー (これが原因でシステムが新しいプロセスを作成またはフォークできないことがあります)

システム・ログを調べることで、**db2diag** ログ・ファイル内のクラッシュ項目を、懸念される要因から除外できることがあります。DB2 管理通知または DB2 診断ログの中で、先にエラーがないのにクラッシュ項目がある場合、DB2 クラッシュ・リカバリーはシステムのシャットダウンの結果生じたものと考えられます。

情報を相関させるというこの原則は、あらゆるソースからのログ、および識別可能なあらゆるユーザーの症状にも適用できます。例えば、別のアプリケーションのログから相関関係にある項目を識別して文書化することは、たとえそれを十分に解釈できないとしても、大いに役立つ場合があります。

この情報を加えることで、サーバー、および問題発生時に生じていたさまざまなイベントすべてについての十分な理解が得られます。

db2cos (コールアウト・スクリプト) 出力ファイル

データベース・マネージャーが、パニック、トラップ、セグメンテーション違反、例外のために処理を続けられなくなると、デフォルトで **db2cos** スクリプトが呼び出されます。それぞれのデフォルトの **db2cos** スクリプトは、**db2pd** コマンドを呼び出してアンラッチ方式で情報を収集します。

db2cos スクリプトの名前は、**db2cos_hang**、**db2cos_trap** などのような形になります。それぞれのスクリプトは同じように動作します。例外は **db2cos_hang** で、これは **db2fodc** ツールから呼び出されます。

デフォルトの **db2cos** スクリプトは、bin ディレクトリーにあります。UNIX オペレーティング・システムの場合、このディレクトリーは読み取り専用です。**db2cos** スクリプト・ファイルは、adm ディレクトリーにコピーし、必要に応じてその場所を変更することも可能です。**db2cos** スクリプトが adm ディレクトリーにあれば、そのスクリプトが実行されます。そうでない場合は、bin ディレクトリーにあるスクリプトが実行されます。

複数のパーティション構成では、トラップを検出したパーティションのトラッピングするエージェントについてのみ、スクリプトが呼び出されます。他のパーティションから情報を集める必要がある場合は **db2_a11** コマンドを使用するように **db2cos** スクリプトを更新し、すべてのパーティションが同じマシンにある場合は **db2pd** コマンドに **-alldbpartitionnums** オプションを指定します。

db2cos の呼び出しをトリガーするシグナルのタイプも、**db2pdcfg -cos** コマンドによって構成できます。デフォルトの構成は、パニックやトラップ発生時には **db2cos** スクリプトを実行します。ただし、生成されたシグナルはデフォルトでは **db2cos** スクリプトを起動しません。

パニック、トラップ、セグメンテーション違反、または例外が起きたときのイベント順序は以下のとおりです。

1. トラップ・ファイルが作成される
2. シグナル・ハンドラーが呼び出される
3. **db2cos** スクリプトが呼び出される (**db2cos** 設定が有効な場合)
4. 項目が管理通知ログに記録される
5. 項目が **db2diag** ログ・ファイルに記録される

db2cos スクリプトの **db2pd** コマンドによって集められるデフォルト情報には、オペレーティング・システム、インストールされている DB2 製品のバージョンとサービス・レベル、データベース・マネージャー、およびデータベース構成についての詳細に加え、エージェント、メモリー・プール、メモリー・セット、メモリー・ブロック、アプリケーション、ユーティリティ、トランザクション、バッファーク・プール、ロック、トランザクション・ログ、表スペース、およびコンテナの状態についての情報が含まれます。さらに、動的キャッシュ、静的キャッシュ、およびカタログ・キャッシュの状態、表と索引の統計、リカバリー状況についての情報や、再最適化された SQL ステートメントとアクティブなステートメントのリストも提供されます。さらに情報を収集する必要がある場合は、他のコマンドを追加して **db2cos** スクリプトを更新します。

デフォルトの **db2cos** スクリプトを呼び出すと、DIAGPATH データベース・マネージャー構成パラメーターで指定されているディレクトリーに出力ファイルが生成されます。それらのファイルには、XXX.YYY.ZZZ.cos.txt という名前が付けられます (XXX はプロセス ID (PID)、YYY はスレッド ID (TID)、ZZZ はデータベース・パーティション番号 (単一パーティション・データベースの場合は 000) です)。複数のスレッドがトラップされる場合は、それぞれのスレッドごとに、**db2cos** スクリプトが別々に呼び出されます。PID および TID の組み合わせが複数回発生する場合、データはファイルに追加されます。タイム・スタンプが表記されるので、出力の反復を区別することができます。

db2cos 出力ファイルには、**db2cos** スクリプトに指定されているコマンドに応じて、さまざまな情報が組み込まれます。デフォルトのスクリプトを変更しなければ、以下のような項目が表示されます (詳細な **db2pd** 出力が続きます)。

```
2005-10-14-10.56.21.523659
PID       : 782348           TID  : 1           PROC  : db2cos
INSTANCE: db2inst1        NODE  : 0           DB    : SAMPLE
APPHDL   :                 APPID: *LOCAL.db2inst1.051014155507
FUNCTION: oper system services, sqloEDUCodeTrapHandler, probe:999
EVENT    : Invoking /home/db2inst1/sqlllib/bin/db2cos from
oper system services sqloEDUCodeTrapHandler
Trap Caught
```

```
Instance db2inst1 uses 64 bits and DB2 code release SQL09010
```

```
...
Operating System Information:
```

```
OSName:   AIX
```

```
NodeName: n1
Version: 5
Release: 2
Machine: 000966594C00
```

...

db2diag ログ・ファイルにはオカレンスに関連した項目も含まれます。例えば、

```
2005-10-14-10.42.17.149512-300 I19441A349 LEVEL: Event
PID : 782348 TID : 1 PROC : db2sysc
INSTANCE: db2inst1 NODE : 000
FUNCTION: DB2 UDB, trace services, pdInvokeCalloutScript, probe:10
START : Invoking /home/db2inst1/sqllib/bin/db2cos from oper system
services sqloEDUCodeTrapHandler

2005-10-14-10.42.23.173872-300 I19791A310 LEVEL: Event
PID : 782348 TID : 1 PROC : db2sysc
INSTANCE: db2inst1 NODE : 000
FUNCTION: DB2 UDB, trace services, pdInvokeCalloutScript, probe:20
STOP : Completed invoking /home/db2inst1/sqllib/bin/db2cos

2005-10-14-10.42.23.519227-300 E20102A509 LEVEL: Severe
PID : 782348 TID : 1 PROC : db2sysc
INSTANCE: db2inst1 NODE : 000
FUNCTION: DB2 UDB, oper system services, sqloEDUCodeTrapHandler, probe:10
MESSAGE : ADM0503C An unexpected internal processing error has occurred. ALL
DB2 PROCESSES ASSOCIATED WITH THIS INSTANCE HAVE BEEN SHUTDOWN.
診断情報が記録されました。Contact IBM Support for
further assistance.

2005-10-14-10.42.23.520111-300 E20612A642 LEVEL: Severe
PID : 782348 TID : 1 PROC : db2sysc
INSTANCE: db2inst1 NODE : 000
FUNCTION: DB2 UDB, oper system services, sqloEDUCodeTrapHandler, probe:20
DATA #1 : Signal Number Recieved, 4 bytes
11
DATA #2 : Siginfo, 64 bytes
0x0FFFFFFFFFD5C0 : 0000 000B 0000 0000 0000 0009 0000 0000 .....
0x0FFFFFFFFFD5D0 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0FFFFFFFFFD5E0 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0FFFFFFFFFD5F0 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

ダンプ・ファイル

ダンプ・ファイルは、エラー発生時に問題を診断するのに役立つと思われる追加情報（内部制御ブロックなど）がある場合に作成されます。ダンプ・ファイルに書き込まれるデータ項目はそれぞれ、問題の判別に役立つようにタイム・スタンプが関連付けられています。ダンプ・ファイルはバイナリー・フォーマットで、IBM ソフトウェア・サポート担当者用です。

ダンプ・ファイルが作成または追加されると、書き込まれたデータの時刻およびタイプを示した項目が **db2diag** ログ・ファイルに作成されます。これらの **db2diag** ログ項目は、以下の例のようになっています。

```
2007-05-18-12.28.11.277956-240 I24861950A192 LEVEL: Severe
PID:1056930 TID:225448 NODE:000 Title: dynamic memory buffer
Dump File:/home/svtdbm5/sqllib/db2dump/1056930.225448.000.dump.bin
```

注: パーティション・データベース環境では、ファイル拡張子でパーティション番号が識別されます。例えば、以下の項目は、ダンプ・ファイルがパーティション 10 で実行されている DB2 プロセスによって作成されたことを示します。

Dump File: /home/db2/sqllib/db2dump/6881492.2.010.dump.bin

First Occurrence Data Capture 情報

First Occurrence Data Capture (FODC) は、問題の発生時に DB2 インスタンス、ホスト、またはメンバーに関する診断情報を収集します。FODC は、問題の発生時に診断情報を収集できるので、診断情報を得るために問題を再現する必要が少なくなります。

FODC は、問題に気付いたときに **db2fodc** コマンドを使用して手動で呼び出すこともできれば、事前定義のシナリオまたは症状が検出されるたびに自動的に呼び出すこともできます。診断情報の収集後、この情報を使用すると、問題の原因になっている可能性があるものを判別するのに役立ちます。問題の原因を自分で判別できる場合もあれば、IBM サポート担当員が関与する必要が生じる場合もあります。

db2fodc コマンドの実行が完了すると、**db2support** ツールを実行して結果の診断ファイルを集集し、IBM サポートに送信するための FODC パッケージを作成する必要があります。**db2support** コマンドは、見つかったすべての FODC パッケージ・ディレクトリーか **-fodcpath** パラメーターを使用して指定したすべての FODC パッケージ・ディレクトリーの内容を集集します。診断情報について IBM サポートからさらに情報を要求されないようにするためです。

一般的な障害問題に基づく診断情報の収集

インスタンス、ホスト、またはメンバーに影響する問題が発生すると、First Occurrence Data Collection (FODC) パッケージ内に診断情報を自動的に集めることができます。手動で FODC パッケージ内に情報を収集することもできます。

診断情報の自動収集

データベース・マネージャーは、自動 First Occurrence Data Capture (FODC) のために **db2fodc** コマンドを呼び出し、次に DB2 コールアウト・スクリプト (COS) の 1 つを呼び出します。

DB2 診断ログと他のトラブルシューティング・ファイルに障害を関連付けるために、管理通知および **db2diag** ログ・ファイルの両方に診断メッセージが書き込まれます。FODC パッケージのディレクトリー名には、接頭部 FODC_、障害のタイプ、FODC ディレクトリーの作成時のタイム・スタンプ、および問題の発生場所のメンバーまたはパーティション番号が含まれます。FODC パッケージの記述ファイルは、新しい FODC パッケージ・ディレクトリーに配置されます。

表 91. 自動 FODC タイプおよびパッケージ

パッケージ	説明	呼び出しタイプ	実行されるスクリプト
FODC_Trap_timestamp_memberNumber	インスタンス全体でのトラブルが発生した	自動	db2cos_trap (.bat)
FODC_Panic_timestamp_memberNumber	エンジンが矛盾を検出し、続行しないことになった	自動	db2cos_trap (.bat)

表 91. 自動 FODC タイプおよびパッケージ (続き)

パッケージ	説明	呼び出しタイプ	実行されるスクリプト
FODC_BadPage_timestamp_memberNumber	不正なページが検出された	自動	db2cos_datacorruption (.bat)
FODC_DBMarkedBad_timestamp_memberNumber	データベースがエラーのために不正とマークされた	自動	db2cos (.bat)
FODC_IndexError_timestamp_PID_EDUID_memberNumber	EDU 全体の索引エラーが発生した。	自動	db2cos_indexerror_short (.bat) または db2cos_indexerror_long (.bat)
FODC_Member_timestamp_memberNumber	メンバーまたはパーティションで障害が起きたか、強制終了シグナルを受信した	自動	db2cos_member (.bat)

診断情報の手動収集

問題が発生した疑いがある場合は、手動で **db2fodc** コマンドを使用します。診断データを収集できる問題のシナリオには、明らかなシステム・ハング、パフォーマンス問題、またはアップグレード操作かインスタンス作成が期待どおりに完了しなかったことが含まれます。**db2fodc** コマンドを手動で実行すると、新しい FODC パッケージ・ディレクトリーが作成されます。FODC パッケージのディレクトリー名には、接頭部 FODC_、問題のシナリオ、FODC ディレクトリーの作成時のタイム・スタンプ、および FODC の実行場所のメンバーまたはパーティション番号が含まれます。

表 92. 手動 FODC タイプおよびパッケージ

パッケージ	説明	呼び出しタイプ	実行されるスクリプト
FODC_Clp_timestamp_member	ユーザーが db2fodc -c1p を呼び出して、環境と構成に関連した情報を収集した。この情報は、インスタンスの作成に関連した問題のトラブルシューティングに使用します。	手動	db2cos_c1p スクリプト (.bat)
FODC_Connections_timestamp_member	ユーザーが db2fodc -connections を呼び出して、接続に関連した診断データを収集した。このデータは、実行状態またはコンパイル状態のアプリケーションの数が急に増加したり、新しいデータベース接続が拒否されたりするなどの問題の診断に使用します。	手動	db2cos_threshold スクリプト (.bat)

表 92. 手動 FODC タイプおよびパッケージ (続き)

パッケージ	説明	呼び出しタイプ	実行されるスクリプト
FODC_Cpu_timestamp_member	ユーザーが db2fodc -cpu を呼び出して、プロセッサに関連したパフォーマンス・データと診断データを収集した。このデータは、プロセッサの使用率が高い、実行中のプロセス数が多い、またはプロセッサの待ち時間が長いなどの問題の診断に使用します。	手動	db2cos_threshold スクリプト (.bat)
FODC_Hang_timestamp_memberList	ハング (またはパフォーマンスの重大な問題) のトラブルシューティングのためのデータを収集するためにユーザーが db2fodc -hang が呼び出した	手動	db2cos_hang (.bat)
FODC_Memory_timestamp_member	ユーザーが db2fodc -memory を呼び出して、メモリーに関連した診断データを収集した。このデータは、使用可能な空きメモリーがない、スワップ・スペースの使用率が高い、ページングが多すぎる、またはメモリー・リークが疑われるなどの問題の診断に使用します。	手動	db2cos_threshold スクリプト (.bat)
FODC_Perf_timestamp_memberList	パフォーマンスのトラブルシューティングのためのデータを収集するためにユーザーが db2fodc -perf を呼び出した	手動	db2cos_perf (.bat)
FODC_Preupgrade_timestamp_member	インスタンスのアップグレードや次バージョンのフィックスパックへの更新などの重大なアップグレードか更新の前に、ユーザーが db2fodc -preupgrade を呼び出して、パフォーマンスに関連した情報を収集した。	手動	db2cos_preupgrade (.bat)

表 92. 手動 FODC タイプおよびパッケージ (続き)

パッケージ	説明	呼び出しタイプ	実行されるスクリプト
Scripts located in FODC_IndexError_ timestamp_PID_EDUID_ memberList	<p>ユーザーは、db2fodc -indexerror FODC_IndexError_directory [basic full] (デフォルトは basic) を発行して、スクリプト内の db2dart コマンドを呼び出すことができます。</p> <p>パーティション・データベース環境では、db2_a11 "<<+node#< db2fodc -indexerror FODC_IndexError_directory [basic full]" を使用します。 <i>node#</i> は、FODC_IndexError_directory ディレクトリー名の最後の番号です。絶対パスが必要となるのは、db2fodc -indexerror を db2_a11 コマンドと共に使用するときです。</p>	手動	db2cos_indexerror_long (.bat) または db2cos_indexerror_short (.bat)

First Occurrence Data Capture の構成

First Occurrence Data Capture 構成 (FODC) の動作は、FODC パッケージの保管に使用されるパスを含め、**DB2FODC** によって制御されます。レジストリー変数このレジストリー変数は、**db2set** コマンドを使用して永続的に設定したり、**db2pdcfg** コマンドを使用して動的に変更 (メモリー内のみ) したりできます。FODC 中に呼び出されるコールアウト・スクリプト (COS) を更新して、FODC の動作をカスタマイズすることもできます。

インスタンス内のパーティションまたはメンバーごとに独自の FODC 設定があり、パーティション・レベルやメンバー・レベルで FODC が行われる方法を制御できます。メンバー・レベルまたはパーティション・レベルとインスタンス・レベルの両方の FODC 設定がある場合は、メンバー・レベルまたはパーティション・レベルの設定がインスタンス・レベルの設定をオーバーライドします。手動の FODC の場合、**-fodcpath** パラメーターなどのコマンド行パラメーターを指定して設定をオーバーライドすることもできます。パーティション・データベース環境や **DB2 pureScale** データベース環境では、手動 FODC のメンバーまたはパーティションのリストを指定すると、最初に指定されたメンバーまたはパーティションの設定が使用されます。

db2set コマンドを使用して行われた永続設定は、インスタンスがリサイクルされるまで有効になりません。**db2pdcfg** コマンドを使用して行われた動的設定は、即時に有効になり、インスタンスがリサイクルされるまでメモリー内で有効の状態が保たれます。

FODC パッケージの処理方法の制御に役立つように、複数の *DB2FODC* レジストリー変数の設定が使用できるようになっていますが、プラットフォームによっては使用できない設定もあります。*DB2FODC* レジストリー変数を使用して、以下の動作を制御できます。

- 生成された FODC パッケージの格納場所 (*FODCPATH* 設定を使用)
- コア・ダンプ・ファイルが生成されるかどうか (*DUMPCORE* 設定を使用)
- コア・ダンプ・ファイルの大きさの上限 (*CORELIMIT* 設定を使用)
- 生成されたダンプ・ファイルの格納場所 (*DUMPDIR* 設定を使用)

データベース・マネージャーが、パニック、トラップ、セグメンテーション違反、例外のために処理を続けられなくなると、FODC はデフォルトで **db2cos** コールアウト・スクリプトを呼び出して診断情報を収集します。FODC 中に呼び出されるコールアウト・スクリプトの制御に役立つように、複数の *COS* パラメーター設定が使用できるようになっています。*DB2FODC* レジストリー変数の *COS* パラメーターを使用して、以下の動作を制御できます。

- データベース・マネージャーが処理を続行できない場合に **db2cos** スクリプトを呼び出すかどうか (*ON* 設定と *OFF* 設定を使用、デフォルトは *ON*)
- **db2cos** スクリプトが生成済みの出力ファイルのサイズを検査する頻度 (*COS_SLEEP* 設定を使用)
- **db2cos** スクリプトが終了するのを FODC が待つ時間 (*COS_TIMEOUT* 設定を使用)
- データベース・マネージャーのトラップ中に **db2cos** スクリプトが呼び出される頻度 (*COS_COUNT* 設定を使用)
- *SQLQ_SIG_DUMP* シグナルを受け取ったときに **db2cos** スクリプトを使用可能にするかどうか (*COS_SQLQ_SIG_DUMP* 設定を使用)

FODC パッケージ・ディレクトリーの設定 (FODCPATH)

FODC パッケージが作成される時、大量の診断データが生成される場合があります。そのような場合、格納するためのスペースが必要になると共に、システムのプロセッサ使用量がかかり必要になる可能性があります。FODC が診断データを送信するディレクトリー・パスを制御して、使用可能なフリー・スペースが十分にあるディレクトリー・パスを選択できます。

使用する FODC パスを判別するには、以下の順序で行います。

自動 FODC

FODCPATH レジストリー変数の設定

DB2FODC レジストリー変数の **FODCPATH** パラメーターは、メンバー・レベルまたはパーティション・レベルと、インスタンス・レベルで設定できます。この値を設定すると、FODC はパーティションまたはメンバーごとに **FODCPATH** パラメーターの設定を使用します。パーティション・レベルかメンバー・レベルの設定がない場合は、インスタンス・レベルの設定が使用されます。

FODC パスの設定なし

デフォルトでは、メンバー・レベルとインスタンス・レベルのどちら

らでも **FODCPATH** 設定を指定しないと、FODC は診断情報を現行の診断ディレクトリー・パス (**diagpath** または **alt_diagpath**) に送信します。

手動 FODC

db2fodc -fodcpath コマンド・パラメーター・オプション

db2fodc コマンドを手動で呼び出す場合には、コマンドと共に **-fodcpath** パラメーター・オプションを指定して、FODC パッケージ・ディレクトリーの作成場所を指示できます。有効なパス名を使用して **-fodcpath** パラメーターを指定すると、そのパス内に FODC パッケージ・ディレクトリーが作成されます。

FODCPATH レジストリー変数の設定

db2fodc コマンドに **-fodcpath** パラメーターを指定せず、パーティションまたはメンバーのリストを指定した場合は、**db2fodc** コマンドは、指定したリスト内の最初のパーティションまたはメンバーの **DB2FODC** レジストリー変数の **FODCPATH** パラメーター設定を使用します。この **FODCPATH** パラメーターの値が設定されていないと、**db2fodc** はインスタンス・レベルの **FODCPATH** 設定を使用します。**-fodcpath** パラメーターを指定せず、パーティションまたはメンバーのリストも指定しない場合は、**db2fodc** コマンドは、まず現行のパーティションまたはメンバーの **FODCPATH** パラメーター設定を使用しようとします。設定されていない場合は、インスタンス・レベルの設定を使用します。

FODC パスの設定なし

デフォルトでは、FODC パスを指定しないと、First Occurrence Data Capture は診断情報を現行の診断ディレクトリー・パス (**diagpath** または **alt_diagpath**) に送信します。

3 つのメンバーまたはパーティション (0、1、および 2) があるパーティション・データベース環境を想定します。以下の例では、**db2set** コマンドを使用して、3 つのパーティションまたはメンバーすべてで、インスタンス・レベルで FODC パスを永続的に設定する方法を示します。

```
db2set DB2FODC=FODCPATH=/home/hotel149/juntang/FODC
```

インスタンス・レベルの設定をオーバーライドして、メンバーごとにメンバー・レベルで FODC パス設定を永続的に実行することもできます。これらの設定を有効にするには、インスタンスをリサイクルしなければなりません。例えば、メンバー 0 の FODC パスを変更するには、以下のコマンドを発行します。

```
db2set DB2FODC=FODCPATH=/home/hotel149/juntang/FODC/FODC0 -i juntang 0
```

次に、メンバー 1 とメンバー 2 の FODC パスを動的に変更する場合には、以下の **db2pdcfg** コマンドを使用します。これらの設定は、即時に有効になり、インスタンスがリサイクルされるまでメモリー内に残ります。

```
db2pdcfg -fodc FODCPATH=/home/hotel149/juntang/FODC/FODC1 -member 1
```

```
db2pdcfg -fodc FODCPATH=/home/hotel149/juntang/FODC/FODC2 -member 2
```

システム内のメンバーまたはパーティションごとに現行の FODC 設定を知りたい場合には、**db2pdcfg -fodc -member all** コマンドを使用できます (この例では、出力は要約されており、FODC パス出力のみ示されています)。

```
Database Member 0  
FODC package path (FODCPATH)= /home/hotel49/juntang/FODC/FODC0/
```

```
Database Member 1  
FODC package path (FODCPATH)= /home/hotel49/juntang/FODC/FODC1/
```

```
Database Member 2  
FODC package path (FODCPATH)= /home/hotel49/juntang/FODC/FODC2/
```

データ収集のカスタマイズ

FODC 中に呼び出される DB2 コールアウト・スクリプトの TOOL OPTIONS セクションに定義されているパラメーターでも、**db2fodc -hang** および **db2fodc -perf** によるデータ収集の動作が制御されます。FODC 中に実行されるスクリプトを変更して、これらのパラメーターをカスタマイズできます。

UNIX システムでデータ収集をカスタマイズするには、`/bin/db2cos_symptom` に置かれているスクリプトを `/adm/db2cos_symptom` にコピーします。`symptom` は `hang` または `perf` のどちらかです。この新規ディレクトリーに置いてから、スクリプトを自由に変更します。Windows システムでは、デフォルト・スクリプト `%bin%db2cos_symptom.bat` を変更するだけです。UNIX システムの場合、**db2fodc** はまず `/adm/db2cos_symptom` にあるスクリプトを実行しようとし、それが見つからないと `/adm/db2cos_symptom` 内の元のスクリプトを実行します。Windows システムでは、必ずスクリプト `%bin%db2cos_symptom.bat` を実行します。

FODC の一部として収集されたデータ

First Occurrence Data Capture (FODC) の結果として、FODC パッケージのディレクトリーとサブディレクトリーが作成され、診断情報が収集されます。親パッケージ・ディレクトリー、サブディレクトリー、および収集されるファイルをまとめて FODC パッケージと呼びます。

FODC によって収集される診断情報が含まれるファイル

FODC は複数のソースから診断情報を収集します。FODC によってキャプチャーされる正確な診断情報は、検出された問題のタイプに応じて異なり、以下の情報が含まれる可能性があります。

管理通知ログ (*instance_name.nfy*)

- オペレーティング・システム: すべて
- デフォルト・ロケーション:
 - Linux および UNIX: **diagpath** データベース・マネージャー構成パラメーターで指定されるディレクトリーにあります。
 - Windows: 「イベント・ビューアー」ツールを使用します (「スタート」>「コントロール パネル」>「管理ツール」>「イベント ビューア」)
- インスタンスが作成されると自動的に作成されます。
- 有効なイベントが発生すると、DB2 は管理通知ログに情報を書き込みます。情報は、データベースおよびシステム管理者が使用するためのもので

す。このファイルに記録されるメッセージのタイプは、**notifylevel** 構成パラメーターによって決定されます。

注: diagsize データベース・マネージャーの構成パラメーターがゼロ以外の値に設定されている場合は、単一の管理通知ログ・ファイルの動作 (「instance_name.nfy」) は、循環ログの動作 (「instance_name.N.nfy」) に変更されます。

DB2 診断ログ (db2diag ログ・ファイル)

- オペレーティング・システム: すべて
- デフォルト・ロケーション: **diagpath** データベース・マネージャー構成パラメーターで示されるディレクトリーにあります。
- インスタンスが作成されると自動的に作成されます。
- このテキスト・ファイルには、インスタンスで発生するエラーおよび警告に関する診断情報が入ります。この情報は、トラブルシューティングに使用され、IBM ソフトウェア・サポートの技術者のためのものです。このファイルに記録されるメッセージのタイプは、**diaglevel** データベース・マネージャー構成パラメーターによって決定されます。

注: diagsize データベース・マネージャー構成パラメーターがゼロ以外の値に設定されている場合は、単一の診断ログ・ファイルの動作 (単一の db2diag.log ファイル) は、循環ログの動作 (db2diag.N.log) に変更されます。

DB2 Administration Server (DAS) 診断ログ (db2dasdiag.log)

- オペレーティング・システム: すべて
- デフォルト・ロケーション:
 - Linux および UNIX: DASHOME/das/dump にあります。DASHOME は、DAS 所有者のホーム・ディレクトリーです。
 - Windows: DAS ホーム・ディレクトリーの「dump」フォルダーにあります。例えば、C:\Program Files\IBM\SQLLIB\DB2DAS00\dump のようになります。
- DAS が作成されると自動的に作成されます。
- このテキスト・ファイルには、DAS で発生するエラーおよび警告に関する診断情報が入ります。

DB2 イベント・ログ (db2eventlog.xxx, xxx はデータベース・パーティション番号)

- オペレーティング・システム: すべて
- デフォルト・ロケーション: **diagpath** データベース・マネージャー構成パラメーターで指定されるディレクトリーにあります。
- インスタンスが作成されると自動的に作成されます。
- DB2 イベント・ログ・ファイルとは、データベース・マネージャーで生じるインフラストラクチャー・レベルのイベントの循環ログのことです。このファイルはサイズが一定で、インスタンスの実行時に記録される特定のイベントに関する循環バッファの働きをします。インスタンスを停止するたびに、以前のイベント・ログは付加されずに置き換えられます。イ

インスタンスがトラップした場合は、db2eventlog.XXX.crash ファイルも生成されます。これらのファイルは、IBM ソフトウェア・サポートが使用するためのものです。

DB2 コールアウト・スクリプト (db2cos) 出力ファイル

- オペレーティング・システム: すべて
- デフォルト・ロケーション: **diagpath** データベース・マネージャー構成パラメーターで指定されるディレクトリーにあります。
- db2cos スクリプトが FODC 停止の結果として実行される場合、db2cos 出力ファイルは、**diagpath** データベース・マネージャー構成パラメーターで指定された場所に作成された FODC ディレクトリーの下に置かれます。
- パニック、トラップ、またはセグメンテーション違反が発生すると自動的に作成されます。 **db2pdcfg** コマンドを使って指定された特定の問題のシナリオで作成される場合もあります。
- デフォルトの db2cos スクリプトは、**db2pd** コマンドを呼び出してアンラッチ方式で情報を集めます。 db2cos 出力ファイルの内容は、オペレーティング・システムのコマンドおよび他の DB2 診断ツールなど、db2cos スクリプトに含まれるコマンドに応じて異なります。db2cos スクリプトで実行されるツールに関する詳細については、テキスト・エディターでスクリプト・ファイルを開いてください。
- db2cos スクリプトは、出荷時に bin/ ディレクトリーに入っています。UNIX の場合、このディレクトリーは読み取り専用です。このスクリプトの独自の変更可能バージョンを作成する場合は、db2cos スクリプトを adm/ ディレクトリーにコピーしてください。そのバージョンのスクリプトは、自由に変更できます。スクリプトが adm/ ディレクトリーにある場合は、そのバージョンが実行されます。そうでない場合は、bin/ ディレクトリーにあるデフォルト・バージョンが実行されます。

ダンプ・ファイル

- オペレーティング・システム: すべて
- デフォルト・ロケーション: **diagpath** データベース・マネージャー構成パラメーターで指定されるディレクトリーにあります。
- これらのファイルが、FODC 停止中にダンプされた場合、それらのファイルは FODC ディレクトリーの下に置かれます。
- 特定の問題のシナリオが現れると自動的に作成されます。
- エラー状態によっては、失敗したプロセス ID の名前が付いたバイナリー・ファイルに追加情報が記録されます。これらのファイルは、IBM ソフトウェア・サポートが使用するためのものです。

トラップ・ファイル

- オペレーティング・システム: すべて
- デフォルト・ロケーション: **diagpath** データベース・マネージャー構成パラメーターで指定されるディレクトリーにあります。
- これらのファイルが、FODC 停止中にダンプされた場合、それらのファイルは FODC ディレクトリーの下に置かれます。

- インスタンスが異常終了すると自動的に作成されます。 **db2pd** コマンドを使って随意に作成することもできます。
- データベース・マネージャーは、トラップやセグメンテーション違反、例外などにより処理が続行できない時、トラップ・ファイルを生成します。

コア・ファイル

- オペレーティング・システム: Linux および UNIX
- デフォルト・ロケーション: **diagpath** データベース・マネージャー構成パラメーターで指定されるディレクトリーにあります。
- これらのファイルが、FODC 停止中にダンプされた場合、それらのファイルは FODC ディレクトリーの下に置かれます。
- DB2 インスタンスが異常終了するとオペレーティング・システムによって作成されます。
- 特に、コア・イメージには、DB2 のほとんどまたはすべてのメモリ割り振りが組み込まれます。その情報は、問題を説明するために必要な場合があります。

FODC パッケージ・パスと内容

FODC は、指定された FODC パス内に FODC パッケージ・ディレクトリーを作成します。FODC パスを指定するには、**FODCPATH** レジストリー変数設定か、**db2fodc -fodcpath** コマンド・パラメーター・オプションを使用します。FODC パスを指定しないと、First Occurrence Data Capture は診断情報を現行の診断ディレクトリー・パス (**diagpath** または **alt_diagpath**) に送信します。**db2diag** ログ・ファイル診断メッセージはログに記録され、FODC に使用したディレクトリー名を識別します。診断情報をキャプチャーすると、指定されているパラメーターによっては、大量の診断データが生成される可能性があるため、FODC が診断情報を格納するディレクトリー・パス内で十分なスペースが使用可能でなければなりません。FODC がファイル・システム内の使用可能なスペースを占有し、データ・サーバーに影響を及ぼすようなシナリオにならないようにするには、FODC が診断データを格納できる FODC パスを指定することをお勧めします。

自動 FODC の場合、問題が発生しているメンバーまたはパーティションに関するパッケージが収集されます。複数のメンバー上で問題が発生している場合は、複数のパッケージが別々の FODC パッケージ・ディレクトリー内に収集されます。FODC パッケージ・ディレクトリーは、命名規則

FODC_outageType_timestamp_member_number に従います。*outageType* は問題の症状、*timestamp* は FODC 呼び出しの時刻、*member_number* は問題の発生場所のメンバーまたはパーティション番号です。例えば、メンバー 1 でトラップが発生した場合は、FODC は **FODC_Trap_2010-11-17-20.58.30.695243_0001** のような名前のパッケージを自動的に作成します。

手動 FODC の場合、指定したメンバーまたはパーティションに関するパッケージが収集されます。FODC パッケージ・ディレクトリーの命名規則は

FODC_manualOutageType_timestamp_memberList になります。*manualOutageType* は問題の症状、*timestamp* は FODC 呼び出しの時刻、*memberList* は問題の発生場所のメンバーまたはパーティションのリストです。例えば、コマンド **db2fodc -hang -basic -member 1,2,3 -db sample** を手動で発行すると、メンバー 1、2、および 3

に関する手動 FODC パッケージが作成され、FODC_hang_2010-11-17-20.58.30.695243_0001.0002.0003 のような名前になります。

FODC パッケージ・ディレクトリーに、以下のサブディレクトリーが 1 つ以上作成されます。

- DB2CONFIG (DB2 の構成出力とファイルが入ります)
- DB2PD (**db2pd** の出力または出力ファイルが入ります)
- DB2SNAPS (DB2 のスナップショットが入ります)
- DB2TRACE (DB2 のトレースが入ります)
- OSCONFIG (オペレーティング・システムの構成ファイルが入ります)
- OSSNAPS (オペレーティング・システムのモニター情報が入ります)
- OSTRACE (オペレーティング・システムのトレースが入ります)

db2fodc コマンドの実行対象の FODC 構成と障害タイプによっては、これらのディレクトリーの一部が存在しない場合があります。

FODC は以下の診断情報を FODC パッケージ・ディレクトリーに送信します。

db2fodc -c1p は、以下の情報を収集します。

- オペレーティング・システム情報
- インスタンスとデータベースの構成情報

db2fodc -hang は、以下の情報を収集します。

db2fodc -hang は、以下の情報を収集します。

- 基本オペレーティング・システム情報。問題は OS レベル、パッチなどに起因する可能性があります。
- 基本 DB2 構成情報。
- オペレーティング・システムは、vmstat、netstat、iostat などの情報をモニターします。
 - 保存されているタイム・スタンプを使用して少なくとも 2 度繰り返されます。
- 部分的な呼び出しスタック。CPU の上位エージェントの DB2 スタック・トレース。
- オペレーティング・システム・トレース。AIX でトレースします。
- **db2pd** によって収集される診断情報。
- DB2 トレース。
- 全 DB2 呼び出しスタック。
- 2 回目の DB2 構成情報。
 - 2 度目の DB2 トレース・コレクションが含まれます。
- スナップショット情報。**db2 get snapshot for** データベース、アプリケーション、表など。
 - 複数の論理ノードがある場合、ノードごとに情報が収集されます。

db2fodc -perf は、以下の情報を収集している可能性のあるシステムをモニターします。

- スナップショット

- スタック・トレース
- 仮想メモリー (Vmstat)
- 入出力情報 (Iostat)
- トレース
- 状況に応じた他のいくつかの情報。詳しくは、スクリプトを参照してください。

db2fodc -indexerror は、以下の情報を収集します。

- 基本モード
 - db2cos_indexerror_short(.bat) スクリプトが実行されます。詳しくは、スクリプトを参照してください。
 - 該当する **db2dart** コマンドがスクリプト内に存在する場合、**db2dart /DD** と **db2dart /DI** の一方または両方のデータ・フォーマット・アクションがページ数を 100 に限定して実行されます。
- 完全モード
 - db2cos_indexerror_short(.bat) および db2cos_indexerror_long(.bat) スクリプトが実行されます。詳しくは、スクリプトを参照してください。
 - 該当する **db2dart** コマンドがスクリプト db2cos_indexerror_short(.bat) 内に存在する場合、**db2dart /DD** と **db2dart /DI** の一方または両方のデータ・フォーマット・アクションがページ数を 100 に限定して実行されます。
 - 該当する **db2dart** コマンドがスクリプト db2cos_indexerror_long(.bat) 内に存在する場合、**db2dart /DD** と **db2dart /DI** の一方または両方のデータ・フォーマット・アクションがページ数の制限なしで実行されます。
 - 該当する **db2dart** コマンドが db2cos_indexerror_long(.bat) スクリプト内に存在する場合、**db2dart /T** コマンドが実行されます。このコマンドでは、データベースがオフラインであることが必要です。

db2fodc -preupgrade は、以下の情報を収集します。

- オペレーティング・システム情報
- **db2level** コマンドの出力、環境変数、**db2 get dbm cfg** コマンドの出力、db2nodes.cfg ファイルなどの、インスタンスとデータベースの構成情報
- **db2support -d dbname -c -s -c1 0** コマンドで収集されるオペティマイザー情報などの、システム・カタログ・データと統計
- **netstat -v** および **ps -elf** コマンドの出力などの、オペレーティング・システム・モニター・データ
- システム・ファイル
- すべてのスキーマ名を対象にした **DB2 LIST PACKAGES FOR SCHEMA schema-name SHOW DETAIL** コマンドで戻されるパッケージ情報
- db2dump/ にある FODC_Preupgrade ディレクトリー。これらのディレクトリーには、パフォーマンス・データ、上位の動的 SQL 照会、EXPLAIN PLAN などの情報が含まれます。

- /tmp/db2ckupgrade.log,processID 内の、**db2ckupgrade** コマンドからのログ・ファイル (存在する場合)
- **db2prereqcheck** コマンドからの出力

収集対象のメンバーを指定すると、以下の診断情報も組み込まれます。

- スナップショット (すべてのモニター・スイッチをオンにした後)
- -everything、-agents、-applications、-mempools、および -fcm パラメーターに関する **db2pd** コマンド出力
- 最も頻繁に使用される動的 SQL ステートメント
- SQL ステートメントに関する照会プラン
- 静的パッケージに関する EXPLAIN PLAN

手動で **db2fodc** コマンドを呼び出すと、db2fodc_symptom.log という名前のログ・ファイルが FODC_symptom ディレクトリー内に作成されます。symptom は hang や perf などの収集タイプの 1 つです。また **db2fodc** コマンドは、FODC サブディレクトリー内の FODC パッケージについて説明した状況情報およびメタデータをこのファイルに保管します。このファイルには、FODC のタイプに関する情報、データ収集の開始と終了のタイム・スタンプ、および FODC パッケージの分析に役立つその他の情報が入れられます。

自動 FODC データ生成

障害が発生したときに、自動 First Occurrence Data Capture (FODC) が有効になっていると、症状に基づいてデータが収集されます。障害の診断に必要な情報に応じて、それぞれ固有のデータが収集されます。

障害の発生元を通知するために、1 つまたは多数のメッセージ (「重大」と定義されているメッセージなど) が使用されます。

トラップ・ファイルには、以下のような情報が含まれます。

- 仮想ストレージの空き容量
- トラップが発生した時点での製品の構成パラメーターに関連した値とレジストリー変数
- トラップが発生した時点で DB2 製品が使用していたメモリーの概算量
- 障害のコンテキストを示す情報

ロー・スタック・ダンプが ASCII トラップ・ファイルに含まれていることもあります。

データベース・マネージャー内の各コンポーネントに固有のダンプ・ファイルが、該当する FODC パッケージ・ディレクトリーに格納されています。

First Occurrence Data Capture (FODC) を使用するモニター機能と監査機能

モニター機能または監査機能の問題を調査する必要がある場合は、起きた障害の推定原因に関する情報が含まれているログがあります。

DB2 監査ログ (db2audit.log)

- オペレーティング・システム: すべて

- デフォルト・ロケーション:
 - Windows: `$DB2PATH¥instance_name¥security` ディレクトリーにあります。
 - Linux および UNIX: `$HOME¥sqllib¥security` ディレクトリーにあります (`$HOME` はインスタンス所有者のホーム・ディレクトリー)。
- **db2audit** 機能の開始時に作成されます。
- 一連の事前定義されたデータベース・イベントについて DB2 監査機能によって生成された監査レコードが含まれます。

DB2 ガバナー・ログ (*mylog.x*, *x* はガバナーが実行されているデータベース・パーティションの数)

- オペレーティング・システム: すべて
- デフォルト・ロケーション:
 - Windows: `$DB2PATH¥instance_name¥log` ディレクトリーにあります。
 - Linux および UNIX: `$HOME¥sqllib¥log` ディレクトリーにあります (`$HOME` はインスタンス所有者のホーム・ディレクトリー)。
- ガバナー・ユーティリティーの使用時に作成されます。ログ・ファイル名のベースは、**db2gov** コマンドで指定されます。
- ガバナー・デーモンによって実行されたアクション (例えば、アプリケーションの強制実行、ガバナー構成ファイルの読み取り、ユーティリティーの開始または終了) に関する情報や、エラーと警告に関する情報を記録します。

イベント・モニター・ファイル (例えば、00000000.evt)

- オペレーティング・システム: すべて
- デフォルト・ロケーション: ファイル・イベント・モニターの作成時に、すべてのイベント・レコードは CREATE EVENT MONITOR ステートメントで指定されたディレクトリーに書き込まれます。
- イベントの発生時にイベント・モニターによって生成されます。
- イベント・モニターと関連したイベント・レコードが含まれます。

First Occurrence Data Capture (FODC) を使用するグラフィック・ツール

データウェアハウス・センターまたはインフォメーション・カタログ・センターの問題を調査する必要がある場合は、起きた障害の推定原因に関する情報が含まれているログがあります。

データウェアハウス・センターの IWH2LOGC.log ファイル

- オペレーティング・システム: すべて
- デフォルト・ロケーション: `VWS_LOGGING` 環境変数で指定されるディレクトリーにあります。デフォルト・パスは、Windows では `$DB2PATH¥sqllib¥logging` ディレクトリー、Linux および UNIX では `$HOME/sqllib/logging` ディレクトリー (`HOME` はインスタンス所有者のホーム・ディレクトリー) です。
- ロガーが停止した場合にデータウェアハウス・センターによって自動的に作成されます。

- データウェアハウス・センターおよび OLE サーバーによって書き込まれたメッセージで、ロガーが停止した状況で送信できなかったものが含まれます。このログは、データウェアハウス・センターの「ログ・ビューアー」ウィンドウを使用して表示できます。

データウェアハウス・センターの IWH2LOG.log ファイル

- オペレーティング・システム: すべて
- デフォルト・ロケーション: VWS_LOGGING 環境変数で指定されるディレクトリにあります。デフォルト・パスは、Windows では \$DB2PATH%sqllib%logging ディレクトリ、Linux および UNIX では \$HOME/sqllib/logging ディレクトリ (HOME はインスタンス所有者のホーム・ディレクトリ) です。
- データウェアハウス・センターが開始できない場合、またはトレースがアクティブにされたときにデータウェアハウス・センターによって自動的に作成されます。
- データウェアハウス・センターのロガーが開始できず、データウェアハウス・センターのログ (IWH2LOGC.log) に書き込めない状況についての診断情報が含まれます。このログは、データウェアハウス・センターの「ログ・ビューアー」ウィンドウを使用して表示できます。

データウェアハウス・センターの IWH2SERV.log ファイル

- オペレーティング・システム: すべて
- デフォルト・ロケーション: VWS_LOGGING 環境変数で指定されるディレクトリにあります。デフォルト・パスは、Windows では \$DB2PATH%sqllib%logging ディレクトリ、Linux および UNIX では \$HOME/sqllib/logging ディレクトリ (HOME はインスタンス所有者のホーム・ディレクトリ) です。
- データウェアハウス・センターのサーバー・トレース機能によって自動的に作成されます。
- データウェアハウス・センターのスタートアップ・メッセージが含まれます。また、サーバー・トレース機能によって作成されるメッセージを記録します。このログは、データウェアハウス・センターの「ログ・ビューアー」ウィンドウを使用して表示できます。

インフォメーション・カタログ・センターのタグ・ファイル EXPORT ログ

- オペレーティング・システム: すべて
- デフォルト・ロケーション: エクスポートされるタグ・ファイル・パスおよびログ・ファイル名は、インフォメーション・カタログ・センターのエクスポート・ツールの「オプション」タブで指定されます。
- インフォメーション・カタログ・センターのエクスポート・ツールによって生成されます。
- タグ・ファイルのエクスポート情報が含まれます。例えば、エクスポート・プロセスの開始と停止の日時などです。また、エクスポート操作中に発生したエラー・メッセージも含まれます。

インフォメーション・カタログ・センターのタグ・ファイル IMPORT ログ

- オペレーティング・システム: すべて

- デフォルト・ロケーション: インポートされるタグ・ファイル・パスおよびログ・ファイル名は、インフォメーション・カタログ・センターのインポート・ツールで指定されます。
- インフォメーション・カタログ・センターのインポート・ツールによって生成されます。
- タグ・ファイルのインポート履歴情報が含まれます。例えば、インポート・プロセスの開始と停止の日時などです。また、インポート操作中に発生したエラー・メッセージも含まれます。

内部戻りコード

内部戻りコードには、ZRC 値と ECF 値という 2 つのタイプがあります。これらは IBM ソフトウェア・サポートが使用するための診断ツールでのみ通常は表示される戻りコードです。

例えば、これらは DB2 トレース出力および **db2diag** ログ・ファイルで表示されます。

ZRC 値と ECF 値は基本的には同じ目的のためのものですが、形式が少し異なります。各 ZRC 値には、以下の特性があります。

- クラス名
- コンポーネント
- 理由コード
- 関連した SQLCODE
- SQLCA メッセージ・トークン
- 説明

一方、ECF 値は以下のもので構成されます。

- セット名
- 製品 ID
- コンポーネント
- 説明

ZRC 値と ECF 値は通常、負の数値で、エラー状態を表すために使用されます。ZRC 値は、それが表すエラーのタイプによってグループ化されています。このグループ分けは、「クラス」と呼ばれます。例えば、「SQLZ_RC_MEMHEP」で始まる名前の ZRC 値は、一般にメモリー不足に関するエラーです。同様に、ECF 値は「セット」にグループ化されています。

ZRC 値を含む **db2diag** ログ・ファイル項目の例を以下に示します。

```
2006-02-13-14.34.35.965000-300    I17502H435    LEVEL: Error
PID      : 940                TID   : 660    PROC  : db2syscs.exe
INSTANCE: DB2                NODE  : 000    DB    : SAMPLE
APPHDL   : 0-1433            APPID: *LOCAL.DB2.050120082811
FUNCTION: DB2 UDB, data protection, sqlpsize, probe:20
RETCODE  : ZRC=0x860F000A=-2045837302=SQLO_FNEX "File not found."
                DIA8411C A file "" could not be found.
```

この ZRC 値に関する完全な詳細は、例えば次のような **db2diag** コマンドを使用して入手できます。

```

c:¥>db2diag -rc 0x860F000A

Input ZRC string '0x860F000A' parsed as 0x860F000A (-2045837302).

ZRC value to map: 0x860F000A (-2045837302)
    V7 Equivalent ZRC value: 0xFFFFE60A (-6646)

ZRC class :
    Critical Media Error (Class Index: 6)
Component:
    SQL0 ; oper system services (Component Index: 15)
Reason Code:
    10 (0x000A)

Identifer:
    SQL0_FNEX
    SQL0_MOD_NOT_FOUND
Identifer (without component):
    SQLZ_RC_FNEX

Description:
    File not found.

Associated information:
    Sqlcode -980
SQL0980C A disk error occurred. Subsequent SQL statements cannot be
processed.

    Number of sqlca tokens : 0
    Diaglog message number: 8411

```

コマンド **db2diag -rc -2045837302** または **db2diag -rc SQL0_FNEX** を発行した場合も、同じ情報が戻されます。

ECF 戻りコードの出力の例を以下に示します。

```

c:¥>db2diag -rc 0x90000076

Input ECF string '0x90000076' parsed as 0x90000076 (-1879048074).

ECF value to map: 0x90000076 (-1879048074)

ECF Set :
    setecf (Set index : 1)
Product :
    DB2 Common
Component:
    OSSe
Code:
    118 (0x0076)

Identifier:
    ECF_LIB_CANNOT_LOAD

Description:
    Cannot load the specified library

```

db2diag コマンド出力内で最も価値のあるトラブルシューティング情報は、説明および関連情報 (ZRC 戻りコードのみ) です。

ZRC 値の完全なリストを参照するには **db2diag -rc zrc** コマンドを使用し、ECF 値の完全なリストを参照するには **db2diag -rc ecf** コマンドを使用します。

メッセージの概要

本書では、DB2 がインストールされたオペレーティング・システムの機能をよくご存じであることが前提となっています。以下の章に記載されている情報を使用すれば、エラーや問題を識別し、適切なリカバリー処置を行って問題を解決することができます。さらに、この情報を使用すると、メッセージが生成され記録される場所を理解することができます。

メッセージ構造

メッセージ・ヘルプは、メッセージの原因と、そのメッセージへの応答として行うべき処置を説明します。

メッセージ ID は、3 文字のメッセージ接頭部と、それに続く 4 桁または 5 桁のメッセージ番号と、それに続く 1 文字の接尾部から成り立っています。例えば、*SQL1042C* です。メッセージ接頭部のリストについては、787 ページの『メッセージ・ヘルプの呼び出し』 および 787 ページの『その他の DB2 メッセージ』を参照してください。1 文字の接尾部は、エラー・メッセージの重大度を示します。

一般に、メッセージ ID が *C* で終了するものは重大メッセージ、*E* で終了するものは緊急メッセージ、*N* で終了するものはエラー・メッセージ、*W* で終了するものは警告メッセージ、*I* で終了するものは情報メッセージであることを示します。

ADM メッセージの場合、メッセージ ID が *C* で終了するものは重大メッセージ、*E* で終了するものは緊急メッセージ、*W* で終了するものは重要メッセージ、*I* で終了するものは情報メッセージであることを示します。

SQL メッセージの場合、メッセージ ID が *C* で終了するものは重大なシステム・エラー、*N* で終了するものはエラー・メッセージ、*W* で終了するものは警告メッセージまたは情報メッセージであることを示します。

メッセージにはトークン (メッセージ変数と呼ばれることもある) が含まれている場合があります。トークンを含むメッセージが DB2 によって生成される場合、各トークンは検出されたエラー条件に固有の値によって置き換えられ、ユーザーがエラー・メッセージの原因を診断できるようにします。例えば、DB2 メッセージ *SQL0107N* は以下ようになります。

- コマンド行プロセッサから:

SQL0107N 名前 "<name>" が長すぎます。最大長は "<length>" です。

- DB2 インフォメーション・センターから:

SQL0107N 名前 *name* が長すぎます。最大長は *length* です。

このメッセージには、2 つのトークン "<name>" および "<length>" が含まれています。このメッセージが実行時に生成される場合、メッセージ・トークンはエラーの原因となったオブジェクトの実際の名前と、オブジェクトのタイプに許可される最大長にそれぞれ置き換えられます。

トークンがエラーの特定のインスタンスに該当しない場合は、代わりに値 *N が戻されます。例えば以下ようになります。

SQL20416N The value provided ("*N") could not be converted to a security label. Labels for the security policy with a policy ID of "1" should be "8" characters long. The value is "0" characters long. SQLSTATE=23523

メッセージ・ヘルプの呼び出し

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を開いて、以下を入力します。

```
? XXXnnnnn
```

ここで、XXX は有効なメッセージ接頭部、nnnnn は有効なメッセージ番号を表します。

SQLSTATE 値に関連したメッセージ・テキストは、次のコマンドを実行して検索できます。

```
? nnnnn
```

または

```
? nn
```

ここで、nnnnn は 5 桁の SQLSTATE (英数字) のことで、nn は 2 桁の SQLSTATE クラス・コード (SQLSTATE 値の最初の 2 桁) です。

注: db2 コマンドのパラメーターとして受け入れられるメッセージ ID では、大文字小文字の区別はありません。また、単一文字の接尾部はオプションであり、無視されます。

そのため、以下のコマンドの結果は同じになります。

- ? SQL0000N
- ? sql0000
- ? SQL0000w

UNIX ベースのシステムのコマンド行でメッセージ・ヘルプを呼び出すには、以下を入力します。

```
db2 "? XXXnnnnn"
```

ここで、XXX は有効なメッセージ接頭語を表し、nnnnn は有効なメッセージ番号を表します。

メッセージ・テキストが長すぎて画面に収まらない場合、以下のコマンドを使用します (UNIX ベースのシステム、または 'more' をサポートする他のシステム)。

```
db2 "? XXXnnnnn" | more
```

その他の DB2 メッセージ

DB2 コンポーネントの中には、オンラインで使用不可であるメッセージや本書で解説されていないメッセージを戻すものもあります。メッセージ接頭部の中には、以下が入っていることがあります。

AUD DB2 監査機能によって生成されるメッセージ。

DIA 多くの DB2 コンポーネントによって生成される診断メッセージ。これらの

メッセージは、DB2 診断 (db2diag) ログ・ファイルに書き込まれ、エラーの調査時にユーザーや DB2 サービス担当者に追加情報を提供することを目的としています。

ほとんどの場合、これらのメッセージから警告やエラーの原因を判別するのに十分な情報が得られます。メッセージを生成したコマンドやユーティリティーに関する詳細な情報は、該当するコマンドやユーティリティーに関して文書化されている適切な資料を参照してください。

その他のメッセージ・ソース

システムで他のプログラムを実行している場合は、本書で解説されていない接頭部が付いたメッセージを受け取ることがあります。

それらのメッセージについては、該当するプログラム製品の資料を参照してください。

プラットフォーム固有のエラー・ログ情報

DB2 以外のファイルやユーティリティーの中にも、問題の分析に役立つものが多数あります。これらのファイルやユーティリティーは、根本原因を判別するうえで、DB2 ファイルに用意される情報と同じほど重要な役割を果たすことがよくあります。

他のファイルやユーティリティーは、以下の領域に関するログやトレースに含まれる情報にアクセスします。

- オペレーティング・システム
- アプリケーションおよびサード・パーティー・ベンダー
- ハードウェア

稼働環境によっては、上記以外の場所でも役立つ情報が見つかる可能性があります。このため、システムの問題をデバッグするときには、可能性のあるすべての領域をよく調査する必要がある場合があります。

オペレーティング・システム

各オペレーティング・システムには、アクティビティーや障害を追跡するための独自の診断ファイル・セットがあります。通常、最も一般的な (そして最も役立つ) ファイルは、エラー・レポートまたはイベント・ログです。この情報を集める方法は以下のとおりです。

- AIX: `/usr/bin/errpt -a` コマンド
- Solaris: `/var/adm/messages*` ファイルまたは `/usr/bin/dmesg` コマンド
- Linux: `/var/log/messages*` ファイルまたは `/bin/dmesg` コマンド
- HP-UX: `/var/adm/syslog/syslog.log` ファイルまたは `/usr/bin/dmesg` コマンド
- Windows: システム、セキュリティ、およびアプリケーションのイベント・ログ・ファイルと、`windir\drwtsn32.log` ファイル (`windir` は Windows のインストール先ディレクトリー)

各オペレーティング・システムには、これ以外にもトレース・ユーティリティーやデバッグ・ユーティリティーが提供されています。どのような詳細情報が利用可能

かについては、ご使用のオペレーティング・システムの資料やサポート資料を参照してください。

アプリケーションおよびサード・パーティー・ベンダー

各アプリケーションには独自のログ・ファイルと診断ファイルがあります。このようなファイルを DB2 からの情報の補足として利用すれば、問題が存在する可能性のある領域をより正確に把握できます。

ハードウェア

ハードウェア装置は、通常、オペレーティング・システムのエラー・ログの中に情報を記録します。ただし、追加情報が必要になることもあります。そのような場合、ご使用の環境内の特定のハードウェアに関するどのような診断ファイルやユーティリティが存在するかを識別する必要がある場合があります。一例として、不正なページや何らかの破損が DB2 によって報告されることがあります。ほとんどの場合、そのような報告の原因はディスクの問題であり、ハードウェアの診断情報を調査する必要があります。どのような詳細情報が利用可能かについては、ご使用のハードウェアの資料やサポート資料を参照してください。

ハードウェア・ログの情報など、一部の情報には時間の制約があります。エラーが発生したら、関係する情報源からできるだけ早くできるだけ多くの情報を収集するように最善を尽くす必要があります。

以上をまとめると、問題を完全に理解して評価するには、DB2、アプリケーション、オペレーティング・システム、および基礎となるハードウェアから得られるすべての情報を集める必要がある場合があります。**db2support** ツールは必要な DB2 情報とオペレーティング・システム情報のほとんどを自動的に収集しますが、こうして得られる情報以外にも、調査に役立つ情報が見つかる可能性があることに注意してください。

システム・コア・ファイル (Linux および UNIX)

プログラムが異常終了した場合、コア・ファイルがシステムによって作成され、終了処理のメモリー・イメージが保管されます。メモリー・アドレス違反、不正命令、バス・エラー、およびユーザー生成終了シグナルなどのエラーで、コア・ファイルにダンプが保管されます。

コア・ファイルの名前は「core」で、**DB2FODC** レジストリー変数の値で別に構成されない限り、デフォルトで **diagpath** データベース・マネージャー構成パラメーターで指定されるディレクトリーに置かれます。システム・コア・ファイルと、DB2 トラップ・ファイルは別であることに注意してください。

コア・ファイルの制御の設定

コア・ファイルが作成されるとき、大量の診断データが生成される場合があります。そのような場合、格納するためのスペースが必要になると共に、システムのプロセッサ使用量がかかなり必要になる可能性があります。コア・ファイルの処理方法の制御に役立つように、複数の **DB2FODC** レジストリー変数の設定を使用できるようになっています。**db2set** コマンドを使用して **DB2FODC** レジストリー変数を永続的に設定したり、**db2pdcfg** コマンドを使用して動的に (メモリー内のみ) 変更したりできます。**db2set** コマンドを使用して行われた永続設定は、インスタンスがリサ

イクルされるまで有効になりません。db2pdcfg コマンドを使用して行われた動的設定は、即時に有効になり、インスタンスがリサイクルされるまで有効です。

DB2FODC レジストリー変数を使用して、以下のコア・ファイルの動作を制御できます。

- コア・ファイルが生成されるかどうか (**DUMPCORE** 設定を使用)
- コア・ファイルの大きさの上限 (**CORELIMIT** 設定を使用)
- 生成されたコア・ファイルの格納場所 (**DUMPDIR** 設定を使用)

原則として、コア・ファイルの大きさの上限は、コア・ファイルの生成場所となるマシンに搭載されている物理メモリーの量までになっています。例えば、データ・サーバーに 64 GB の物理メモリーがある場合は、コア・ファイルの格納場所のディレクトリー・パス内で 64 GB 以上のスペースが使用可能になっていなければなりません。コア・ファイルのサイズは制限できますが、その代わりに十分なスペースが使用可能なファイル・システムを指すようにコア・ファイルの動作を構成することをお勧めします。コア・ファイル用に使用できるスペースの量を制限しなければならぬ場合は、使用可能なスペースの量がマシン上の物理メモリーの量以上であることを確認してください。この量より少ないと、コア・ファイルが切り捨てられ、診断情報が失われるリスクがあります。例えば、コア・ファイルの生成に使用できるスペースの量を 64 GB に制限し、コア・ファイルを永続的に /tmp にリダイレクトするには、以下のコマンドを発行します。設定が有効になるのは、インスタンスがリサイクルされた後に限られます。

```
db2set DB2FODC="CORELIMIT=64000000000 DUMPDIR=/tmp"
```

コア・ファイルが生成される時、システムのプロセッサ使用量がかなり増える可能性があります、システム可用性にも影響する可能性があります。コア・ファイル生成時のシステム可用性に対するパフォーマンス上の影響が容認できない場合は、コア・ファイルの生成を使用不可にできますが、永続的に使用不可にしないことをお勧めします。コア・ファイルには、問題のトラブルシューティングを正しく行うために必要になることがある診断情報が含まれるからです。コア・ファイルの生成を永続的にオフにしたために、診断情報を使用できない場合は、データ・サーバーの問題をトラブルシューティングできなくなる可能性があります。例えば、コア・ファイルの生成を動的にオフにし、即時に有効にしてインスタンスがリサイクルされるまで有効にしておくには、以下のコマンドを発行します。

```
db2pdcfg DB2FODC="DUMPCORE=OFF"
```

システム・コア・ファイル情報へのアクセス (Linux および UNIX)

dbx システム・コマンドは、どの関数がシステム・コア・ファイルの作成の原因になったかを判別するのに役立ちます。これは、データベース・マネージャーがエラーであるか、またはオペレーティング・システムまたはアプリケーション・エラーがこの問題の原因かどうかのいずれかかを識別するのを助ける、簡単なチェックです。

始める前に

- **dbx** コマンドがインストールされている。このコマンドは、オペレーティング・システム固有のコマンドです。AIX と Solaris の場合は、**dbx** を使用します。HP-UX の場合は **xdb**、Linux の場合は **gdb** をそれぞれ使用します。

- AIX では、**chdev** コマンドまたは **smitty** を使用して、フル・コア・オプションが使用可能かどうかを確認してください。

手順

コア・ファイル・ダンプが発生する原因となった関数を判別するには、次のようにします。

1. UNIX コマンド・プロンプトから、以下のコマンドを入力します。

```
dbx program_name core_filename
```

program_name は異常終了したプログラムの名前、*core_filename* はコア・ファイル・ダンプを含むファイルの名前です。 *core_filename* パラメーターはオプションです。指定しない場合は、デフォルト名 "core" が使用されます。

2. コア・ファイル内の呼び出しスタックを調べます。これを行う方法についての情報は、UNIX コマンド・プロンプトから `man dbx` を発行することによって入手できます。
3. **dbx** コマンドを終了するには、**dbx** プロンプトで **quit** と入力します。

例

以下の例は、**dbx** コマンドを使用して、"main" というプログラムのコア・ファイルを読み取る方法を表示しています。

1. コマンド・プロンプトで、次を入力します。

```
dbx main
```

2. 以下のような出力がディスプレイに表示されます。

```
dbx version 3.1 for AIX.  
Type 'help' for help.  
reading symbolic information ...  
[using memory image in core]  
segmentation.violation in freeSegments at line 136  
136      (void) shmdt((void *) pcAddress[i]);
```

3. コア・ダンプの原因の関数名は "freeSegments" です。dbx プロンプトに **where** と入力して、障害点をポイントするプログラム・パスを表示します。

```
(dbx) where  
freeSegments(numSegs = 2, iSetId = 0x2ff7f730, pcAddress = 0x2ff7f758, line  
136  
in "main.c"  
main (0x1, 2ff7f7d4), line 96 in "main.c"
```

この例では、main.c の行 96 から呼び出された、freeSegments の行 136 でエラーが発生しました。

4. **dbx** コマンドを終了するには、**dbx** プロンプトで **quit** と入力します。

イベント・ログへのアクセス (Windows)

このタスクでは、Windows イベント・ログへのアクセス方法について説明します。

このタスクについて

Windows イベント・ログの情報もまた、役立つ場合があります。以下の 3 種類のイベント・ログをすべて取得することをお勧めします。とくに、DB2 のクラッシュ

ユ、またはシステム・リソースに関連したその他の不可解なエラーが発生した場合には、通常、システム・イベント・ログが最も役立ちます。

- システム
- アプリケーション
- セキュリティー

手順

Windows イベント ビューアを使用してイベント・ログを表示します。ビューアを開く方法は、ご使用の Windows オペレーティング・システムによって異なります。

例えば、Windows XP でイベント ビューアを開くには、「スタート」 > 「コントロール パネル」をクリックします。「管理ツール」を選択してから、「イベント ビューア」をダブルクリックします。

イベント・ログのエクスポート (Windows)

このタスクでは、Windows イベント・ログのエクスポート方法について説明します。

このタスクについて

Windows イベント ビューアから、以下の 2 つの形式でイベント・ログをエクスポートすることができます。

- ログ・ファイル形式
- テキストまたはコンマ区切り形式

手順

Windows イベント ビューアからイベント・ログをエクスポートします。

- ログ・ファイル形式 (*.evt) データを (例えば、他のワークステーション上の) イベント・ビューアに再ロードできます。この形式を使用した場合、ビューアを使って時間の順序を切り替えたり、特定のイベントのフィルター処理や前後への移動を行うことができ便利です。
- テキスト (*.txt) またはコンマ区切り (*.csv) 形式のログは、ほとんどのテキスト・エディターで開くことができます。また、タイム・スタンプに関連して起こる可能性がある問題を回避できます。イベント・ログを .evt 形式でエクスポートした場合、タイム・スタンプは協定世界時で保存され、ビューア内でそのワークステーションの現地時間に変換されます。このため、よく注意しないと、タイム・ゾーンの違いのために重要なイベントを見逃す可能性があります。さらに、テキスト・ファイルは検索が容易です。

ワトソン博士のログ・ファイルへのアクセス (Windows)

このタスクでは、Windows システムのワトソン博士のログ・ファイルにアクセスする方法について説明します。

このタスクについて

ワトソン博士のログ `drwtsn32.log` は、システムで発生したすべての例外の履歴です。DB2 トラップ・ファイルは、ワトソン博士のログよりも役立ちますが、後者

はシステム全体の安定性を評価する場合、および DB2 トラップの履歴のドキュメントとして役立ちます。

手順

ワトソン博士のログ・ファイルを見つけます。デフォルト・パスは `<install_drive>:\Documents and Settings \All Users\Documents\DrWatson` です。

トラップ・ファイル

DB2 は、トラップやセグメンテーション違反、例外などにより処理が続行できない時、トラップ・ファイルを生成します。

DB2 によって予約されたすべてのシグナルまたは例外は、トラップ・ファイルに記録されます。トラップ・ファイルには、エラーが発生したときに実行されていた関数シーケンスも含まれています。このシーケンスは、「関数呼び出しスタック」または「スタック・トレース」と呼ばれることもあります。トラップ・ファイルには、シグナルまたは例外がキャッチされた時のプロセスの状態に関する追加情報も含まれています。

トラップ・ファイルは、`fenced` スレッド・セーフ・ルーチンの実行中にアプリケーションが強制的に停止されるときにも生成されます。トラップは、プロセスのシャットダウン中に発生します。これは致命的エラーではなく、心配する必要はありません。

ファイルは、`diagpath` データベース・マネージャー構成パラメーターで指定されるディレクトリにあります。

すべてのプラットフォームで、トラップ・ファイル名はプロセス ID (PID) で始まり、スレッド ID (TID)、パーティション番号 (単一パーティション・データベースの場合は 000) と続き、「.trap.txt」で終わります。

必ずしもインスタンスが異常終了したとは限らないものの、スタックを参照すると役立つ特定の条件が生じた場合に、コードによって生成される診断トラップもあります。これらのトラップの名前は、10 進形式の PID の後にパーティション番号 (単一パーティション・データベースの場合は 0) を付けた形になります。

例:

- 6881492.2.000.trap.txt は、プロセス ID (PID) が 6881492、スレッド ID (TID) が 2 のトラップ・ファイルです。
- 6881492.2.010.trap.txt は、プロセスとスレッドがパーティション 10 で実行されているトラップ・ファイルです。

`db2pd` コマンドに `-stack all` または `-dump` オプションを指定すると、オンデマンドでトラップ・ファイルを生成することができます。しかし、一般的にこれは、IBM ソフトウェア・サポートが要求した場合にのみ行います。

スタック・トレース・ファイルは、**db2pd -stacks** コマンドまたは **db2pd -dumps** コマンドで生成できます。これらのファイルは、トラップ・ファイルと同じ内容ですが、診断専用として生成されます。それらの名前は、6881492.2.000.stack.txt のようになります。

トラップ・ファイルのフォーマット (Windows)

db2xprt という名前のコマンドでトラップ・ファイル (*.TRP) をフォーマットできます。これは、DB2 データベースのバイナリー・トラップ・ファイルを、人間が理解できる ASCII ファイルにフォーマットします。

このタスクについて

db2xprt ツールは、トラップ・ファイルをフォーマットするのに DB2 シンボル・ファイルを使用します。これらの .PDB ファイルのサブセットが DB2 データベース製品に付属しています。

例

「DB30882416.TRP」という名前のトラップ・ファイルが **diagpath** データベース・マネージャー構成パラメーターで指定したディレクトリーに作成された場合、以下のようにフォーマットできます。

```
db2xprt DB30882416.TRP DB30882416.FMT
```

第 10 章 知識ベースの検索

既知の問題の検索方法

既知の問題を説明している資料には、DB2 APAR、ホワイトペーパー、IBM Redbooks® 資料、技術情報、マニュアルなど、多数のリソースがあります。発生した問題の解決策がすでに存在するかどうかを素早く判別するには、これらの資料（およびその他のリソース）を効率的に検索する必要があります。

検索する前に、問題の状況をはっきり把握しておく必要があります。

問題の状況をはっきり理解したら、既存の解決策が見つかる可能性を高くするために、検索キーワードのリストを作成する必要があります。以下のヒントを参考にしてください。

1. 複数のキーワードを検索で使用します。より適切な検索語を使用すれば、より良い検索結果が得られます。
2. まず結果を絞り込んで検索し、次に必要に応じて検索の幅を広げていきます。例えば、戻された結果が少なすぎる場合には、重要度の低い検索語をいくつか消去して再び検索してみます。あるいは、どのようなキーワードを使うべきかわからない場合には、最初に少数のキーワードを使って幅広く検索し、戻された結果を調べれば、追加すべきキーワードを選択できるでしょう。
3. (複数の単語からなる) 特定の句を検索した方が効率的な場合もあります。例えば、引用符を使って "administration notification file" (管理通知ファイル) と入力すれば、入力内容と同じ語句がこの順序で含まれるドキュメントだけが検出されます。(これら 3 つの単語の任意の組み合わせが含まれるドキュメントがすべて検出されるわけではありません。)
4. ワイルドカードを使用します。特定の SQL エラーが発生した場合、キーワードに「SQL5005<wildcard>」を使うことができます (<wildcard> は検索対象のリソースに応じた適切なワイルドカード)。こうすれば、単に「SQL5005」または「SQL5005c」を使って検索した場合よりも多くの結果が戻されるでしょう。
5. 例えば、インスタンスが異常終了してトラップ・ファイルが生成された場合には、トラップまたはコア・ファイルのスタック・トレースバック内の最初のいくつかの関数をキーワードとして使って既知の問題を検索してください。戻された結果の数が多すぎる場合には、「trap」、「abend」、「crash」などのキーワードを追加してみてください。
6. オペレーティング・システム固有の語句 (例えばシグナル番号やエラー番号の値) を見つけたい場合には、値そのものではなく、定数をキーワードとして検索してみてください。例えば、エラー番号 27 ではなく「EFBIG」を検索します。

一般的には、以下のような検索語を使用すれば成功率が高くなります。

- 実行したコマンドを記述する語句
- 症状を記述する語句
- 診断情報に含まれる語句

トラブルシューティングのリソース

DB2 データベース製品を使用する際に役立つ、トラブルシューティング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 (PDF 資料) で提供されています。

DB2 Technical Support の Web サイト

問題が発生した場合、可能性のある原因と解決策を見つけるうえで DB2 Technical Support の Web サイトが役立ちます。Technical Support のサイトには、最新の DB2 資料、技術情報、プログラム診断依頼書 (APAR)、フィックスパックおよびその他のリソースへのリンクが掲載されています。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 サイトのアドレス: www.ibm.com/software/data/db2/support/db2_9/

第 11 章 DB2 製品フィックスの入手

フィックスパックには、製品テスト中に IBM が検出した問題、および製品使用中にお客様が見つけた問題に対する、コード更新および修正が入っています。最新のフィックスパックを見つける方法、およびデータベース環境にフィックスを適用する方法が論じられます。

フィックスの取得

問題を解決するための製品フィックスが使用可能な場合もあります。フィックスは、これらのステップに従うことによって取得できます。

手順

1. 以下の Web ページからフィックス・リストを表示し、フィックスパックを取得できます。
 - IBM Support Portal: ダウンロード
 - DB2 for Linux, UNIX, and Windows のバージョンごとのフィックス
2. どのフィックスパックが必要かを判断します。一般的には、既知または修正済みのソフトウェア障害が原因となる問題が発生しないように、最新フィックスパックのインストールが推奨されています。
3. フィックスパックをダウンロードし、自己解凍型実行可能パッケージをダブルクリックしてファイルを解凍します。SERVER/doc/your_language/readme.txt 資料を開き、DB2 インフォメーション・センターへの指定リンクに従って、インストール指示を取得します。
4. フィックスを適用します。この説明については、「DB2 サーバー機能 インストール」の『フィックス・パックの適用』を参照してください。

フィックスパック、暫定フィックスパック、およびテスト・フィックス

プログラム診断依頼書 (APAR) は、変更の加えられていない現行リリースの IBM プログラムの欠陥とされるものによって生じた問題の正式な報告書です。APAR は、IBM によってテスト中に見つかった問題、およびお客様によって報告された問題を説明しています。

APAR で説明されている問題を解決する、修正された DB2 コードは、フィックス・パック、暫定フィックスパック、およびテスト・フィックスで配布されることがあります。

フィックスパック

フィックスパックは、APAR フィックスの累積のコレクションです。特に、フィックスパックでは、DB2 の新規リリースの間に生じた APAR を扱っています。これは、特定の保守レベルに上げることを目的としたものです。フィックスパックには、以下の特性があります。

- 累積的なものです。DB2 のリリースの中には、フィックスパックで、そのリリースの以前のフィックスパックおよび暫定フィックスパックで出荷されたすべての APAR フィックスが置き換えられる、つまり、それらがすべて含まれるものもあります。
- サポートされているすべてのオペレーティング・システムとすべての DB2 データベース製品で使用できます。
- 多数の APAR を含んでいます。
- DB2 Technical Support の Web サイトで公開されており、パスポート・アドバンテージ・プログラムを利用して製品を購入されたお客様向けに一般出荷可能になっています。
- IBM によって十分にテストされています。
- データベース製品に対する変更内容、およびフィックスパックのインストール方法と除去方法について説明している資料が付属しています。

注: APAR の修正がフィックスパックで提供されると、APAR のステータスは「オープン」から「プログラム・エラーとしてクローズ」に変わります。個々の APAR のステータスは、DB2 Technical Support の Web サイト上の APAR の説明を調べることによって判別できます。

暫定フィックスパック

暫定フィックスパックは、2 つのフィックスパックの間に生じた重要な APAR フィックスの累積のコレクションです。暫定フィックスパックへの組み込みを限定するために、APAR は普及している、または重要であるとみなされる必要があります。候補の APAR は、DB2 技術サポート・チームのエキスパートによって評価され承認されます。暫定フィックスパックには、以下の特性があります。

- 累積的なものです。DB2 のリリースの中には、暫定フィックスパックで、そのリリースの以前のフィックスパックおよび暫定フィックスパックで出荷されたすべての APAR フィックスが置き換えられる、つまり、それらがすべて含まれるものもあります。
- 各種オペレーティング・システムと各 DB2 データベース製品のサブセットで使用できます。
- 通常は、20 から 30 の新規 APAR を含みます。
- DB2 Technical Support の Web サイトで公開されており、パスポート・アドバンテージ・プログラムを利用して製品を購入されたお客様向けに一般出荷可能になっています。
- IBM によって十分にテストされています。
- 暫定フィックスパックのインストール方法と除去方法について説明している資料が付属しています。

暫定フィックスパックは、それらのリリース後 2 年間実動環境でサポートされます。暫定フィックスパックは、2 つのフィックスパックの間のおよそ中間点で入手可能になり、テスト・フィックスに対する優先の代替手段として意図されています。このテスト・フィックスは、暫定フィックスパックと同じレベルのテストを受けておらず、また同じレベルのサポートも受けられません。

テスト・フィックス

テスト・フィックスは、問題の報告を受けて、テストの目的で特定のお客様に提供される一時的な解決策です。テスト・フィックスは「特別なビルド」と呼ばれることがあり、以下の特性があります。

- 通常は、1 つの APAR を含みます。
- DB2 サポートから入手するもので、一般に出荷されるものではありません。
- IBM によって限られた範囲でテストされています。
- 最小限のドキュメンテーション (テスト・フィックスの適用方法、修正される APAR、テスト・フィックスの除去に関する指示など) が含まれます。

テスト・フィックスは、新たな問題が明らかになり、その問題に対する回避策がなく、次のフィックスパックまたは暫定フィックスパックが入手可能になるまで待つことができないという状況の場合に提供されます。例えば、問題によって業務に重大な影響が生じている場合、APAR がフィックスパックまたは暫定フィックスパックで対応されるまでの間その状況を軽減するためのテスト・フィックスが提供されることがあります。

問題のない運用を続けるために、ご使用の DB2 環境を常に最新のフィックスパック・レベルで実行することをお勧めします。新しいフィックスパックが使用可能になったことの通知を受け取るには、DB2 技術サポートの Web サイト (http://www.ibm.com/software/data/db2/support/db2_9/) で「My Notifications」の E メール更新をサブスクライブしてください。

DB2 フィックスおよびフィックスパックの役割および目的については、support ポリシー・ステートメントを参照してください。

テスト・フィックスの適用

テスト・フィックスは、問題の報告を受けて、テストの目的で特定のお客様に提供される一時的なフィックスです。各テスト・フィックスには、Readme ファイルが付属しています。テスト・フィックスの Readme ファイルには、テスト・フィックスのインストールおよびアンインストールに関する指示や、テスト・フィックスに組み込まれている APAR (ある場合) のリストが記載されています。

始める前に

各テスト・フィックスには固有の前提条件があります。詳細は、テスト・フィックスに付属の Readme ファイルを参照してください。

このタスクについて

テスト・フィックスには以下の 2 つのタイプがあります。

- 個々の DB2 製品についてのテスト・フィックス。これらのテスト・フィックスは、製品の既存のインストールに適用することも、既存の DB2 インストールがない場合に製品のフル・インストールを行うために使用することもできます。
- ユニバーサル・テスト・フィックス (Linux および UNIX のみ)。ユニバーサル・テスト・フィックスは、複数の DB2 製品がインストールされているインストール環境のためのものです。

各国語がインストールされている場合、個別の各国語テスト・フィックスも必要になることがあります。各国語テスト・フィックスは、インストールされている DB2 製品と同じテスト・フィックス・レベルである場合にのみ適用できます。ユニバーサル・テスト・フィックスを適用する場合、DB2 製品を更新するには、ユニバーサル・テスト・フィックスと各国語テスト・フィックスの両方を適用する必要があります。

手順

テスト・フィックスを IBM ソフトウェア・サポートから入手し、Readme ファイルの指示に従って、テスト・フィックスのインストール、テスト、および除去 (必要な場合) を行います。

テスト・フィックスを複数パーティションのデータベース・パーティション環境にインストールする場合、システムはオフラインになっている必要があり、インスタンスに参加するすべてのコンピューターを同じテスト・フィックス・レベルにアップグレードする必要があります。

タスクの結果

第 12 章 サポート

IBM ソフトウェア・サポートへの連絡

IBM ソフトウェア・サポートでは、製品の障害について支援を行っています。

始める前に

IBM ソフトウェア・サポートに連絡する前に、会社にはアクティブな IBM ソフトウェア保守契約があり、問題を IBM に送信することが許可されている必要があります。選択可能な保守契約のタイプについては、「*Software Support Handbook*」の『Premium Support』(techsupport.services.ibm.com/guides/services.html) を参照してください。

手順

以下の手順を完了し、IBM ソフトウェア・サポートに連絡して問題を伝えてください。

1. 問題の明示、バックグラウンド情報の収集、および問題の重大度の決定を行ってください。ヘルプについては、「*Software Support Handbook*」の『Contacting IBM』(techsupport.services.ibm.com/guides/beforecontacting.html) を参照してください。
2. 診断情報を収集してください。
3. 以下のいずれかの方法で問題を IBM ソフトウェア・サポートに連絡してください。
 - オンライン: IBM ソフトウェア・サポートの「**ESR (ESR)**」(Electronic Service Request) リンクをクリックして、サービス要求サイト (www.ibm.com/software/support/probsub.html) を開いてください。
 - 電話: ご自分の国/地域で電話する場合の電話番号については、「*Software Support Handbook*」(techsupport.services.ibm.com/guides/contacts.html) の『Contacts』ページに移動してください。

注: DB2 pureScale ソフトウェア・スタックの一部としてパッケージ化される IBM 製品のサポートが必要な場合は、IBM DB2 pureScale Feature のサービス要求または Problem Management Record (PMR) を開きます。DB2 pureScale Feature の PMR を開くと、問題をより効率的に解決できます。

次のタスク

送信する問題がソフトウェア障害、あるいは欠落または不正確な資料に関する場合は、IBM ソフトウェア・サポートによりプログラム診断依頼書 (APAR) が作成されます。APAR では、問題が詳細に説明されています。可能な限りいつでも、IBM ソフトウェア・サポートからは、APAR が解決され、フィックスできるまで実行できる次善策が示されます。IBM は、IBM ソフトウェア・サポート Web サイトに解決された APAR を毎日公開するので、同じ問題が発生している他のユーザーが同じ解決方法を活用できます。

IBM との情報交換

問題を診断または特定するために、ご使用のシステムのデータおよび情報を IBM サポートに提供する必要がある場合があります。また、問題判別に使用できるツールまたはユーティリティーを、IBM サポートがお客様に提供する場合があります。

IBM サポートへの情報の送信

FTP を経由してまたは Electronic Service Request (ESR) ツールを使用して、IBM ソフトウェア・サポートにデータを送信できます。ここに指示が記載されています。

始める前に

この手順では、IBM ソフトウェア・サポートを使用してすでに Problem Management Record (PMR) を開いていると想定しています。

このタスクについて

ログ・ファイル、構成ファイルなどの診断データを、以下のいずれかの方法を使って IBM ソフトウェア・サポートに送信することができます。

- FTP
- Electronic Service Request (ESR) ツール

IBM サポートの Web サイトでは、これらのデータ交換方式とその他の使用可能なデータ交換方式についての説明があります。

手順

- ファイルを (FTP で) Enhanced Centralized Client Data Repository (EcuRep) に送信するには、次のようにします。

1. 収集したデータ・ファイルを ZIP または TAR 形式に圧縮し、Problem Management Record (PMR) ID にしたがって圧縮ファイルに名前を付けます。

ファイルを PMR に正しく関連付けるために、xxxxx.bbb.ccc.yyy.yyy という命名規則を使用する必要があります (xxxxx は PMR 番号、bbb は PMR のブランチ番号、ccc は PMR のテリトリー・コード、yyy.yyy はファイル名です)。

2. FTP ユーティリティーを使用して、サーバー ftp.emea.ibm.com に接続します。
3. ユーザー ID は「anonymous」でログインして、パスワードに自分の E メール・アドレスを入力します。
4. toibm ディレクトリーに進みます。例えば、`cd toibm` を実行します。
5. オペレーティング・システムに固有のサブディレクトリーの 1 つに移動します。例えば、aix、linux、unix、windows などのサブディレクトリーがあります。
6. バイナリー・モードに変更します。例えば、コマンド・プロンプトで `bin` を入力します。
7. `put` コマンドを使用して、ファイルをサーバー上に配置します。ファイルに名前を付けてサーバー上に配置する場合は、以下のファイル命名規則を使用します。PMR が更新され、xxxx.bbb.ccc.yyy.yyy という形式でファイルの格納場

所がリストされます。(xxx は PMR 番号、bbb はブランチ、ccc はテリトリー・コード、yyy.yyy は tar.Z または xyz.zip のようなファイル・タイプの記述です。) ファイルを FTP サーバーに送信することはできますが、それを更新することはできません。後でファイルを変更する必要がある場合には、新しいファイル名を作成する必要があります。

8. **quit** コマンドを入力します。
- ESR ツールを使用してファイルを送信するには、以下のようにします。
 1. ESR にサインオンします。
 2. ウェルカム・ページで、「**Enter a report number**」フィールドに PMR 番号を入力し、「**Go**」をクリックします。
 3. 「**Attach Relevant File**」フィールドまでスクロールダウンします。
 4. 「**Browse**」をクリックして、IBM ソフトウェア・サポートに送信するログ、トレース、またはその他の診断ファイルを見つけます。
 5. 「**Submit**」をクリックします。ファイルは FTP で IBM ソフトウェア・サポートに転送され、PMR と関連付けられます。

次のタスク

EcuRep サービスの詳細については、IBM EMEA Centralized Customer Data Store Service を参照してください。

ESR についての詳細は、Electronic Service Request (ESR) ヘルプを参照してください。

IBM サポートからの情報の受け取り

場合によっては、IBM 技術サポート担当員が、診断ツールまたはその他のファイルをダウンロードするようお客様にお願いすることがあります。お客様は FTP を使用して、それらのファイルをダウンロードすることができます。

始める前に

ファイルのダウンロードに使用する推奨サーバーに関する情報と、アクセスするディレクトリーおよびファイルの正確な名前を IBM 技術サポート担当員から受け取ったことを確認してください。

手順

ファイルを IBM サポートからダウンロードするには、次のようにします。

1. FTP を使用して、IBM 技術サポート担当員から通知されたサイトに接続し、`anonymous` としてログインします。パスワードには、自分の E メール・アドレスを使用します。
2. 該当するディレクトリーに移動します。
 - a. `/fromibm` ディレクトリーに移動します。

```
cd fromibm
```
 - b. IBM 技術サポート担当員から通知されたディレクトリーに移動します。

```
cd nameofdirectory
```
3. セッションでバイナリー・モードを使用可能にします。

binary

4. **get** コマンドを使用して、IBM 技術サポート担当員が指定したファイルをダウンロードします。

`get filename.extension`

5. FTP セッションを終了します。

`quit`

サポート更新情報のサブスクライブ

DB2 Database for Linux, UNIX, and Windows に関する更新情報を受け取るようにサブスクライブすることで、特定の IBM サポート・ツールおよびリソースに関する重要な技術情報および更新情報を受け取ることができます。

このタスクについて

更新情報をサブスクライブするには、以下のいずれかのアプローチを使用します。

RSS フィードおよびソーシャル・メディアのサブスクリプション

DB2 Database for Linux, UNIX, and Windows では、以下の RSS フィードとソーシャル・メディアのサブスクライブができます。

- DB2 for Linux UNIX and Windows
- DB2 Merge Backup for Linux UNIX and Windows

RSS に関する一般情報 (RSS の始め方や RSS に対応した IBM Web ページのリストなど) は、IBM Software Support RSS feeds サイトを参照してください。

My Notifications

「My Notifications」を使用すると、任意の IBM 製品のサポート更新情報をサブスクライブできます。以前に使用されていた類似ツールの「My Support」は「My Notifications」に置き換えられました。「My Notifications」を使用すると、毎日または毎週 E メールによる告知を受け取るように指定できます。資料、ヒント、製品フラッシュ (アラートとも呼ばれる)、ダウンロード、ドライバーなど、受信する情報のタイプを指定することができます。「My Notifications」では、通知を希望する製品と最もニーズに適した配信方式をカスタマイズおよびカテゴリー化できます。

手順

サポート更新情報をサブスクライブするには、次のようにします。

1. ブラウザーで RSS フィードの 1 つの URL を位置指定して、「**Subscribe Now**」をクリックすることで、DB2 の RSS フィードをサブスクライブします。
2. 「My Notifications」をサブスクライブするには、IBM Support Portal にアクセスし、「**通知**」ポートレットの「**My Notifications**」をクリックします。
3. IBM ID とパスワードを使用してサインインし、「**送信**」をクリックします。
4. 受け取る更新内容と更新の受け取り方法を指定します。
 - a. 「**サブスクライブ**」タブをクリックします。
 - b. 該当するソフトウェア・ブランドまたはハードウェアのタイプを選択します。

- c. 1 つ以上の製品名を選択し、「次へ進む」をクリックします。
- d. 更新情報の受信方法 (E メール受信、指定したフォルダーにオンライン受信、RSS フィードまたは Atom フィードによる受信) の設定を選択します。
- e. 受信する文書更新情報のタイプ (製品ダウンロードに関する新情報やディスカッション・グループのコメントなど) を選択します。
- f. 「Submit」をクリックします。

タスクの結果

RSS フィードと「My Notifications」の設定を変更するまで、要求した更新の通知を受信しません。設定は必要に応じて変更できます。

第 3 部 付録

付録 A. DB2 技術情報の概説

DB2 技術情報は、さまざまな方法でアクセスすることが可能な、各種形式で入手できます。

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2インフォメーション・センター
 - トピック (タスク、概念、およびリファレンス・トピック)
 - サンプル・プログラム
 - チュートリアル
- DB2 資料
 - PDF ファイル (ダウンロード可能)
 - PDF ファイル (DB2 PDF DVD に含まれる)
 - 印刷資料
- コマンド行ヘルプ
 - コマンド・ヘルプ
 - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。DB2 Information Management ソフトウェア・ライブラリー・サイト (<http://www.ibm.com/software/data/sw-library/>) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、IBM Publications Center (www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) から利用できる DB2 ライブラリーについて説明しています。英語および翻訳された DB2 バージョン 10.1 のマニュアル (PDF 形式) は、www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 からダウンロードできます。

この表には印刷資料が入手可能かどうかを示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: DB2 インフォメーション・センターは、PDF やハードコピー資料よりも頻繁に更新されます。

表 93. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SA88-4671-00	入手可能	2012 年 4 月
管理ルーチンおよびビュー	SA88-4672-00	入手不可	2012 年 4 月
コール・レベル・イン ターフェース ガイドお よびリファレンス 第 1 巻	SA88-4676-00	入手可能	2012 年 4 月
コール・レベル・イン ターフェース ガイドお よびリファレンス 第 2 巻	SA88-4677-00	入手可能	2012 年 4 月
コマンド・リファレン ス	SA88-4673-00	入手可能	2012 年 4 月
データベース: 管理の 概念および構成リファ レンス	SA88-4662-00	入手可能	2012 年 4 月
データ移動ユーティリ ティー ガイドおよびリ ファレンス	SA88-4693-00	入手可能	2012 年 4 月
データベースのモニタ リング ガイドおよび リファレンス	SA88-4663-00	入手可能	2012 年 4 月
データ・リカバリーと 高可用性 ガイドおよび リファレンス	SA88-4694-00	入手可能	2012 年 4 月
データベース・セキュ リティ・ガイド	SA88-4695-00	入手可能	2012 年 4 月

表 93. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 ワークロード管理ガイドおよびリファレンス	SA88-4685-00	入手可能	2012 年 4 月
ADO.NET および OLE DB アプリケーションの開発	SA88-4665-00	入手可能	2012 年 4 月
組み込み SQL アプリケーションの開発	SA88-4666-00	入手可能	2012 年 4 月
Java アプリケーションの開発	SA88-4669-00	入手可能	2012 年 4 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SA88-4670-00	入手不可	2012 年 4 月
SQL および外部ルーチンの開発	SA88-4667-00	入手可能	2012 年 4 月
データベース・アプリケーション開発の基礎	GI88-4279-00	入手可能	2012 年 4 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4280-00	入手可能	2012 年 4 月
グローバル化セッション・ガイド	SA88-4696-00	入手可能	2012 年 4 月
DB2 サーバー機能 インストール	GA88-4679-00	入手可能	2012 年 4 月
IBM データ・サーバー・クライアント機能インストール	GA88-4680-00	入手不可	2012 年 4 月
メッセージ・リファレンス 第 1 巻	SA88-4688-00	入手不可	2012 年 4 月
メッセージ・リファレンス 第 2 巻	SA88-4689-00	入手不可	2012 年 4 月
Net Search Extender 管理およびユーザズ・ガイド	SA88-4691-00	入手不可	2012 年 4 月
パーティションおよびクラスタリングのガイド	SA88-4697-00	入手可能	2012 年 4 月
pureXML ガイド	SA88-4686-00	入手可能	2012 年 4 月
Spatial Extender ユーザズ・ガイドおよびリファレンス	SA88-4690-00	入手不可	2012 年 4 月

表 93. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能 かどうか	最終更新
SQL プロシージャ言語: アプリケーション のイネーブルメントお よびサポート	SA88-4668-00	入手可能	2012 年 4 月
SQL リファレンス 第 1 巻	SA88-4674-00	入手可能	2012 年 4 月
SQL リファレンス 第 2 巻	SA88-4675-00	入手可能	2012 年 4 月
Text Search ガイド	SA88-4692-00	入手可能	2012 年 4 月
問題判別およびデータ ベース・パフォーマンス のチューニング	SA88-4664-00	入手可能	2012 年 4 月
DB2 バージョン 10.1 へのアップグレード	SA88-4678-00	入手可能	2012 年 4 月
DB2 バージョン 10.1 の新機能	SA88-4684-00	入手可能	2012 年 4 月
XQuery リファレンス	SA88-4687-00	入手不可	2012 年 4 月

表 94. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能 かどうか	最終更新
DB2 Connect Personal Edition インストールお よび構成	SA88-4681-00	入手可能	2012 年 4 月
DB2 Connect サーバー 機能 インストールおよ び構成	SA88-4682-00	入手可能	2012 年 4 月
DB2 Connect ユーザー ズ・ガイド	SA88-4683-00	入手可能	2012 年 4 月

コマンド行プロセッサから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果の原因に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 状態ヘルプを開始するには、コマンド行プロセッサを開いて以下のように入力します。

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。

例えば、? 08003 を指定すると SQL 状態 08003 のヘルプが表示され、? 08 を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

他のバージョンの DB2 製品の資料は、ibm.com® のそれぞれのインフォメーション・センターにあります。

このタスクについて

DB2 バージョン 10.1 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1> です。

DB2 バージョン 9.8 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/> です。

DB2 バージョン 9.7 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/> です。

DB2 バージョン 9.5 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5> です。

DB2 バージョン 9.1 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9/> です。

DB2 バージョン 8 のトピックについては、DB2 インフォメーション・センターの URL (<http://publib.boulder.ibm.com/infocenter/db2luw/v8/>) を参照してください。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

ローカルにインストールした DB2 インフォメーション・センターは、定期的に更新する必要があります。

始める前に

DB2 バージョン 10.1 インフォメーション・センターが既にインストール済みである必要があります。詳しくは、「DB2 サーバー機能 インストール」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

このタスクについて

既存の DB2 インフォメーション・センターは、自動で更新することも手動で更新することもできます。

- 自動更新は、既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、手動更新と比べて、更新中にインフォメーション

ン・センターが使用できなくなる時間が短くなるというメリットがあります。さらに、自動更新は、定期的に行う他のバッチ・ジョブの一部として実行されるように設定することができます。

- 手動更新は、既存のインフォメーション・センターのフィーチャーと言語の更新に使用できます。自動更新は更新処理中のダウン時間を減らすことができますが、フィーチャーまたは言語を追加する場合は手動処理を使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。自動更新処理では、インフォメーション・センターは、更新を行った後に、インフォメーション・センターを再始動するための停止が発生するだけで済みます。

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

手順

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動更新する手順を以下に示します。

1. Linux オペレーティング・システムの場合、次のようにします。
 - a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`/opt/ibm/db2ic/V10.1` ディレクトリーにインストールされています。
 - b. インストール・ディレクトリーから `doc/bin` ディレクトリーにナビゲートします。
 - c. 次のように `update-ic` スクリプトを実行します。

```
update-ic
```
2. Windows オペレーティング・システムの場合、次のようにします。
 - a. コマンド・ウィンドウを開きます。
 - b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`<Program Files>¥IBM¥DB2 Information Center¥バージョン 10.1` ディレクトリーにインストールされています (`<Program Files>` は「Program Files」ディレクトリーのロケーション)。
 - c. インストール・ディレクトリーから `doc¥bin` ディレクトリーにナビゲートします。
 - d. 次のように `update-ic.bat` ファイルを実行します。

```
update-ic.bat
```

タスクの結果

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、`doc\%eclipse%configuration` ディレクトリにあります。ログ・ファイル名はランダムに生成された名前です。例えば、`1239053440785.log` のようになります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

このタスクについて

ローカルにインストールされた *DB2* インフォメーション・センター を手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の *DB2* インフォメーション・センター を停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。*DB2* インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

注: ご使用の環境において、インターネットに接続されていないマシンに *DB2* インフォメーション・センター の更新をインストールする必要がある場合、インターネットに接続されていて *DB2* インフォメーション・センター がインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。

更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。

3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の *DB2* インフォメーション・センター を再開します。

注: Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、「管理者として実行」を選択します。

手順

コンピューターまたはイントラネット・サーバーにインストール済みの *DB2* インフォメーション・センター を更新するには、以下のようにします。

1. *DB2* インフォメーション・センター を停止します。
 - Windows では、「スタート」 > 「コントロール パネル」 > 「管理ツール」 > 「サービス」をクリックします。次に、「**DB2** インフォメーション・センター」サービスを右クリックして「停止」を選択します。
 - Linux では、以下のコマンドを入力します。

```
/etc/init.d/db2icdv10 stop
```
 2. インフォメーション・センターをスタンドアロン・モードで開始します。
 - Windows の場合:
 - a. コマンド・ウィンドウを開きます。
 - b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、`Program_Files\IBM\DB2 Information Center\バージョン 10.1` ディレクトリーにインストールされています (`Program_Files` は Program Files ディレクトリーのロケーション)。
 - c. インストール・ディレクトリーから `doc\bin` ディレクトリーにナビゲートします。
 - d. 次のように `help_start.bat` ファイルを実行します。

```
help_start.bat
```
 - Linux の場合:
 - a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、`/opt/ibm/db2ic/V10.1` ディレクトリーにインストールされています。
 - b. インストール・ディレクトリーから `doc/bin` ディレクトリーにナビゲートします。
 - c. 次のように `help_start` スクリプトを実行します。

```
help_start
```
- システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「更新」ボタン (🔄) をクリックします。(ブラウザーで JavaScript が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「更新の検索」をクリックします。既存の文書に対する更新のリストが表示されます。
 4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「更新のインストール」をクリックします。
 5. インストール・プロセスが完了したら、「完了」をクリックします。
 6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
 - Windows の場合は、インストール・ディレクトリーの `doc\bin` ディレクトリーにナビゲートしてから、次のように `help_end.bat` ファイルを実行します。

help_end.bat

注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。

- Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

help_end

注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを停止しないでください。

7. DB2 インフォメーション・センター を再開します。

- Windows では、「スタート」 > 「コントロール パネル」 > 「管理ツール」 > 「サービス」をクリックします。次に、「DB2 インフォメーション・センター」サービスを右クリックして「開始」を選択します。
- Linux では、以下のコマンドを入力します。

```
/etc/init.d/db2icdv10 start
```

タスクの結果

更新された DB2 インフォメーション・センター に、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 データベース製品のさまざまな機能について学習するための支援となります。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/>) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML* ガイド」の『**pureXML**®』

XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 の資料

トラブルシューティング情報は、「問題判別およびデータベース・パフォーマンスのチューニング」または *DB2* インフォメーション・センターの『データベースの基本』セクションにあります。ここには、以下の情報が記載されています。

- *DB2* 診断ツールおよびユーティリティーを使用した、問題の切り分け方法および識別方法に関する情報。
- 最も一般的な問題のうち、いくつかの解決方法。
- *DB2* データベース製品で発生する可能性のある、その他の問題の解決に役立つアドバイス。

IBM サポート・ポータル

現在問題が発生していて、考えられる原因とソリューションを見つけるには、IBM サポート・ポータルを参照してください。Technical Support サイトには、最新の *DB2* 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

IBM サポート・ポータル (http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows) にアクセスしてください。

ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

適用度: これらのご利用条件は、IBM Web サイトのあらゆるご利用条件に追加で適用されるものです。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

権利: ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態を提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

IBM の商標: IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、<http://www.ibm.com/legal/copytrade.shtml> をご覧ください。

付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態を提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

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

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができませんが、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。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認できません。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をお願いします。

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ます、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、

利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは、現存するままの状態を提供されるものであり、いかなる種類の保証も提供されません。IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、<http://www.ibm.com/legal/copytrade.shtml> をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- UNIX は The Open Group の米国およびその他の国における登録商標です。
- インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Celeron、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、Windows、Windows NT、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

アーキテクチャー

概要 37

アイドルでの「オフラインに失敗」状態のクリア

トラブルシューティング

DB2 pureScale Feature 710

アクセス要求エレメント

ACCESS 436

IXAND 438

IXOR 441

IXSCAN 442

LPREFETCH 443

TBSCAN 443

XANDOR 444

XISCAN 445

アクセス・タイプ

Explain 情報 349

アクセス・プラン

共有 371

グループ化 302

再利用

詳細 372

索引

構造 73

スキャン 272

情報キャプチャー

Explain 機能 318

ソート 302

複数述部の列相関 453

ロック

細分性 221

標準の表のモード 228

モード 225

REFRESH TABLE ステートメント 350

SET INTEGRITY ステートメント 350

アクセス・プランへのグループ化の影響 302

圧縮

索引

パフォーマンスの影響 516, 661

データ行

パフォーマンスの影響 516, 661

アプリケーション

パフォーマンス

アプリケーション設計 172

カタログ統計を使用するモデル化 514

アプリケーション (続き)

パフォーマンス (続き)

手動調整したカタログ統計を使用するモデル化 513

ロック管理 221

アプリケーション設計

アプリケーション・パフォーマンス 172

アプリケーション・プロセス

詳細 172

ロックへの影響 225

アラート 712

DB2 pureScale

解釈 703, 709, 723

ホスト 703, 708, 711

メンバー 703, 704, 708, 709

DB2 pureScale 環境

ホスト 704

host 718

アラートなし

DB2 pureScale

ホスト 723

メンバー 723

アンインストール

トラブルシューティング 735

暗黙的な隠し列

DB2 pureScale 環境 88

意思決定支援システム (DSS) 579

一時表の情報

db2expln コマンド 359

一般最適化ガイドライン 417

イベント・モニター

トラブルシューティング 781

インスタンス

Explain 情報 325

インストール

インフォメーション・センター

問題 635

エラー・ログ 631

問題

トラブルシューティング 634

分析 632

DB2 pureScale Feature

トラブルシューティング 673, 683

DB2 製品

既知の問題 635

トラブルシューティング 631

DB2 データベース製品のリスト 551

インフォメーション・カタログ・センター

トラブルシューティング 782

インプレース表再編成 147

インライン LOB 518

- エージェント
 - 管理 46
 - クライアント接続 53
 - 作業エージェント・タイプ 45
 - パーティション・データベース 55
- エクスポート
 - 表示 673
- オーバーフロー・レコード
 - パフォーマンスの影響 157
 - 標準の表 68
- 応答ファイル
 - DB2 pureScale Feature
 - トラブルシューティング 686
- オプティマイザー
 - 調整 380
 - 統計ビュー
 - 概要 455
 - 作成 457
- オフライン索引再編成
 - スペース所要量 161
- オフライン表再編成
 - 欠点 141
 - 実行 145
 - 障害 146
 - スペース所要量 161
 - その間に作成される一時ファイル 144
 - パフォーマンスの改善 147
 - フェーズ 144
 - リカバリー 146
 - 利点 141
 - ロック状態 144
- オンライン索引再編成
 - 並行性 154
 - ログ・スペース所要量 161
 - ロック 154
- オンライン表再編成
 - 一時停止 150
 - 欠点 141
 - 再始動 150
 - 実行 149
 - 詳細 147
 - 並行性 150
 - リカバリー 149
 - 利点 141
 - ログ・スペース所要量 161
 - ロック 150

[カ行]

- カーソル固定 (CS)
 - 詳細 175
- カーディナリティ推定値
 - 統計ビュー 458
- 外部結合
 - 不要な 194

- カタログ統計
 - 概要 467
 - カタログ表の説明 470
 - 索引クラスター率 279
 - 実動データベースのモデル化 514
 - 収集
 - 一般 492
 - ガイドライン 490
 - 索引統計 498
 - 特定の列に関する分散統計 504
 - 手動更新の回避 514
 - 手動更新の規則
 - 一般 495
 - 索引統計 499
 - ニックネーム統計 496
 - 表統計 496
 - 分散統計 509
 - 列統計 496
 - 詳細索引データ 497
 - 分散統計 501, 506
 - モデル化の手動調整 513
 - ユーザー定義関数 510
 - 列内のサブエレメント 494
- 監査機能
 - トラブルシューティング 781
- 管理通知ログ
 - 解釈 755
 - 詳細 753
 - First Occurrence Data Capture (FODC) 769
- 管理用タスク・スケジューラー
 - トラブルシューティング 618
- キー
 - コンパイル 420
 - ステートメント 420
- キューのモニター 741
- 行 ID
 - 表アクセスの前の準備 364
- 行圧縮
 - パフォーマンスの影響 516, 661
- 行ブロッキング
 - 指定 216
- クラスター・ファイル・システム
 - 失敗 703, 704, 706
- クラスター・マネージャー・ドメイン
 - 修復 732
- クラスター・マネージャー・リソース・モデル
 - 修復 731
- クラスタリング索引
 - パーティション表 95
- グローバルな最適化
 - ガイドライン 417
- グローバル変数
 - トラブルシューティング 626
- グローバル・レジストリー
 - 変更 546

結合

- 最適マイザー選択 289
- 最適マイザーによる副照会トランスフォーメーション 251
- 概要 286
- 共有集約 251
- ジグザグ
 - 索引ギャップのある 203
 - 例 199
- スター・スキーマ 195
- ネスト・ループ 287
- パーティション・データベース環境
 - 表キュー・ストラテジー 295
 - 方式 296
- ハッシュ 287
- 不要な外部 194
- 方式 296
- マージ 287
- Explain 情報 349, 361

結合述部 193

結合要求エレメント

- HSJOIN 448
- JOIN 447
- MSJOIN 448
- NLJOIN 449

幻像読み取り

- 分離レベル 175
- 並行性の制御 174

コーディネーター・エージェント

- 詳細 39, 48
- 接続コンセントレーターの使用 53

コード・ページ

- ベスト・プラクティス 56

コール・レベル・インターフェース (CLI)

- アプリケーション
 - トレース機能の構成 592
- トレース機能
 - 開始 592
- トレース・ファイル
 - 概要 598
 - トラブルシューティングの概要 587

- 分離レベル 182

コア・ファイル

- Linux システム 789
- UNIX システム 789

交換サーバー属性コマンド 580

更新

- 消失 174
- データ
 - パフォーマンス 122
- DB2 インフォメーション・センター 813, 815

構成

- IOCP (AIX) 135

構成アドバイザー

- パフォーマンスの調整 64

構成設定

- ベスト・プラクティス 56

構成パラメーター

- データベース・メモリー 100
- appl_memory 100
- asl_heap_sz 100
- audit_buf_sz 100
- fcm_num_buffers 100
- fcm_num_channels 100
- mon_heap_sz 100
- sheapthres 100

構成ファイル

- ガバナー・ユーティリティー
 - 規則節 26
 - 規則の詳細 22

コマンド

- ACCRDB 580
- ACCRDBRM 580
- ACCSEC 580
- commit 580
- db2cklog 535
- db2dart
 - 概要 538
 - INSPECT コマンドの比較 539
- db2diag
 - db2diag ログ・ファイルの分析 541
- db2drdat
 - 概要 579
- db2gov
 - DB2 ガバナーを開始する 20
 - DB2 ガバナーを停止する 35
- db2inspf
 - 検査結果のフォーマット 628
- db2level
 - バージョン・レベルおよびサービス・レベルの判別 547
- db2look
 - 類似したデータベースの作成 547
- db2ls
 - DB2 製品およびフィーチャーのリスト表示 551
- db2pd
 - 例 553
 - db2cos コマンドによって実行される 766
- db2pdcfg
 - 概要 769
- db2support
 - 環境情報の収集 568
 - 例 763
- db2trc
 - トレースの取得 574
 - トレース・ファイルのフォーマット 577
- EXCSAT 580
- EXCSATRD 580
- INSPECT
 - db2dart コマンドの比較 539
- SECCHK 580

- コミット
 - ロックの解除 172
- コミット・コマンド 580
- ご利用条件
 - 資料 818
- コンセントレーター
 - statement 371
- コンパイラー
 - Explain 機能を使用した情報キャプチャー 318
- コンパイラー書き直し
 - 暗黙の述部の追加 257
 - 相関副照会 253
 - ビューのマージ 251
- コンパイル時間
 - 動的照会
 - パラメーター・マーカーを使用した削減 210
 - DB2_REDUCED_OPTIMIZATION レジストリー変数 210
- コンパイル・キー 420

[サ行]

最終作業単位の応答メッセージ (ENDUOWRM) 580

最適化

アクセス・プラン

- 索引アクセス方式 277
- 索引の使用 272
- ソートとグループ化の影響 302
- 列相関 453

ガイドライン

- 一般 392
- 照会書き直し 393
- 使用されているかの確認 405
- タイプ 392
- トラブルシューティング 662
- 表参照 401
- プラン 396

クラス

- 詳細 373
- 設定 378
- 選択 376

結合ストラテジー 289

照会

- 制約を介した改善 207
- 照会書き直しのメソッド 250
- データ・アクセス方式 271
- 統計 458
- パーティション内並列処理 304
- パーティション表 310
- パーティション・データベース環境での結合 296
- 表と索引の再編成 139
- MDC 表 307

最適化ガイドライン

- 概要 380
- ステートメント・レベル 400
- XML スキーマ
 - 一般最適化ガイドライン 427

最適化ガイドライン (続き)

XML スキーマ (続き)

- 照会書き直し最適化ガイドライン 431
- プラン最適化ガイドライン 433

最適化クラス

- 概要 373

最適化プロファイル

- アプリケーションでの指定 389
- オプティマイザー用の指定 388
- 概要 379, 380
- 管理 451
- 削除 391
- 作成 384
- 詳細 381
- 使用するようにデータ・サーバーを構成する 387
- トラブルシューティング 662
- パッケージへのバインディング 390
- 不完全一致突き合わせ 421
- 不完全一致突き合わせの例 424
- 変更 390
- マッチング 420
- SQL コンパイラー・レジストリー変数の設定 385
- SYSTOOLS.OPT_PROFILE 表 450
- XML スキーマ 406

最適化プロファイル・キャッシュ 450

細分性

- ロック 222

再編成

- エラー処理 151

索引

- オンライン (ロックと並行性) 索引の再利用 154
- 概要 152
- コスト 161
- 自動 165
- 手順 139
- 必要性の判別 157

自動 164

- VOLATILE 表の索引 166

シナリオ

- ExampleBANK のスペース再利用シナリオを参照 167
- 必要性を少なくする 162

表

- オフライン (オンラインと比較) 141
- オフライン (失敗とリカバリー) 146
- オフライン (詳細) 144
- オフライン (パフォーマンスの改善) 147
- オンライン (一時停止と再開) 150
- オンライン (失敗とリカバリー) 149
- オンライン (詳細) 147
- オンライン (手順) 149
- オンライン (ロックと並行性) 150
- コスト 161
- 自動 165
- 手順 139
- 必要性 140
- 必要性の判別 157

再編成 (続き)

- 方式 141
- モニター 151
- ExampleBANK
 - ExampleBANK のスペース再利用シナリオを参照 167

作業単位

- 概要 172

先読みプリフェッチ 129

索引

- オンライン・デフラグ 80
- カタログ統計 498
- 管理
 - 概要 75
 - 標準の表 68
 - ITC 表 71
 - MDC 表 71
 - クラスター率 279
 - クラスタリング
 - 詳細 95
 - 計画 83
 - 構造 73
 - 使用状況を分析するための Explain 情報 349
 - 据え置きクリーンアップ 79
 - 設計アドバイザー 521
 - データの整合性 628
 - データ・アクセス方式 277
 - 統計
 - 手動更新の規則 499
 - 詳細 497
 - パーティション表
 - 詳細 89
 - パフォーマンスのヒント 85
 - 非同期クリーンアップ 77, 79
 - 利点 81
- 索引圧縮
 - データベース・パフォーマンス 516, 661
- 索引スキャン
 - 検索プロセス 73
 - 詳細 272
 - 直前のリーフ・ポインター 73
 - ロック・モード 228
- 索引の再編成
 - 概要 139, 152
 - コスト 161
 - 自動 165
 - 必要性を少なくする 162
 - VOLATILE 表 166
- サブエレメント統計
 - runstats ユーティリティ 495
- サマリー表
 - マテリアライズ照会表 (MQT) 316
- 暫定フィックスパック
 - 詳細 797
- サンプリング
 - データ 218

式

- 検索条件 191
- 列に対する 191
- ジグザグ結合
 - 前提条件 198
- システム・コア・ファイル
 - Linux
 - 概要 789
 - 情報へのアクセス 790
 - UNIX
 - 概要 789
 - 情報へのアクセス 790
- システム・コマンド
 - dbx (UNIX) 790
 - gdb (Linux) 790
 - xdb (HP-UX) 790
- システム・パフォーマンス
 - モニター 13
- システム・プロセス 39
- 自動再編成
 - 使用可能化 165
 - 詳細 164
- 自動統計収集
 - 使用可能化 482
 - ストレージ 484
- 自動統計プロファイル作成
 - ストレージ 484
- 自動保守
 - 索引の再編成
 - VOLATILE 表 166
- 自動メモリー・チューニング 110
- シナリオ
 - アクセス・プラン 350
 - カーディナリティー推定値を向上させる 458
- シャドー・ページング
 - LONG オブジェクト 517
- 集約関数
 - db2expln コマンド 365
- 従来の表再編成 144
- 述部
 - 暗黙の
 - 例 257
 - オプティマイザーによる変換 250
 - 結合
 - 式での 191
 - 非等価 193
 - 照会処理 283
 - 冗長の回避 206
 - 単純な等価 454
 - 特性 258
 - ノーオペレーション式 193
 - ローカル
 - 列に対する式を使用した 191
- 述部プッシュダウン照会最適化
 - 結合された SQL/XQuery ステートメント 255
- 順次プリフェッチ 128

- 照会
 - スター・スキーマ結合の基準 195
 - 調整
 - SELECT ステートメント 212
 - SELECT ステートメントの制限 213
 - 動的 210
 - 入力変数 208
- 照会書き直し
 - 最適化ガイドライン 393
 - 例 253
- 照会の最適化
 - カタログ統計 452
 - クラス 373, 376
 - 述部でのノーオペレーション式 193
 - 制約を介した改善 207
 - データベース・パーティション・グループの影響 452
 - パフォーマンス 246
 - 表スペースの影響 65
 - プロファイル 379
 - 分散統計 503
- 照会の作成
 - ベスト・プラクティス 189
- 照合
 - ベスト・プラクティス 56
- 状態 712
 - ホスト、WAITING_FOR_FAILBACK 713
 - メンバー
 - WAITING_FOR_FAILBACK 718
 - ロック・モード 223
- DB2 pureScale
 - ホスト、ACTIVE 708
 - ホスト、INACTIVE 703
 - ホスト、WAITING_FOR_FAILBACK 709
 - ホスト、WAITING_FOR_FAILBACK、INACTIVE、メン
バー 703
 - メンバーERROR 709
 - メンバーWAITING_FOR_FAILBACK 703, 708
 - WAITING_FOR_FAILBACK 711
- DB2 pureScale インスタンス
 - ホスト、INACTIVE 704
 - メンバーWAITING_FOR_FAILBACKINACTIVE 704
- 資料
 - 印刷 810
 - 概要 809
 - 使用に関するご利用条件 818
 - PDF ファイル 810
- 診断
 - トラブルシューティング 669
- 診断情報
 - アプリケーション 788
 - インスタンス管理の問題 609
 - インストールの問題 631
 - 概要 607
 - データ移動の問題 608
 - ハードウェア 788
 - 分析 613, 636
- 診断情報 (続き)
 - ワトソン博士のログ 792
 - DB2 Administration Server (DAS) の問題 609
 - First Occurrence Data Capture (FODC)
 - 構成 772
 - 詳細 769
 - ファイル 769
 - IBM ソフトウェア・サポートへの送信 802
 - Linux
 - システム・コア・ファイル 789
 - 情報の取得 788
 - 診断ツール 604
 - UNIX
 - システム・コア・ファイル 789
 - 情報の取得 788
 - 診断ツール 604
 - Windows
 - イベント・ログ 791
 - 情報の取得 788
 - 診断ツール 603
- 数式
 - バッファ・プールのヒット率 124
- 据え置き索引クリーンアップ
 - モニター 79
- スキャン・シェアリング
 - 概要 280
- スクリプト
 - トラブルシューティング 667
- ステートメント・キー 420
- ステートメント・コンセントレーター
 - 詳細 371
- ストレージ・キー
 - トラブルシューティング 668
- スナップショット・モニター
 - システム・パフォーマンス 13
- スレッド
 - スクリプトのトラブルシューティング 667
 - プロセス・モデル 39, 48
- 整合性
 - 点 172
- 整合点
 - データベース 172
- 静的 SQL
 - 分離レベル 182
- 静的照会
 - 最適化クラスの設定 378
- 制約
 - 照会の最適化の改善 207
- セクション実行時統計
 - Explain 機能出力 345
- 設計アドバイザー
 - 実行 525
 - 詳細 521
 - 制約事項 527
 - 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・データベースへの変換 527

- 設計アドバイザー (続き)
 - 定義、ワークロード 525
- 接続
 - 最初のユーザーのシナリオ 136
- 接続コンセントレーター
 - クライアント接続の改善 53
 - パーティション・データベースにおけるエージェント 55
- セルフチューニング・メモリー
 - 概要 107
 - 使用可能化 108
 - 詳細 107
 - 使用不可 109
 - パーティション・データベース環境 111, 114
 - モニター 110
- セルフチューニング・メモリー・マネージャー
 - セルフチューニング・メモリーを参照 107
- ソート
 - アクセス・プラン 302
 - パフォーマンスの調整 137, 657
- 相関
 - 単純な等価述部 454
- 操作
 - オプティマイザーによる移動 250
 - オプティマイザーによるマージ 250
- 送信バッファ
 - トレース・データ 579
- 挿入時クラスタリング (ITC) 表
 - 表と索引の管理 71
 - ロック・モード
 - 表および RID 索引スキャン 231

[夕行]

- ダーティ読み取り 175
- タイムアウト
 - ロック 242
- タスク
 - トラブルシューティング 618
- ダンプ・ファイル
 - エラー・レポート 768
- チュートリアル
 - トラブルシューティング 818
 - 問題判別 818
 - リスト 817
 - pureXML 817
- チューニング・パーティション
 - 判別 114
- 調整
 - ガイドライン 1, 655
 - 照会 189
 - 制限 1, 655
 - ソート 137, 657
 - SQL、Explain 機能による 319
- ツール
 - 診断
 - Linux 604

- ツール (続き)
 - 診断 (続き)
 - UNIX 604
 - Windows 603
- 通知レベル構成パラメーター
 - 更新 756
- 次キーロック 227
- データウェアハウス・センター
 - トラブルシューティング 782
- データ演算子
 - Explain 情報 324
- データの挿入
 - パフォーマンス 211
 - 非コミット挿入を無視する 186
- データ表記
 - アクセス
 - スキャン・シェアリング 280
 - 方式 271
 - 圧縮 139
 - 照会でのサンプリング 218
 - 不整合 628
- データベース
 - 名前
 - RDBNAM オブジェクト 580
 - 破損 628
 - 非アクティブ化
 - 最初のユーザー接続のシナリオ 136
- データベース分析およびレポート・ツール・コマンド
 - 概要 538
- データベース・エージェント
 - 管理 46
- データベース・エンジンのプロセス 667
- データベース・パーティション
 - 作成 664
- データベース・パーティション・グループ
 - 照会最適化の影響 452
- データベース・マネージャー
 - 共有メモリー 102
- データ・オブジェクト
 - Explain 情報 324
- データ・ストリーム情報
 - db2expln コマンド 363
- データ・ソース
 - パフォーマンス 267
- データ・パーティションの除去 310
- データ・ページ
 - 標準の表 68
- デーモン
 - ガバナー・ユーティリティ 21
- ディスク
 - ストレージのパフォーマンス要因 65
- テスト・フィックス
 - 詳細 797
 - タイプ 799
 - 適用 799

- デッドロック
 - 回避 184
 - 概要 244
- デッドロック 検出機能 244
- デフラグ
 - 索引 80
- 等価述部 454
- 統計
 - カタログ
 - 手動更新の回避 514
 - 詳細 467
 - 収集
 - ガイドライン 490
 - サンプル表データに基づく 493
 - 自動 477, 482
 - 手動での更新 495
 - 照会の最適化 452
 - 列グループ 453
 - 統計ビュー
 - カーディナリティー推定値を向上させる 458
 - 概要 455
 - 最適化統計 458
 - 作成 457
 - 参照整合性制約を使用して統計ビューの数を削減する 464
 - 式列から使用される統計 463
 - 列グループ統計から使用される統計 466
- 統計プロファイル 482
- 動的 SQL
 - 分離レベル 182
- 動的照会
 - 最適化クラスの設定 378
 - パラメーター・マーカーを使用したコンパイル時間の削減 210
- 特記事項 821
- ドメイン
 - ピア
 - 手操作による Reliable Scalable Cluster Technology (RSCT) クリーンアップ 736
- トラップ・ファイル
 - 概要 793
 - フォーマット (Windows) 794
- トラブルシューティング 629
 - 一時表用のディスク・ストレージ・スペース 622
 - インストールの問題 631, 634, 635
 - オンライン情報 818
 - 概要 531, 745
 - 管理用タスク・スケジューラー 618
 - コンプレッション・ディクショナリーが自動的に作成されない 621
 - 持続されたトラップ 613
 - 情報の収集 547, 553, 568, 607, 802
 - 診断データ
 - インスタンスの管理 609
 - インストール 631
 - 基本セットの収集 607
 - 自動収集 769
- トラブルシューティング (続き)
 - 診断データ (続き)
 - 収集の構成 772
 - 手動での収集 769
 - データ移動 608
 - データベース・パーティション・サーバー、データベース・パーティション、またはその両方での分割 750
 - ディレクトリー・パス 747
 - alt_diagpath 747
 - DAS 609
 - diagpath 747
 - 診断ログ 757
 - ストレージ・キー 668
 - 製品のベータ版 635
 - セクション実行時統計の収集 668
 - 操作の失敗 619
 - タスク 618
 - チュートリアル 818
 - ツール 535
 - データ再配分 665
 - データベースが現在使用中 619
 - データベースの作成 664
 - デッドロック問題
 - 解決 645
 - 診断 643
 - トレース
 - 概要 573
 - CLI アプリケーション 587, 592
 - db2trc コマンドを使用して取得 774
 - DRDA 582, 586
 - JDBC アプリケーション 587
 - ODBC アプリケーション 587, 592
 - 内部戻りコード 784
 - パフォーマンス
 - ソート・ヒープ 653
 - メモリー 653
 - パフォーマンス関連の問題 610
 - 表の状態 665
 - フィックスの取得 797
 - ホスト検証 680
 - 問題の解決策の検索 795
 - 問題の再作成 547
 - リソース 796
 - リソース関連の問題 763
 - ロード操作 615
 - ログ・ファイル、db2cklog コマンド 535
 - ログ・レコードの圧縮解除 623
 - ロックの問題
 - 概要 638
 - ロック待機 640, 642
 - ロック・エスカレーション 650, 651
 - ロック・タイムアウト 647, 648
 - DB2 pureScale
 - 作成 673, 683
 - DB2 セットアップ・ウィザード 673

トラブルシューティング (続き)

DB2 pureScale (続き)

db2diag ログ・ファイル項目と cfdump.out* ファイルの
マッピング 693

db2start コマンド 688

restart light を伴うホスト・リブート 727

RSCT ピア・ドメイン 686

Tivoli SA MP の状態とDB2 クラスター・サービスの状
態のマッピング 676

uDAPL over InfiniBand 接続のトレース 697

uDAPL 通信エラー 694

DB2 pureScale Feature

アイドルでの「オフラインに失敗」状態のクリア 710

アンインストール 735

インストール後 688, 700

応答ファイル・インストール 686

概要 669

除去 735, 736

診断 669

ライセンス 687

GPFS 685

/var ファイル・システム 722

DB2 pureScale FeatureDB2 pureScale

インストール 673, 683

DB2 pureScale インスタンス

ホスト 701

メンバー 701

DB2 pureScale 環境

GPFS 障害を伴うカーネル・パニック 729

DB2 データベース製品 607

db2diag ログ・ファイル項目の解釈 758

db2fodc -clp 633

db2pd -load 615

db2pd -utilities 615

FCM の問題 664

IBM サポートへの連絡 801

SQL1035N 619

トラブルシューティングおよびサポート

サポート更新情報のサブスクライブ 804

トレース

概要 573

出力ファイル 579, 580

出力ファイル・サンプル 582

トラブルシューティングの概要 573

CLI

概要 587

取得 592

分析 595, 596, 597

DB2 574, 577

DB2 Connect とサーバー間のデータ 579

DRDA

解釈 579

サンプル 582

バッファ情報 586

JDBC アプリケーション

DB2 Universal JDBC ドライバー 587

トレース・ユーティリティ (db2drdat) 579

[ナ行]

ニックネーム

統計 496

入出力完了ポート (IOCPs)

構成 135

AIX 135

ネスト・ループ結合

詳細 287

ノーオペレーション式 193

[ハ行]

パーティション間のモニター 13

パーティション内並列処理

最適化ストラテジー 304

詳細 219

パーティション表

クラスタリング索引 95

最適化ストラテジー 310

索引 89

ロック 240

パーティション・データベース

結合ストラテジー 295

結合方式 296

照会の非相関化 253

セルフチューニング・メモリー 111, 114

複製されたマテリアライズ照会表 293

ベスト・プラクティス 56

ハードウェア

構成ベスト・プラクティス 56

バインド

分離レベル 182

ハッシュ結合

詳細 287

バッファ・プール

概要 115

大規模の利点 119

調整

ページ・クリーナー 117

複数の管理 119

ブロック・ベースの 130

ページ・クリーニング方式 122

メモリー

開始時の割り振り 119

バッファ・プールのヒット率 124

パフォーマンス

アプリケーション設計 172

ガイドライン 1, 655

概要 1, 655

限界 1, 655

向上

リレーショナル索引 85

パフォーマンス (続き)

- 構成アドバイザー 64
 - システム
 - モニター 13, 15
- 照会 189, 246
- ディスク・ストレージの要因 65
- トラブルシューティング 1, 655
- 評価 319
- 分離レベルの影響 175
- 変更の分析 319
- ロック
 - 管理 221
- db2batch コマンド 7
- Explain 情報 347
- runstats
 - 改善 515, 660
- SQL 照会
 - セクションから生成される Explain 332
 - セクション実行時統計の使用 341
- パラメーター
 - オートノミック
 - ベスト・プラクティス 56
 - メモリーの割り振り 106
 - PRDID 580
- パラメーター・マーカー
 - 動的照会のコンパイル時間の削減 210
- 反復可能読み取り (RR)
 - 詳細 175
- 反復不能読み取り
 - 分離レベル 175
 - 並行性の制御 174
- ピア・ドメイン
 - Reliable Scalable Cluster Technology (RSCT)
 - 手操作によるクリーンアップ 736
- 非コミット読み取り (UR) 分離レベル
 - 詳細 175
- 非コミット・データ
 - 並行性の制御 174
- 非同期索引クリーンアップ
 - 詳細 77
- ビュー
 - オプティマイザーによる述部プッシュダウン 253
 - オプティマイザーによるマージ 251
- 表
 - アクセス情報 354
 - オフライン再編成
 - 詳細 144
 - パフォーマンスの改善 147
 - リカバリー 146
 - オンライン再編成
 - 一時停止と再開 150
 - 詳細 147
 - リカバリー 149
 - 概要 140
 - キュー 295

表 (続き)

- 再編成
 - エラー処理 151
 - オフライン 145
 - オンライン 149
 - 概要 140
 - コスト 161
 - 自動 165
 - 手順 139
 - 必要性の削減 162
 - 必要性の判別 157
 - 方式 141
 - モニター 151
- 挿入時クラスタリング (ITC) 71
- 統計
 - 手動更新の規則 496
- パーティション
 - クラスタリング索引 95
- パーティション・データベースでの結合ストラテジー 295
- 標準
 - 管理 68
- マルチディメンション・クラスタリング (MDC) 71
- ロック・モード 228
- 表およびスペースの再利用シナリオ
 - ExampleBANK のスペース再利用シナリオを参照 167
- 表スペース
 - 照会の最適化 65
- 頻出値分散統計 501
- ファイルへのトレースのダンプ
 - 概要 577
- フィックスパック
 - 概要 797
 - 入手 797
- フェデレーテッド照会の情報
 - db2expln コマンド 368
- フェデレーテッド・データベース
 - グローバルな最適化 267
 - サーバー・オプション 97
 - 照会が評価される場所の判別 265
 - 照会のグローバル分析 270
 - プッシュダウン分析 260
 - 並行性の制御 174
- 副照会
 - 相関する 253
- 複数の再始動
 - メンバー 726
- 複数パーティション・データベース
 - 単一パーティション・データベースからの変換 527
- 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 260
- 物理的なデータベース設計
 - ベスト・プラクティス 56
- フラグメントの除去
 - 「データ・パーティションの除去」を参照 310
- フリー・スペース制御レコード (FSCR)
 - 標準の表 68

- フリー・スペース制御レコード (FSCR) (続き)
 - ITC 表 71
 - MDC 表 71
- プリコンパイル
 - 分離レベルの指定 182
- プリフェッチ
 - 先読み 129
 - 順次 128
 - 入出力サーバー構成 132
 - パフォーマンスの影響 126
 - ブロック・ベースのバッファ・プール 130
 - 並列入出力 132
 - リスト 131
- プログラム診断依頼書 (APAR) 797
- プロセス
 - 概要 37
- プロセス状況ユーティリティー
 - コマンド 580
- プロセス・モデル
 - 詳細 39, 48
- ブロッキング
 - 行 216
- ブロック ID
 - 表アクセスの前の準備 364
- ブロック・ベースのバッファ・プール 130
- プロファイル
 - 最適化
 - 概要 379
 - 詳細 381
 - 統計 482
- 分散データ管理 (DDM)
 - db2drdat 出力 579
- 分散統計
 - 手動更新の規則 509
 - 照会の最適化 503
 - 詳細 501
 - 例 506
- 分離レベル
 - カーソル固定 (CS) 175
 - 指定 182
 - パフォーマンス 175
 - 反復可能読み取り (RR) 175
 - 比較 175
 - 非コミット読み取り (UR) 175
 - 読み取り固定 (RS) 175
 - ロックの細分性 221
- ページ
 - 概要 68
- ページ・クリーナー
 - 調整 117
- 並行性
 - 改善 184
 - フェデレーテッド・データベース 174
 - 問題 174
 - ロック 221

- 並列処理
 - 入出力サーバー構成 132
 - パーティション内
 - 概要 219
 - 最適化ストラテジー 304
 - 非 SMP 環境 219
 - db2expln コマンド情報 365
- I/O
 - 管理 134
- ベスト・プラクティス
 - 照会 189
- ヘルプ
 - SQL ステートメント 812
- 変位値分散統計 501
- ベンチマーク
 - 概要 5
 - サンプル・レポート 11
 - 実行 9
 - 準備 6
 - db2batch コマンド 7
 - SQL ステートメント 6
- ホスト 723
 - DB2 pureScale
 - アラート 711
 - DB2 pureScale 環境
 - 状態、アラート 702
 - トラブルシューティング 701
- 本書について vii
- 本書の構成 vii

[マ行]

- マージ結合
 - 詳細 287
- マテリアライズ照会表 (MQT)
 - 自動サマリー表 316
 - パーティション・データベース 293
 - replicated 293
- マルチディメンション・クラスタリング (MDC) 表
 - 最適化ストラテジー 307
 - 据え置き索引クリーンアップ 79
 - 表と索引の管理 71
 - ブロック・レベルのロック 221
 - ロールアウト削除 307
 - ロック・モード
 - 表および RID 索引スキャン 231
 - ブロック索引スキャン 236
- メッセージ 786
- メモリー
 - 開始時のバッファ・プールの割り振り 119
 - 構成
 - メモリー・セットも参照 100
 - セルフチューニング 107
 - データベース・マネージャー 102
 - トラブルシューティング
 - ソート・ヒープ 653

- メモリー (続き)
 - パーティション・データベース環境 114
 - 割り振り
 - 概要 98
 - パラメーター 106
 - メモリー・セットも参照 100
 - FCM バッファ・プール 105
- メモリー・セット
 - 概要 100
 - 構成パラメーター 100
 - タイプ 100
 - レジストリー変数 100
- メモリー・トラッカー・コマンド
 - 出力例 115
- メンバー 712
 - アラート
 - 解釈 702
 - 状態
 - 解釈 702
 - トラブルシューティング
 - 概要 701
- 戻りコード
 - 内部 784
- モニター
 - アプリケーション動作 20
 - 異常値 19
 - 索引の再編成 155
 - システム・パフォーマンス 13, 15
 - セクション Explain 情報のキャプチャー 330
 - パーティション間 13
 - runstats 512
- 問題判別
 - インストールの問題 631
 - チュートリアル 818
 - 利用できる情報 818

[ヤ行]

- ユーティリティ
- トレース 579
 - db2drdat 579
- ps (プロセス状況) 580
- 有効化
 - DB2 コピー 573
- 読み取り固定 (RS)
 - 詳細 175

[ラ行]

- ラージ・オブジェクト (LOB)
 - インライン 518
- ライセンス
 - 準拠
 - レポート 636
- リスト・プリフェッチ 131

- リレーショナル索引
 - 利点 81
- レコード ID (RID)
 - 標準の表 68
- レジストリー変数
 - DB2_FMP_COMM_HEAPSZ 100
- 列
 - グループ統計 453
 - サブエレメント統計 494
 - 統計 496
 - 分散統計
 - 収集 504
- ロールアウト削除
 - 据え置きクリーンアップ 79
- ロールバック
 - 概要 172
- ロギング
 - スタンドアロン DB2 Text Search サーバー 739
- ログ
 - アーカイブ 517
 - ガバナー・ユーティリティ 31
 - 管理 753
 - 循環ロギング 517
 - 統計 484, 489
- ログ・シーケンス番号 (LSN)
 - ギャップ 122
- ログ・バッファ
- DML のパフォーマンスの向上 517
- ログ・ファイル
 - 妥当性検査 535
- ロック
 - アプリケーション・タイプの影響 225
 - アプリケーション・パフォーマンス 221
 - オブジェクト 223
 - 概要 172
 - 据え置き 187
 - 待機
 - 解決 243
 - 概要 242
 - タイムアウト
 - 回避 184
 - 概要 242
 - 次キー・ロッキング 227
 - データ・アクセス・プランの影響 225
 - デッドロック 244
 - 同時に付与 226
 - パーティション表 240
 - 標準の表 228
 - 分離レベル 175
 - 並行性の制御 221
 - 変換 242
 - ロック・カウント 223
- ロックの細分性
 - 影響を与える要素 224
 - 概要 222

ロック・モード
 互換性 226
 詳細 223
 挿入時クラスタリング (ITC) 表
 表スキャン 231
 RID 索引スキャン 231
 マルチディメンション・クラスタリング (MDC) 表
 表スキャン 231
 ブロック索引スキャン 236
 RID 索引スキャン 231
 IN (意図なし) 223
 IS (意図的共有) 223
 IX (意図的排他) 223
 NS (スキャン共有) 223
 NW (次キーの弱い排他) 223
 S (共有) 223
 SIX (意図的排他共有) 223
 U (更新) 223
 X (排他) 223
 Z (超排他) 223
 論理パーティション
 複数の 48

[ワ行]

ワークロード
 パフォーマンスの調整
 設計アドバイザー 521, 525

[数字]

1 次役割の移動
 メンバー 723

A

ACCRDB コマンド 580
ACCRDBRM コマンド 580
ACCSEC コマンド 580
AIX
 構成
 ベスト・プラクティス 56

C

CLI および ODBC
 トレース
 ファイル 587
CURRENT EXPLAIN MODE 特殊レジスター
 Explain データ 328
CURRENT EXPLAIN SNAPSHOT 特殊レジスター
 Explain データ 328
CURRENT LOCK TIMEOUT 特殊レジスター
 ロック待機モードの方針 243

CURRENT MEMBER
 DB2 pureScale 環境 88
cur_commit データベース構成パラメーター
 概要 184

D

database_memory データベース構成パラメーター
 セルフチューニング 107
DB2 pureScale
 インストール、トラブルシューティング 673, 683
 再始動イベント 720
 作成 683
 除去 735, 736
 診断データの収集
 手動 671
 診断データの収集、トラブルシューティング
 自動、自動データ収集 673
 トラブルシューティング
 アンインストール 735
 インストール、ロールバック、インスタンスの作成
 673, 683
 インストール後 688, 700
 応答ファイル・インストール 686
 概要 669
 手動によるデータ収集 671
 除去 735, 736
 診断 669
 メンバー間での 共有の削減 88
 ライセンス 687
 DB2 セットアップ・ウィザード 673
 db2diag ログ・ファイル項目と cfdump.out* ファイルの
 マッピング 693
 db2start コマンド 688
 GPFS 685
 GPFS 障害を伴うカーネル・パニック 729
 restart light を伴うホスト・リポート 727
 RSCT ピア・ドメイン 686
 Tivoli SA MP の状態とDB2 クラスタ・サービスの状
 態のマッピング 676
 uDAPL over InfiniBand 接続のトレース 697
 uDAPL 通信エラー 694
 トレース・ファイル 671
 ログ・ファイル 671
 restart light を伴うホスト・リポート 727
DB2 pureScale Feature
 トラブルシューティング
 アイドルでの「オフラインに失敗」状態のクリア 710
DB2 pureScale インスタンス
 トラブルシューティング
 ホスト 701
 メンバー 701
 ホスト
 トラブルシューティング 701
 メンバー
 トラブルシューティング 701

- DB2 Text Search 739, 741
 - トラブルシューティング 739
 - トラブルシューティングのヒント 743
 - 保守 739
 - 問題のトレース 739
 - 問題判別 739
- DB2 Universal JDBC ドライバー
 - トレース機能の構成 587
- DB2 インフォメーション・センター
 - 更新 813, 815
 - バージョン 813
- DB2 ガバナー
 - 開始 20
 - 概要 20
 - 規則節 26
 - 構成ファイル 22
 - デーモン 21
 - 停止 35
 - トラブルシューティング 781
 - ログ・ファイル 31
- DB2 製品
 - リスト 551
- DB2 セットアップ・ウィザード
 - DB2 pureScale
 - 起動の問題 673
- db2batch コマンド
 - 概要 7
- db2cklog コマンド
 - トラブルシューティング 535
- db2cli.ini ファイル
 - トレース構成 592
- db2cluster コマンド
 - クラスター・キャッシング・ファシリティの自動オフライ
ン化 733
 - クラスター・マネージャー・ドメインの修復 732
 - 自動ホスト・レポート 733
 - 出力例 708
 - トラブルシューティング・オプション 731
 - メンバーの自動オフライン化 733
- db2cos スクリプト 766
- db2dart コマンド
 - トラブルシューティングの概要 538
 - INSPECT コマンドの比較 539
- db2diag コマンド
 - 例 541
- db2diag ログ
 - 解釈
 - 概要 758
 - db2diag ツールの使用 541
 - 詳細 757
 - マージ 747
 - First Occurrence Data Capture (FODC) 情報 769
- db2drdat コマンド
 - 出力ファイル 579
- db2expln コマンド
 - 出力の説明 353
- db2expln コマンド (続き)
 - 表示される情報
 - 一時表 359
 - 各種 369
 - 行 ID の準備 364
 - 結合 361
 - 更新 363
 - 削除 363
 - 集約 365
 - 挿入 363
 - データ・ストリーム 363
 - 表アクセス 354
 - フェデレーテッド照会 368
 - ブロック ID の準備 364
 - 並列処理 365
- DB2FODC レジストリー変数
 - 診断情報の収集 769
- db2gov コマンド
 - 詳細 20
 - DB2 ガバナーの開始 20
 - DB2 ガバナーの停止 35
- db2inspf コマンド
 - トラブルシューティング 628
- db2instance コマンド
 - 出力例 703, 708, 723, 726
 - ホストの状況 702
 - メンバーの状況 702
- db2level コマンド
 - サービス・レベルの識別 547
 - バージョン・レベルの識別 547
- db2licm コマンド
 - 準拠レポート 636
- db2look コマンド
 - データベースの作成 547
- db2ls コマンド
 - インストールされている製品およびフィーチャーのリスト表
示 551
- db2mtrk コマンド
 - 出力例 115
- db2pd コマンド
 - デフォルトの db2cos スクリプトによって収集される出力
766
 - トラブルシューティングの例 553
- db2pdcfg コマンド
 - DB2FODC レジストリー変数でのオプションの設定 769
- db2start コマンド
 - トラブルシューティング
 - DB2 pureScale 688
- db2support コマンド
 - 実行 763
 - 詳細 568
 - 診断データの収集 673
- db2trc コマンド
 - 概要 574
 - トレース出力のダンプ 577
 - トレース出力のフォーマット 577

db2val コマンド
DB2 コピーの検証 573
DB2_EVALUNCOMMITTED レジストリー変数
行ロックの据え置き 187
DB2_FMP_COMM_HEAPSZ 変数
FMP メモリー・セット構成 100
DB2_REDUCED_OPTIMIZATION レジストリー変数
コンパイル時間の削減 210
DB2_SKIPINSERTED レジストリー変数
詳細 186
DB2_USE_ALTERNATE_PAGE_CLEANSING レジストリー変数
先行ページ・クリーニング 122
ddcstrc ユーティリティ 580
DEGREE 一般要求エレメント 428
diaglevel 構成パラメーター
更新 762
DPFXMLMOVEMENT 一般要求エレメント 428

E

ECF 戻りコード 784
ExampleBANK のスペース再利用シナリオ
概要 167
索引の保守 170
スペース管理ポリシー 167
スペースの再利用
索引の 168
表の 168
挿入時クラスタリング表 167
挿入時クラスタリング表への変換 169
AUTO_REORG ポリシー 171
EXCSAT コマンド 580
EXCSATRD コマンド 580
Explain 機能
インスタンス情報 325
概要 318, 351, 353
出力
セクション実行時統計 345
パッケージ・キャッシュのセクションからの生成 332
情報 321
情報のキャプチャー 328
情報の使用のガイドライン 347
情報の分析 349
スナップショットの作成 328
セクション Explain 336
セクション Explain 情報のキャプチャー 330
セクション実行時統計情報のキャプチャー 338
データ演算子情報 324
データ・オブジェクト情報 324
フェデレーテッド照会が評価される場所の判別 265
フェデレーテッド・データベース 270
db2exfmt コマンド 351
db2expln コマンド 351
Explain インスタンス 321
EXPLAIN ステートメント 336
Explain 表 321

Explain 機能 (続き)
SQL のチューニング 319
EXPLAIN ステートメント
EXPLAIN_FROM_SECTION との比較 332
Explain 表
編成 321
EXPLAIN_FROM_SECTION プロシージャ
例 332
EXTNAM オブジェクト 580

F

FCM
メモリーの要件 105
FETCH FIRST N ROWS ONLY 節
OPTIMIZE FOR N ROWS 節と共に使う 195
First Failure Data Capture (FFDC) トラップ・ファイル 793
First Occurrence Data Capture
FODC を参照 769
FODC
サブディレクトリー 775
詳細 769
ダンプ・ファイル 768
データの生成 781
トラップ・ファイル 794
プラットフォーム固有の 788

G

GPFS
トラブルシューティング 685
トレース 699
GPFS エラー・メッセージ 718

H

host 712
HP-UX
構成ベスト・プラクティス 56

I

IBM
サポート更新情報
サブスクライブ 804
ソフトウェア・サポートへのデータの送信 802
IBM Data Server
メッセージ 786
IBM Tivoli System Automation for Multiplatforms リカバリー・
リソース・マネージャー・デーモン
障害、ディスク使用量の増大、SQL30108N 714
IN (意図なし) 223
INLIST2JOIN 照会書き直し要求エレメント 432
INSPECT コマンド
CHECK 節 628

INSPECT コマンド (続き)
db2dart の比較 539
instance_name.nfy ログ・ファイル 753
IOCPs (入出力完了ポート)
AIX 135
IS (意図的共有) 223
ISV アプリケーション
ベスト・プラクティス 56
IX (意図的排他) ロック・モード 223
I/O
並列処理
管理 134
プリフェッチ 132

J

JDBC
アプリケーション
トレース機能の構成 587
分離レベル 182

L

Linux
構成
ベスト・プラクティス 56
DB2 データベース製品のリスト 551
locklist 構成パラメーター
ロックの細分性 221

M

maxappls 構成パラメーター
メモリー使用に与える影響 98
maxcoordagents 構成パラメーター 98

N

NOTEX2AJ 照会書き直し要求エレメント 432
NOTIN2AJ 照会書き直し要求エレメント 432
NS (スキャン共有) ロック・モード 223
numdb データベース・マネージャー構成パラメーター
メモリー使用に与える影響 98
NW (次キーの弱い排他) ロック・モード 223

O

ODBC
アプリケーション
トレース機能の構成 592
トレース 598
分離レベルの指定 182
OPTGUIDELINES エレメント
ステートメント・レベル 426

OPTGUIDELINES エレメント (続き)
global 415
OPTIMIZE FOR N ROWS 節 195
OPTPROFILE エレメント 414

P

PRDID パラメーター 580
ps コマンド
EXTNAM オブジェクト 580

Q

QRYOPT 一般要求エレメント 429

R

RECEIVE BUFFER 579
REGISTRY 一般要求エレメント 430
Reliable Scalable Cluster Technology (RSCT)
ピア・ドメイン
手操作によるクリーンアップ 736
トラブルシューティング 686
REOPT BIND オプション 208
REOPT 一般要求エレメント 430
REORG TABLE コマンド
オフラインで実行 145
reorg ユーティリティ
進捗のモニター 155
restart light
失敗 703, 709
REXX 言語
分離レベルの指定 182
RSCT (Reliable Scalable Cluster Technology)
ピア・ドメイン
手操作によるクリーンアップ 736
トラブルシューティング 686
RTS 一般要求エレメント 431
RUNSTATS コマンド
自動統計収集 477
統計のサンプリング 493
RUNSTATS ユーティリティ
サブエレメントについての情報 495
自動統計収集 482
収集される統計 467
パフォーマンスの改善 515, 660
runstats ユーティリティ
進捗のモニター 512

S

S (共有) ロック・モード
詳細 223
SARGable 述部
概要 258

SECCHK コマンド 580
 SELECT ステートメント
 出力の優先順位付け 213
 DISTINCT 節の除去 253
 SET CURRENT QUERY OPTIMIZATION ステートメント
 照会最適化クラスの設定 378
 SIX (意図的排他共有) ロック・モード 223
 Solaris オペレーティング・システム
 構成ベスト・プラクティス 56
 sortheap データベース構成パラメーター
 トラブルシューティング 653
 SQL コンパイラー
 プロセス詳細 246
 SQL ステートメント
 書き直し 250
 作成
 ベスト・プラクティス 190
 調整
 Explain 機能 319
 SELECT ステートメント 212
 SELECT ステートメントの制限 213
 分離レベル 182
 ヘルプ
 表示 812
 ベンチマーク 6
 Explain ツール 353
 SQLCA
 データのバッファ 579
 SQLCODE フィールド 579
 SQLCODE
 SQLCA 内のフィールド 579
 SQLJ
 分離レベル 182
 SRVNAM オブジェクト 580
 STMM
 セルフチューニング・メモリーを参照 107
 STMTKEY エレメント 419
 STMTKEY フィールド 388
 STMTMATCH エレメント 418, 419
 STMTPROFILE エレメント 417
 SUBQ2JOIN 照会書き直し要求エレメント
 XML スキーマ 433
 SYSTOOLS.OPT_PROFILE 表 450

T

TCP/IP
 ACCSEC コマンド 580
 SECCHK コマンド 580

U

U (更新) ロック・モード 223
 uDAPL 通信
 障害 716

UDF
 統計の入力 510
 UNIX
 DB2 データベース製品のリスト 551

W

WAITING_FOR FAILBACK 712

X

X (排他) ロック・モード 223
 XML スキーマ
 アクセス要求エレメント 434
 一般最適化ガイドライン 427
 グローバル OPTGUIDELINES エレメント 415
 結合要求エレメント 446
 現行最適化プロファイル 406
 照会書き直し最適化ガイドライン 431
 プラン最適化ガイドライン 433
 ACCESS アクセス要求エレメント 436
 accessRequest グループ 433
 computationalPartitionGroupOptimizationChoices グループ
 416
 DEGREE 一般要求エレメント 428
 DPFXMLMOVEMENT 一般要求エレメント 428
 generalRequest グループ 427
 HSJOIN 結合要求エレメント 448
 INLIST2JOIN 照会書き直し要求エレメント 432
 IXAND アクセス要求エレメント 438
 IXOR アクセス要求エレメント 441
 IXSCAN アクセス要求エレメント 442
 JOIN 結合要求エレメント 447
 joinRequest グループ 445
 LPREFETCH アクセス要求エレメント 443
 MQTOptimizationChoices グループ 415
 MSJOIN 結合要求エレメント 448
 NLJOIN 結合要求エレメント 449
 NOTEX2AJ 照会書き直し要求エレメント 432
 NOTIN2AJ 照会書き直し要求エレメント 432
 OPTGUIDELINES エレメント 426
 OPTPROFILE エレメント 414
 QRYOPT 一般要求エレメント 429
 REGISTRY 一般要求エレメント 430
 REOPT 一般要求エレメント 430
 rewriteRequest グループ 431
 RTS 一般要求エレメント 431
 STMTKEY エレメント 419
 STMTMATCH エレメント 418, 419
 STMTPROFILE エレメント 417
 SUBQ2JOIN 照会書き直し要求エレメント 433
 TBSCAN アクセス要求エレメント 443
 XANDOR アクセス要求エレメント 444
 XISCAN アクセス要求エレメント 445

XML データ
パーティション索引 89
XQuery コンパイラー
プロセス詳細 246
XQuery ステートメント
書き直し 250
分離レベル 182
Explain ツール 353

Z

Z (超排他) ロック・モード 223
ZRC 戻りコード 784

[特殊文字]

db2diag ログ・ファイル
解釈
情報レコード 762



Printed in Japan

SA88-4664-00



日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

問題判別およびデータベース・パフォーマンスのチューニング

