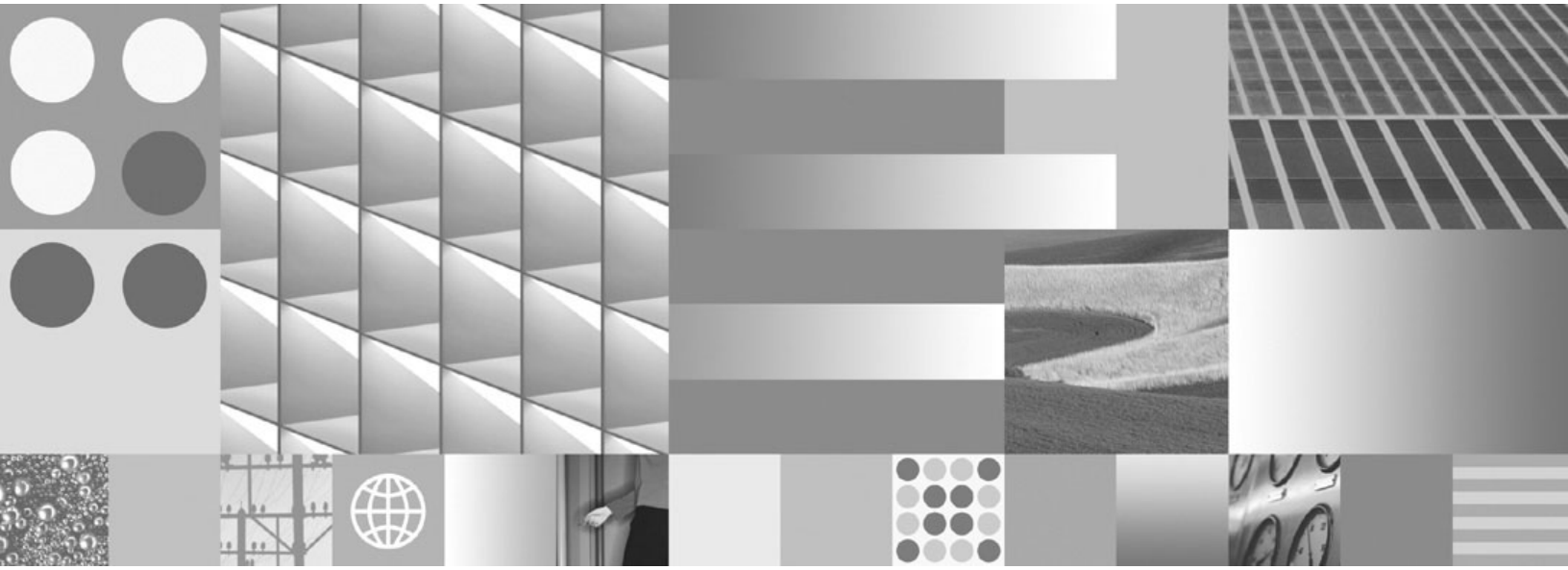
IBM DB2 9.7
for Linux, UNIX, and Windows

バージョン 9 リリース 7



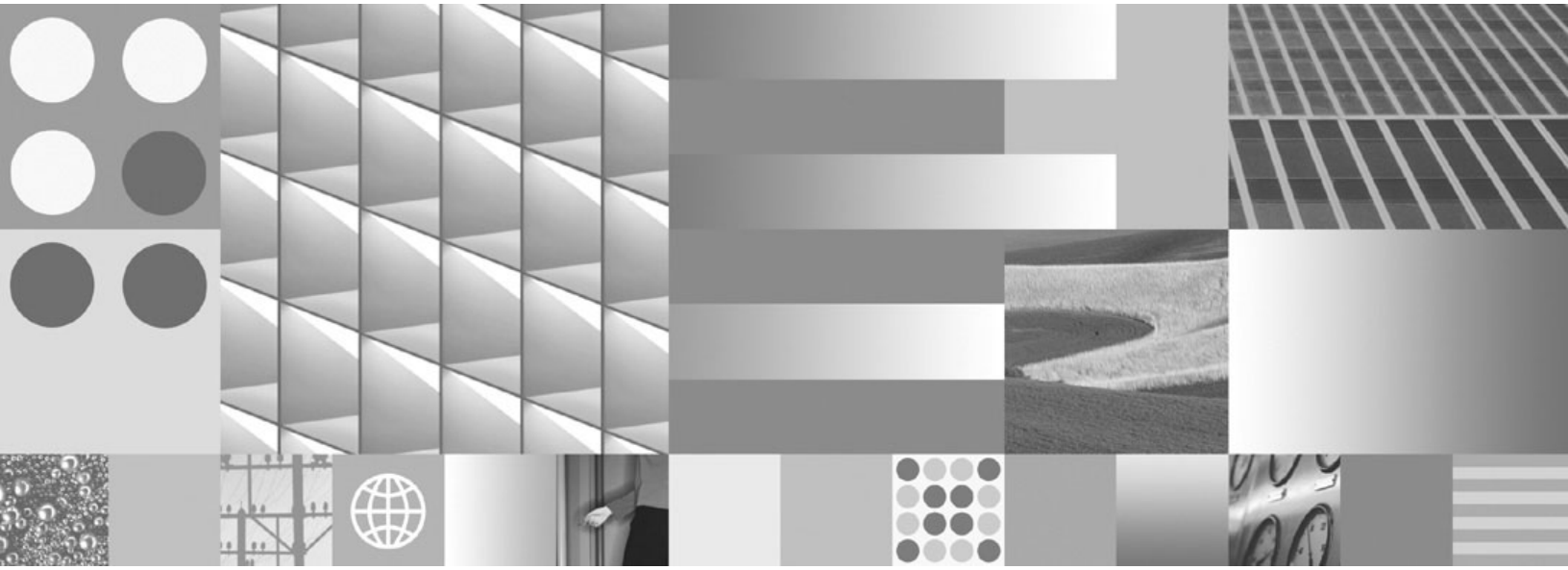
問題判別およびデータベース・パフォーマンスのチューニング

2009 年 11 月更新版

SC88-5871-01
(英文原典：SC27-2461-01)

IBM DB2 9.7
for Linux, UNIX, and Windows

バージョン 9 リリース 7



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

2009 年 11 月更新版

SC88-5871-01
(英文原典：SC27-2461-01)

ご注意

本書および本書で紹介する製品をご使用になる前に、 641 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。

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、www.ibm.com/shop/publications/order にある IBM Publications Center 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、www.ibm.com/planetwide にある IBM Directory of Worldwide Contacts 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： SC27-2461-01
IBM DB2 9.7 for Linux, UNIX, and Windows
Version 9 Release 7
Troubleshooting and Tuning Database Performance
Updated November, 2009

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 2009.11

目次

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 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 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と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、問題解決のための系統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用の程度

一般に、システムを調整することにより、その費用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ーの能力を高めたりすることでパフォーマンスを即座に改善できるように思えても、時間を取って障害がどこに存在しているのかを理解してください。費用をかけてディスク装置を追加しても、それを活用するだけの処理能力やチャネルがないことに気付く場合があります。
- **チューニングを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にない場合は、ユーザーの意見を聞いてパフォーマンスの問題と取り組むことができます。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か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。

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 1 つです。これはアプリケーション開発者とデータベース管理者 (DBA) の両方が関係するチームの作業です。

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、通常の環境でテスト・アプリケーションを実行することによって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ー時間の 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 か最悪のケースの SQL のいずれかにする必要があります。これらは以下で説明します。

典型的な SQL

典型的な SQL には、ベンチマーク・テスト中のアプリケーションの一般的操作で実行されるステートメントが含まれます。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

このカテゴリーに該当するステートメントには、次のものがあります。

- 頻繁に実行されるステートメント。
- 大量のデータを処理するステートメント。
- 時間が重要なステートメント。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。
- 結合の数が多いステートメント、またはアプリケーションの中でも最も複雑なステートメント。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、およびバッファー・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ー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ー (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 `sqllib` ディレクトリーの `bin` サブディレクトリー内に提供されています。このツールは、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、プロセッサー時間、バッファー・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ー・プールの初期化などのいくつかの開始アクティビティーが含まれるため、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ー・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、バッファー・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。バッファー・プールの内容は、無関係な他のデータをバッファー・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ー・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、次回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、またはプロセッサー時間を計算します。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、またはプロセッサー時間を計算します。

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     Iter.   Stmt    Timing         SQL Statement
Numbr    Numbr   Numbr   (hh:mm:ss.ss)
 002      05      01     00:00:01.34    CONNECT TO SAMPLE
 002      05      10     00:02:08.15    OPEN cursor_01
 002      05      15     00:00:00.24    FETCH cursor_01
 002      05      15     00:00:00.23    FETCH cursor_01
 002      05      15     00:00:00.28    FETCH cursor_01
 002      05      15     00:00:00.21    FETCH cursor_01
 002      05      15     00:00:00.20    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      20     00:00:00.84    CLOSE cursor_01
 002      05      99     00:00:00.03    CONNECT RESET
```

図 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 以外のデータは単に「あればよい」という以上に重要です。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、イベントとアクティビティーのモニター・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、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、UNIX ベースのシステムでは 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_SYS_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にまれなので、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・パフォーマンスの約 10 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、以下の約 10 のメトリックが含まれています。

- 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・アクティビティーの基本レベルの優れた測定結果が提供されます。

- データ、索引、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。

  ```
  100 * (POOL_DATA_L_READS – POOL_DATA_P_READS) / POOL_DATA_L_READS
  100 * (POOL_INDEX_L_READS – POOL_INDEX_P_READS) / POOL_INDEX_L_READS
  100 * (POOL_TEMP_DATA_L_READS - POOL_TEMP_DATA_P_READS) / POOL_TEMP_DATA_L_READS
  100 * (POOL_TEMP_INDEX_L_READS - POOL_TEMP_INDEX_P_READS)
    / POOL_TEMP_INDEX_L_READS
  ```

  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。

  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。

- 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。

```
(POOL_DATA_P_READS + POOL_INDEX_P_READS +
  POOL_TEMP_DATA_P_READS + POOL_TEMP_INDEX_P_READS)
  / TOTAL_COMMITS

(POOL_DATA_WRITES + POOL_INDEX_WRITES)
  / TOTAL_COMMITS
```

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。

- 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、この数値がとても大きくなる可能性があります。一方、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。
- 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_COMMITS
```

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オーバーヘッド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。

- 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_COMMITS
  ```

  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・アップするだけで、処理しやすい大きさになります。
- 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_COMMITS
  ```

  ほとんどの実動システムでデッドロックは比較的にまれなことですが、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、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。
- 1000 トランザクションごとのダーティー・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_COMMITS
  ```

  『ダーティー・スチール』は、バッファー・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、新しいバッファー・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。ダーティー・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。
- 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_COMMITS
  ```

  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は静的 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。

- 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_COMMITS
  ```

  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。
- 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) バッファー数は、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が示されます。種々のパーティションから受け取るバッファー数がかなり異なる場合、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一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ー・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 バッファーの送受信
- 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、より低いバッファー・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% 以上のバッファー・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、現行のバッファー・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ー・プール・ヒット率が 99% の場合、85% の現行のバッファー・プール・ヒット率は、異常であると考えられます。

ガバナー・ユーティリティー

ガバナーは、データベースに対して実行しているアプリケーションの動作をモニターし、ガバナー構成ファイルで指定した規則に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フィーチャーが導入されたことで、バージョン 9.7 では DB2 ガバナー・ユーティリティー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・ユーティリティー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・ユーティリティー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・ユーティリティーおよび 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す。デフォルトでは、ガバナーを開始すると、パーティション・データベースの各データベース・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。

ガバナーの開始と停止

ガバナー・ユーティリティーは、データベースに接続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。

重要: DB2 バージョン 9.5 で導入された新しい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。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『DB2 ガバナーと Query Patroller が推奨されなくなった』を参照してください。

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* 権限を持っていなければなりません。

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。

1. 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生じた場合には、ガバナーを停止しないで構成ファイルを編集します。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ー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. **applname**: 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 `rowssel 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、

rowssel 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 rowssel 5000 action force;

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

緩やかな規則で、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 rowssel 5000 action force;

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

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

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 rowssel 1000000 rowsread 5000000;

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

desc 'Slow down a subset of applications.'
applname jointA, jointB, jointC, quryA
setlimit cpu 3 locks 1000 rowssel 500 rowsread 5000;

desc "Have 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 rowssel 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, tpvG
setlimit cpu 600 rowssel 120000 action force;

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

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 rowssel -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 rowssel 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・ユーティリティーでは、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ー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ー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・スイッチを非活動化した場合には、ガバナーが作業単位経過時間に基づいてアプリケーションを管理する機

能に影響を与えます。ガバナーはモニターを使って、システムについての情報を収集します。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・パーティ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。 UNIX ベースのシステムでは、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。

- UNIX ベースのシステムでは、**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® 5.3 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 機能を持っていなければなりません。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
```

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・スイッチを非活動化した場合には、ガバナーがステートメント経過時間に基づいてアプリケーションを管理する機能に影響を与えます。ガバナーはモニター

を使って、システムについての情報を収集します。データベース・マネージャー構成ファイルでスイッチをオフにすると、インスタンス全体でオフになり、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きに起こる可能性のある、ファイル・ロッキングのボトルネックを防ぐことができます。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に書き込まれます。 NICE レコードは、以下のようなフォーマットになります。

```
<appl_name> <auth_id> <appl_id> <nice_value> <cfg_line>
  <restriction_exceeded>
```

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ルする規則が位置する、ガバナー構成ファイル内の行番号を指定します。

restriction_exceeded

規則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 sqlefrce API が呼び出されたが、正の SQLCODE が戻された。
- スナップショット呼び出しが 1611 以外の正の SQLCODE を戻した (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。
- UNIX ベースの環境で、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/sqllib/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。

重要: DB2 バージョン 9.5 で導入された新しい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。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『DB2 ガバナーと Query Patroller が推奨されなくなった』を参照してください。

ガバナーを停止するには、*sysadm* または *sysctrl* 権限を持っていなければなりません。

ガバナーを停止するには、STOP オプションを指定して db2gov コマンドを使用します。

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、共用メモリーおよびセマフォーを使用して通信します。リモート・クライアントは、名前付きパイプ (NPIPE) 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。

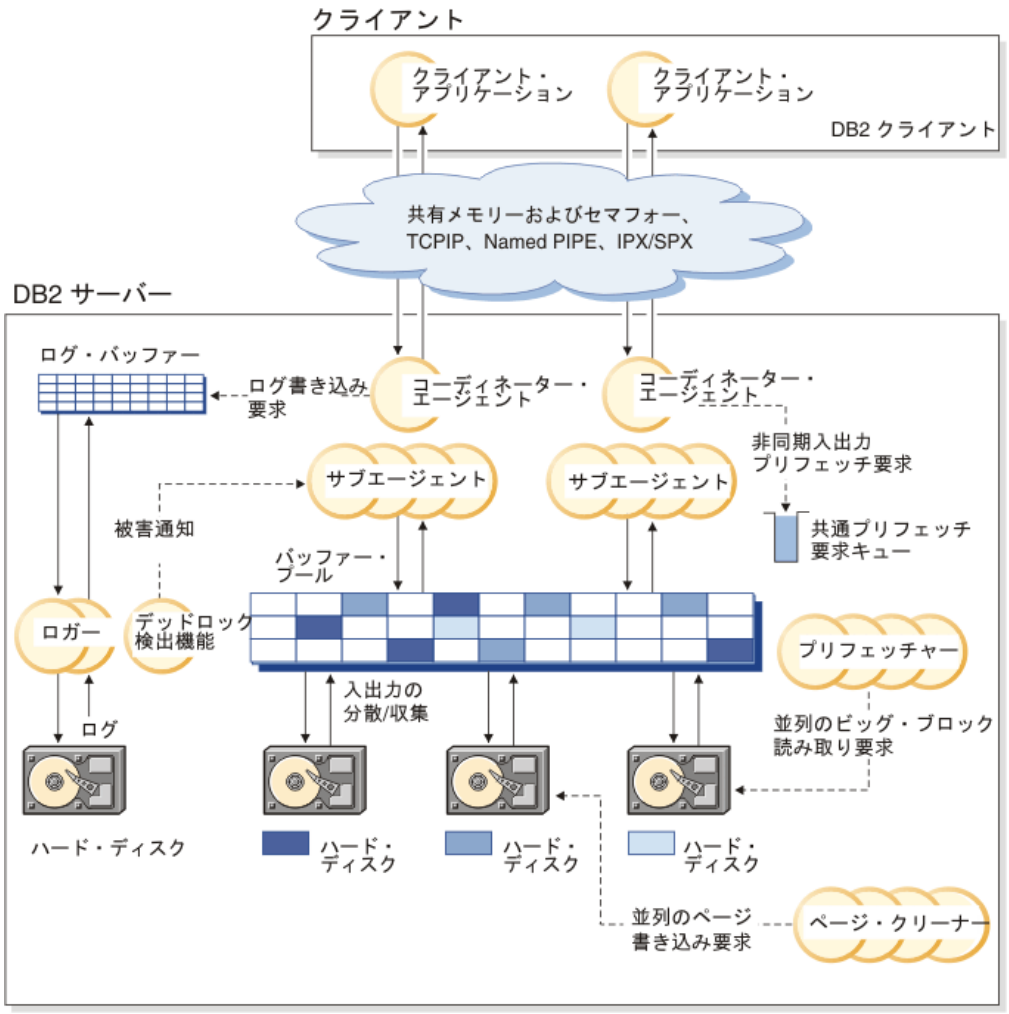


図 3. クライアント接続とデータベース・サーバーのコンポーネント

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ーがある場合、またはそのマシンがパーティション・データベース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、複数のプロセッサーを活用することができます。

すべてのエージェントおよびサブエージェントは、プール・アルゴリズム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。

バッファー・プールは、データベース・サーバー・メモリーのエリアであり、ここで、ユーザー・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、バッファー・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。

バッファー・プールの構成は、プリフェッチャーおよびページ・クリーナー EDU と共に、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ー・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ー・プールに移動するのを待機しなければなりません。アプリケーションのエージェントは、非同期読み取り先行要求を共通プリフェッチ・キューに送信します。使用可能になると、プリフェッチャーは大きなブロックを使用してこれらの要求をインプリメントするか、または読み取り入力操作を分散させてディスクからバッファー・プールに要求されたページを移動します。データ・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、データをバッファー・プールからディスクに戻します。ページ・クリーナーはアプリケーション・エージェントから独立したバックグラウンド EDU です。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ー・プール内に確保されます。

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

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・マネージャーおよび関連したコンポーネントの対話方法を理解するのに役立ち、生じる可能性のある問題のトラブルシューティングにも役立ちます。

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。

アクセスされるそれぞれのデータベースごとに、異なる 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 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 (38 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・マネージャーを、アプリケーション、ストアード・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。

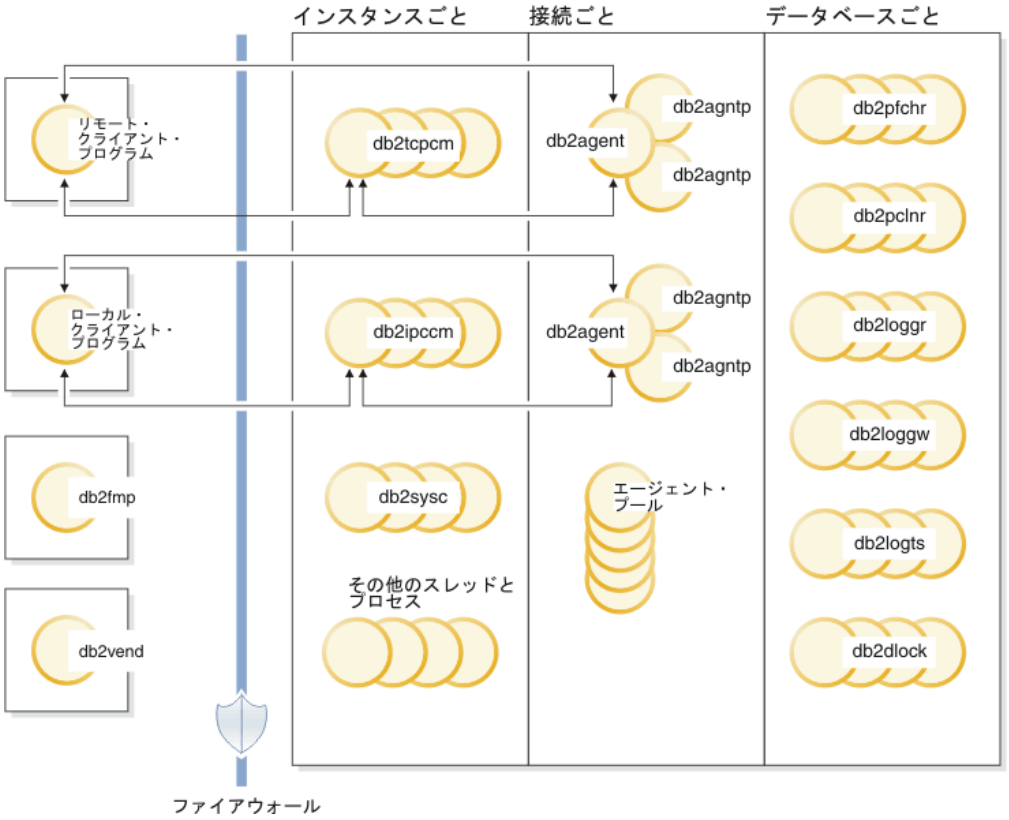


図 4. DB2 データベース・システムのプロセス・モデル

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
- db2tcpcm、TCP/IP 接続用
- db2tcpdm、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 (それぞれ 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行して行います。

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操作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操作するサブエージェントからの要求に対して使用可能です。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全く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。

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・プロシージャー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

これは、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。

- 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。
- 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
- db2loggw、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
- db2pfchr、バッファー・プール・プリフェッチャー用
- db2pclnr、バッファー・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。
- db2wlmd、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* の部分は以下のようになります。

    - 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 の部分は以下のようになります。

    - 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ー・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 % の部分は以下のようになります。
  - *%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。

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)
- db2cart、ログ・ファイルのアーカイブ用 (**userexit** データベース構成パラメーターが有効な場合)
- 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
- db2fmtlg、ログ・ファイルのフォーマット設定用 (**logretain** データベース構成パラメーターが有効で、**userexit** データベース構成パラメーターが無効な場合)
- db2licc、インストール済みの DB2 ライセンスを管理する
- db2panic、エージェント限度が特定のデータベース・パーティション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 (パーティション・データベース環境でのみ使用)
- 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
- db2srvlst、 DB2 for z/OS® などのシステムのアドレスのリストを管理する
- 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。
- db2thcln、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

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が、データベース・マネージャーおよびデータベース・グローバル・リソース (バッファー・プールなど) は他のエージェントと共用します。トランザクションが完了すると、アクティブ・コーディネーター・エージェントは非アクティブ・エージェントになる場合があります。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ので、アイドル・エージェントをクライアントに対してアクティブにできる場合、CONNECT および ATTACH のパフォーマンスは向上します。

あるサブエージェントがアプリケーションの作業を行うとき、そのサブエージェントはアプリケーションに関連付けられます。割り当てられた作業が完了すると、サブエージェントはエージェント・プールに入れられますが、元のアプリケーションとの関連付けはそのまま残されます。そのアプリケーションが追加の作業を要求した場合、データベース・マネージャーは新規エージェントを作成する前に、まずアイドル・プール内にそのアプリケーションと関連するエージェントがないか検査します。

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 の関係を確立します。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間の多対 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にのみ適用されます。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用する。
- 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用する。
- 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。

46 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。

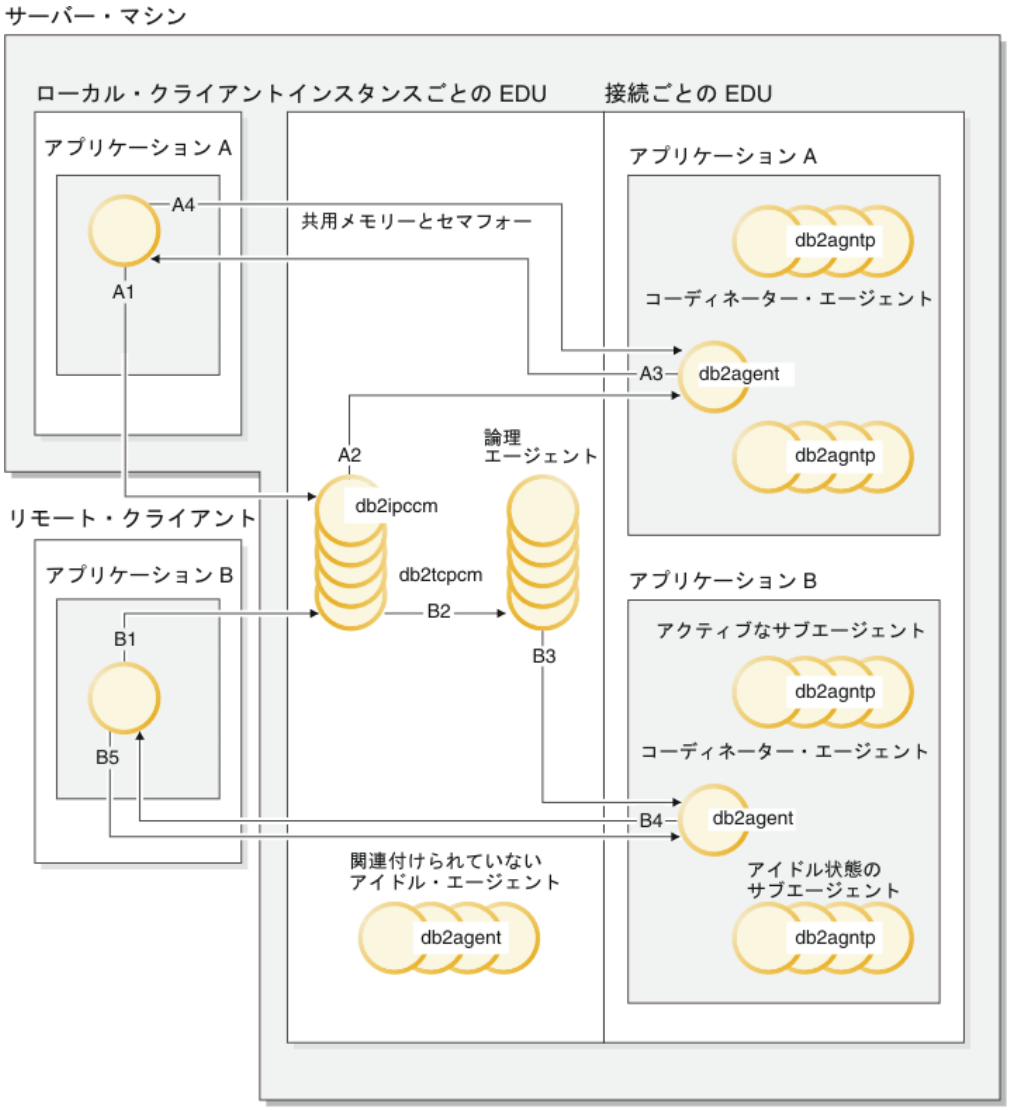


図 5. クライアント/サーバー処理モデルの概要

- 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用メモリー通信を確立します。
- A4 で、ローカル・クライアントのアプリケーションは、データベースに接続します。
- B1 では、リモート・クライアントが db2tcpcm 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、db2tcpcm 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。

48 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) の共用プール、そして独自のロガー (db2loggr) およびデッドロック検出機能 (db2dlock) があります。

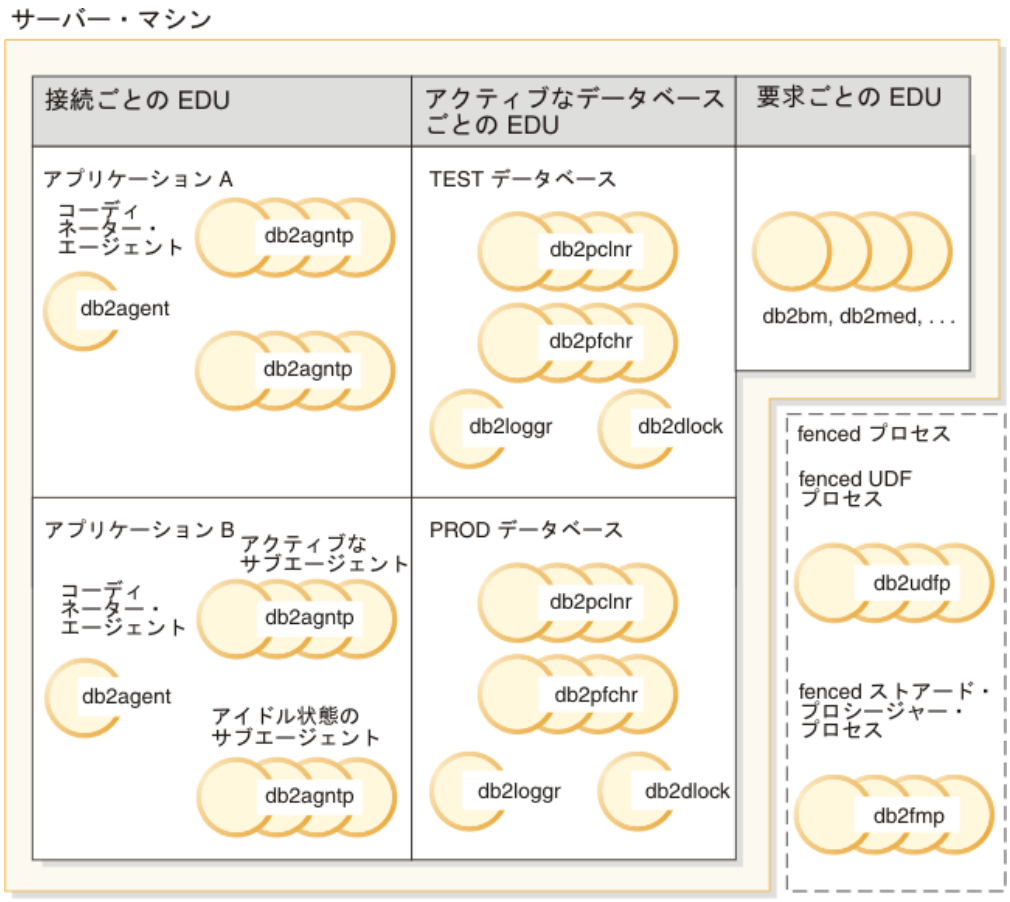


図 6. データベース・サーバーにおける EDU

図には示されていませんが、fenced ユーザー定義関数 (UDF) およびストアード・プロシージャーは、その作成と破棄に関連するコストを最小限に抑える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、ストアード・プロシージャー・プロセスを、次のプロシージャー呼び出しで再使用できます。

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

49 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。

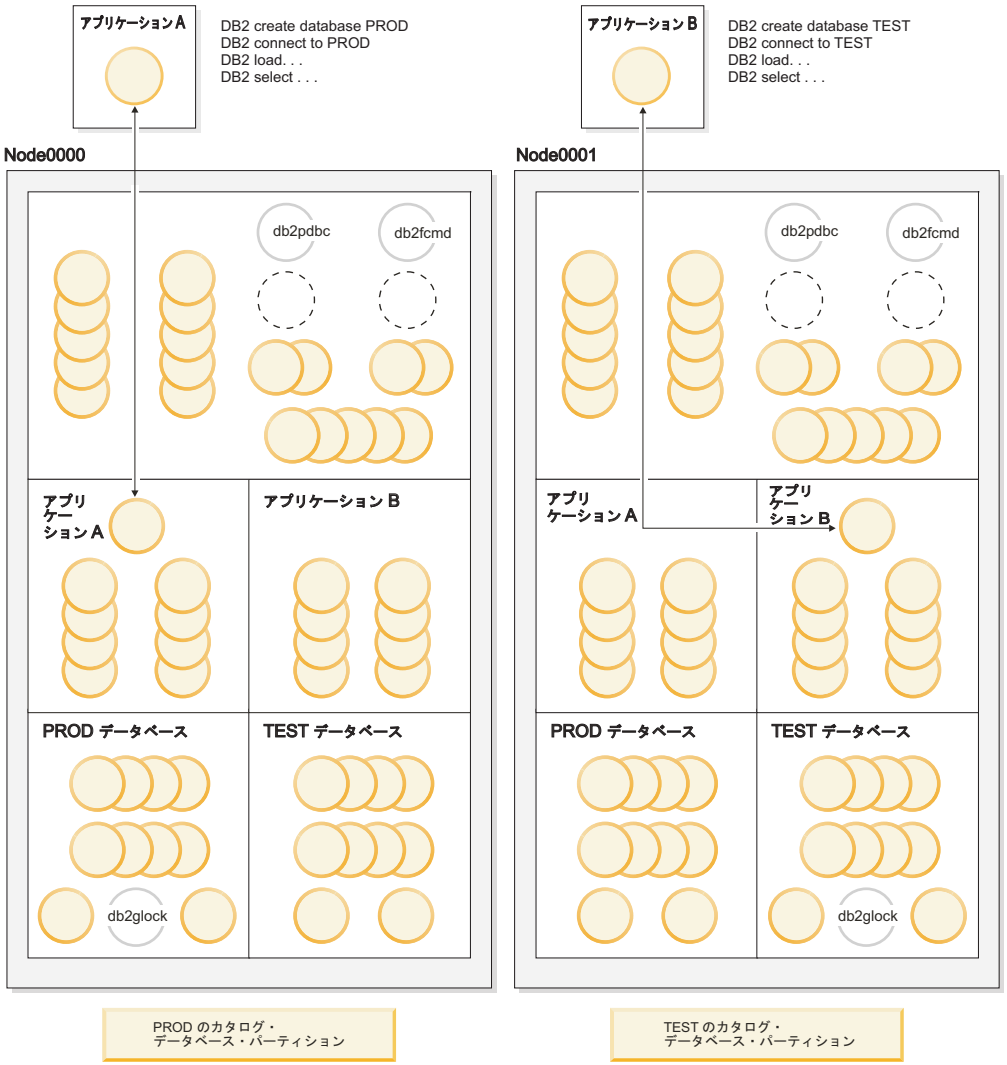


図 7. 複数のデータベース・パーティションのプロセス・モデル

複数データベース・パーティション環境において、CREATE DATABASE コマンドが発行されたデータベース・パーティションは、*カタログ・データベース・パーティション* と呼ばれます。システム・カタログ表は、このデータベース・パーティションに保管されます。システム・カタログは、そのデータベース内のオブジェクトについてのすべての情報のリポジトリーです。

図 7 で示すとおり、アプリケーション A は Node0000 で PROD データベースを作成するため、PROD データベースのカタログもこのデータベース・パーティションに作成されます。同様に、アプリケーション B が Node0001 に TEST データベースを作成するため、 TEST データベースのカタログがこのデータベース・パーティションに作成されます。ご使用の環境のデータベース・パーティション間で各データベースのカタログに関連する付加的なアクティビティーの均衡を保つためには、異なる複数のデータベース・パーティションにそれぞれのデータベースを作成するのが得策です。

ここには、インスタンスに関連付けられる追加の EDU (db2pdbc および db2fcmd) があり、これらは、複数パーティション・データベース環境の各データベース・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ー (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、共用メモリーおよびセマフォーを使用するように最適化されます。

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も削減され、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のある環境では、システム・リソースをより効率的に使用するために、接続コンセントレーターを使用可能にすることができます。このフィーチャーは、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が、250 を超えることは決してありません。トランザクションは短期間です。

  このワークロードを処理するために、管理者は以下のようなデータベース・マネージャー構成パラメーターを設定できます。

  - **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ことを常に行わなければならないため、パフォーマンスが低下します。

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・アクティビティーを処理するために、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ーは、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ーがあります。

- データベース入出力サーバーまたは入出力プリフェッチャー
- データベース非同期ページ・クリーナー
- データベース・ロガー
- データベース・デッドロック検出機能
- 通信および IPC listener
- 表スペース・コンテナーのリバランサー

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い、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、プロセッサー・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と同様の複雑さの 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 つのプロセッサー・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 (非共用) 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ー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 秒当たりに最大 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、プロセッサー・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のある AIX パラメーターは比較的わずかです。こうした推奨値の目的上、AIX レベルは 5.3 以降であると想定しています。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。これらは、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 `MAX( NUM_IOCLEANERS x 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 x *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

Linux システムを DB2 サーバーとして使用する場合、一部の Linux カーネル・パラメーターを変更しなければならないことがあります。 Linux 配布版が変更されることがあるため、およびこの環境がとても柔軟であるため、基本的な Linux インプリメンテーションで検証する必要のある最も重要な設定の一部のみを考慮します。

64 ビット・システム上の **SHMMAX** (共有メモリー・セグメントの最大サイズ) を最小の 1 GB (1 073 741 824 バイト) に設定しなければなりません。一方、パラメーター **SHMALL** はデータベース・サーバーで使用可能なメモリーの 90% に設定する必要があります。 **SHMALL** は、デフォルトで 8 GB です。他の重要な Linux カーネル構成パラメーターと DB2 におけるその推奨値は、以下のとおりです。

- **kernel.sem** (4 つのカーネル・セマフォー設定値 SEMMSL、SEMMNS、SEMOPM、および SEMMNI を指定する): 250 256000 32 1024
- **kernel.msgmni** (メッセージ・キュー ID の数): 1024
- **kernel.msgmax** (メッセージのバイト単位での最大サイズ): 65536
- **kernel.msgmnb** (メッセージ・キューのバイト単位でのデフォルト・サイズ): 65536

DB2 データベース・パーティション・フィーチャー

通常、DB2 データベース・パーティション・フィーチャー (DPF) を使用するとの判断は、純粋なデータ量に基づくものではなく、むしろワークロードに基づきます。一般的な指針として、ほとんどの DPF デプロイメント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。 DPF 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 (ESE) 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、DPF 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 つのプロセッサー・コアがあります。例えば、*n* 個のプロセッサー・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)、プロセッサー・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、UCA500R1_NO 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、文化的に正しい照合 (UCA500R1_* など) 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。
- 文化的に正しい照合には、正規化されたものと非正規化の両方があります。正規化照合 (UCA500R1_NO など) には誤った形式の文字を処理するための追加検査がありますが、非正規化照合 (UCA500r1_NX など) にはありません。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、非正規化の文化的に正しい照合でさえコストはとても大きくなります。
- 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。

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、システムの特性に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、この量はバッファー・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 (バッファー・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。

偏ったデータ分布のあるパーティション・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用できず、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、共用の 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・コンテナーに対して一度に 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、コンテナー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に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ー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、すべてのコンテナー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、ご使用のコンテナーが 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、エクステント・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ー数によっても異なります。

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 Note) 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 - DB6: DB2 9.5 Standard Parameter Settings」があります。

DB2 DPF フィーチャーを使用する場合、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 DPF 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 DB2 DPF インストールに比べて異なることになります。 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 個のプロセッサー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、バッファー・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 AUTOCONFIGURE を CREATE DATABASE コマンドのオプションとして指定します。推奨値は表示することができ、また CREATE DATABASE コマンドの APPLY オプション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。
- 構成アシスタントを使用すると、データベース・オブジェクトの構成と保守、新規オブジェクトの追加、アプリケーションのバインド、データベース・マネージャー構成パラメーターの設定、および構成情報のインポートとエクスポートが可能です。構成アシスタントを開くには、db2ca コマンドを起動します。インスタ

ンス構成で構成アシスタントを使用すると、データベース・マネージャー構成パラメーターの設定、DB2 レジストリー変数の設定、他のインスタンスの構成、または構成の初期化に役立ちます。

- 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。

  ```
  OVERHEAD = ミリ秒単位の平均シーク時間
             + (0.5 * 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 / 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 であるとします。回転待ち時間の公式を使用すると、次のようになります。

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

  この値は、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もりに使用できます。

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

  ```
  TRANSFERRATE = (1 / spec_rate) * 1000 / 1024000 * 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 であるとします。この場合は以下のようになります。

```
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・コンテナー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ーの数で除算します。上記の公式の *spec_rate* をこの計算結果で置き換えます。例えば、4 つのコンテナー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、コンテナー当たり 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ーは、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に設定できます。デフォルトのプリフェッチ・サイズは **dft_prefetch_sz** データベース構成パラメーターの値によって決まります。このパラメーター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。

データベース設計

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るようになります。

それぞれのデータ・ページに入るレコードの数は、データ・ページのサイズやレコードのサイズによって異なります。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、標準の表の場合に、データ・ページで 500 ページごとにフリー・スペース制御レコード (FSCR) があるとします (66 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。

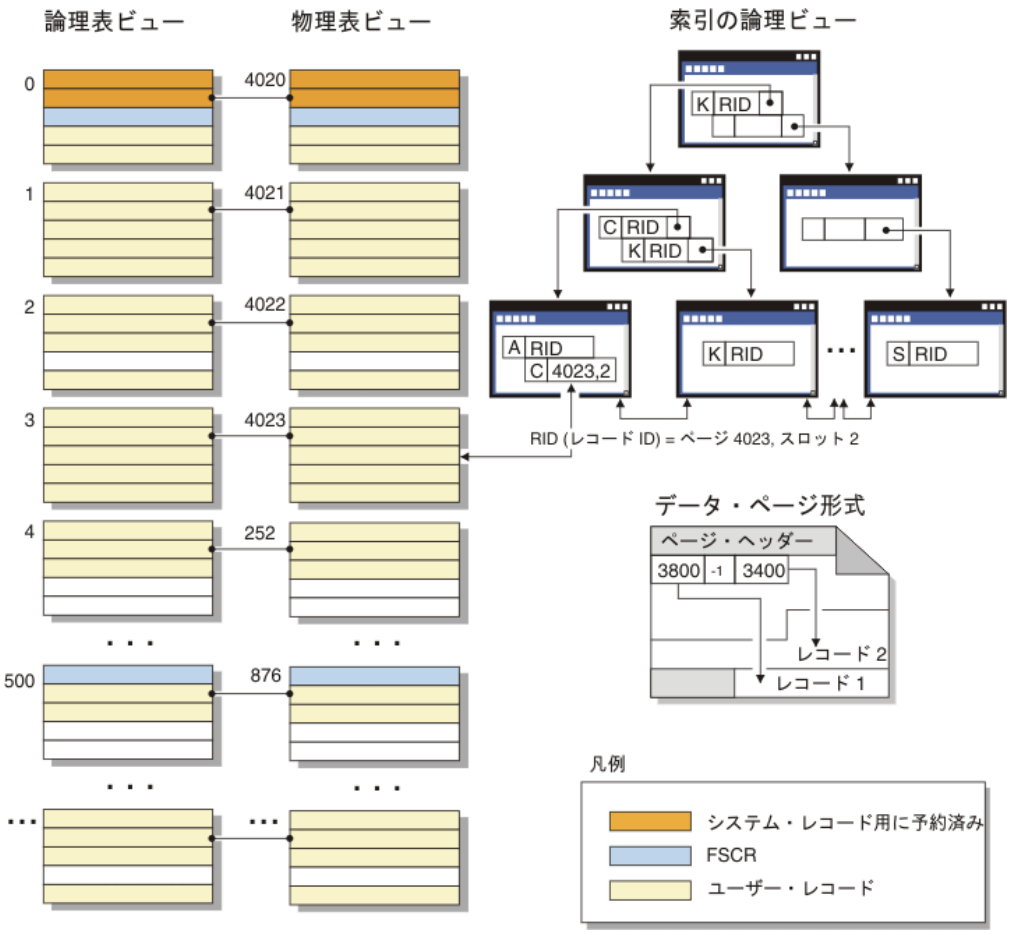


図 8. 標準の表の論理表、レコード、および索引構造

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ーの数は固定しておらず、キーのサイズによって異なります。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 (67 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。

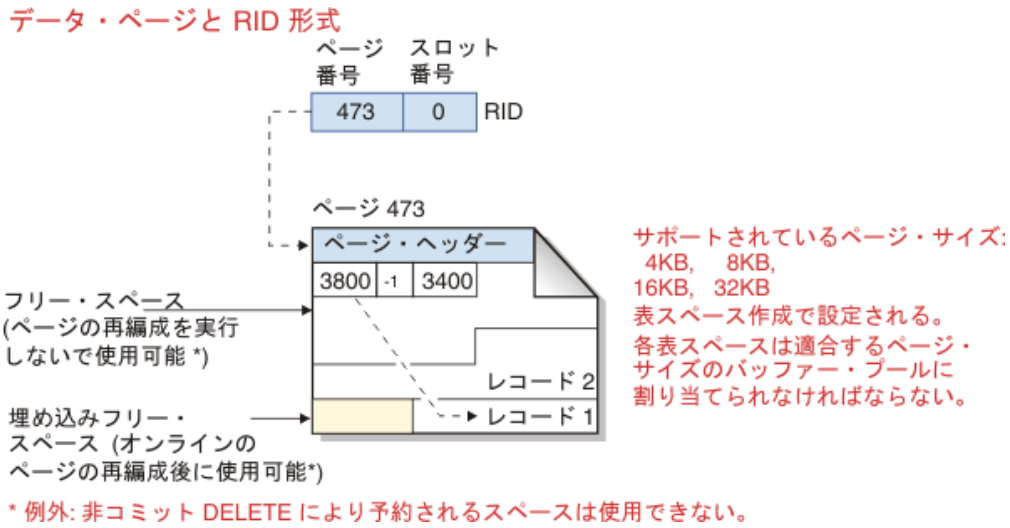


図 9. データ・ページとレコード ID (RID) 形式

表ページが再編成されるとき、レコードが物理的に削除された後でページに残された埋め込みフリー・スペースは、使用可能なフリー・スペースに変換されます。

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の数を増やすことができます。ただし、同時に索引スペースの要件も増やすことになり、組み込まれた列が頻繁に更新される場合には、索引の保守にかかるコストも増える可能性があります。組み込み列更新の保守にかかるコストは、キー列の更新の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 表のための表および索引管理

マルチディメンション・クラスタリング (MDC) 表の表および索引の編成は、標準の表編成と同じ論理構造に基づいています。

標準の表と同じく、MDC 表はデータの行を含むページに編成されて、列に分割されます。各ページの行はレコード ID (RID) によって識別されます。しかし、MDC 表のページはエクステント・サイズのブロックにグループ化されます。例えば、 69 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。

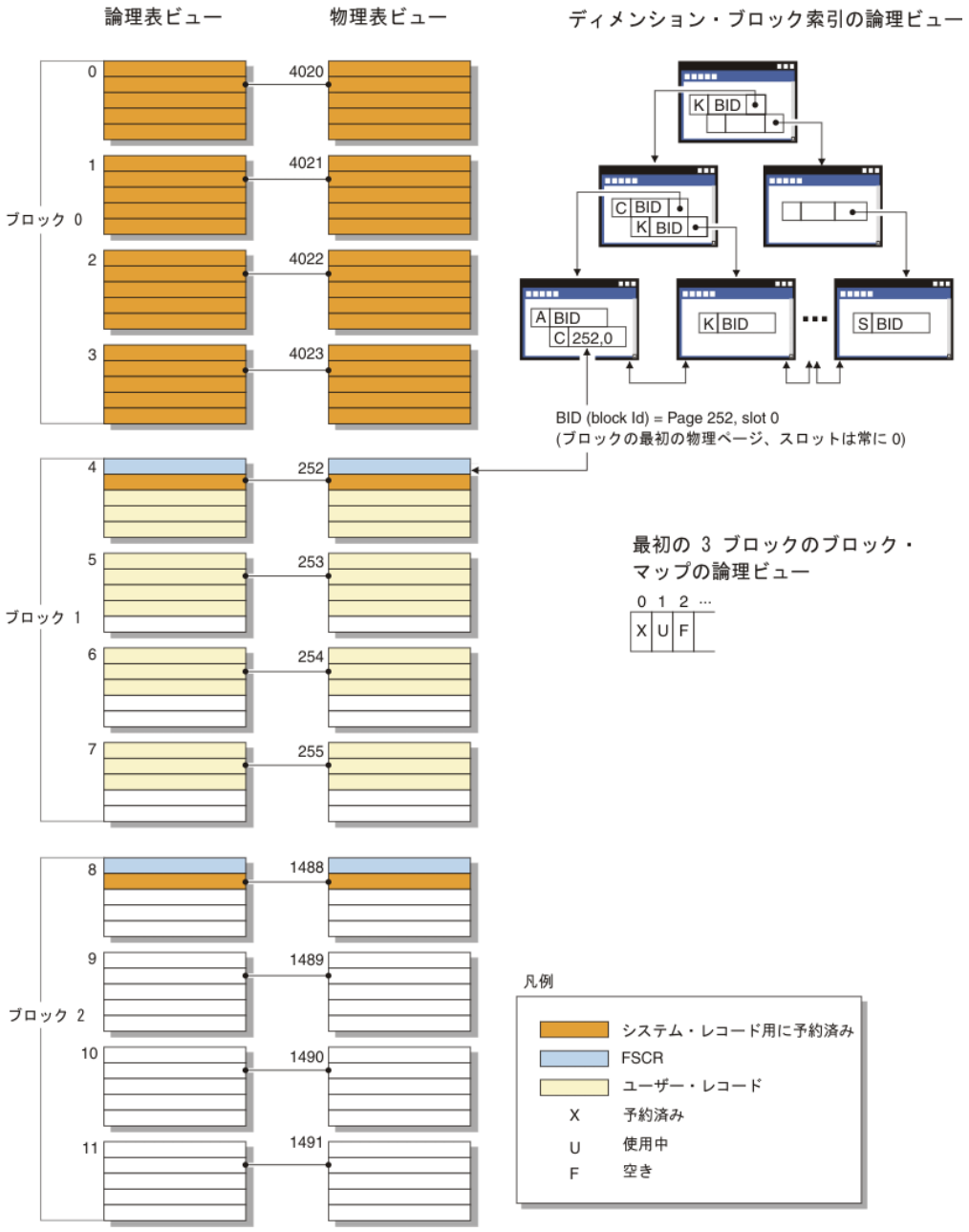


図 10. MDC 表の論理表、レコード、および索引構造

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ーの際にクラスタリングを保守するために使用されます。

オプティマイザーは、特定の照会に最適のアクセス・プランを判別する際にディメンション・ブロック索引を使用するアクセス・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、オプティマイザーはディメンション・ブロック索引を使用して、これらの値を含むエクステントを識別し、そこからフェッチします。エクステントはディスク上の物理的に連続したページなので、これにより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。

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、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。

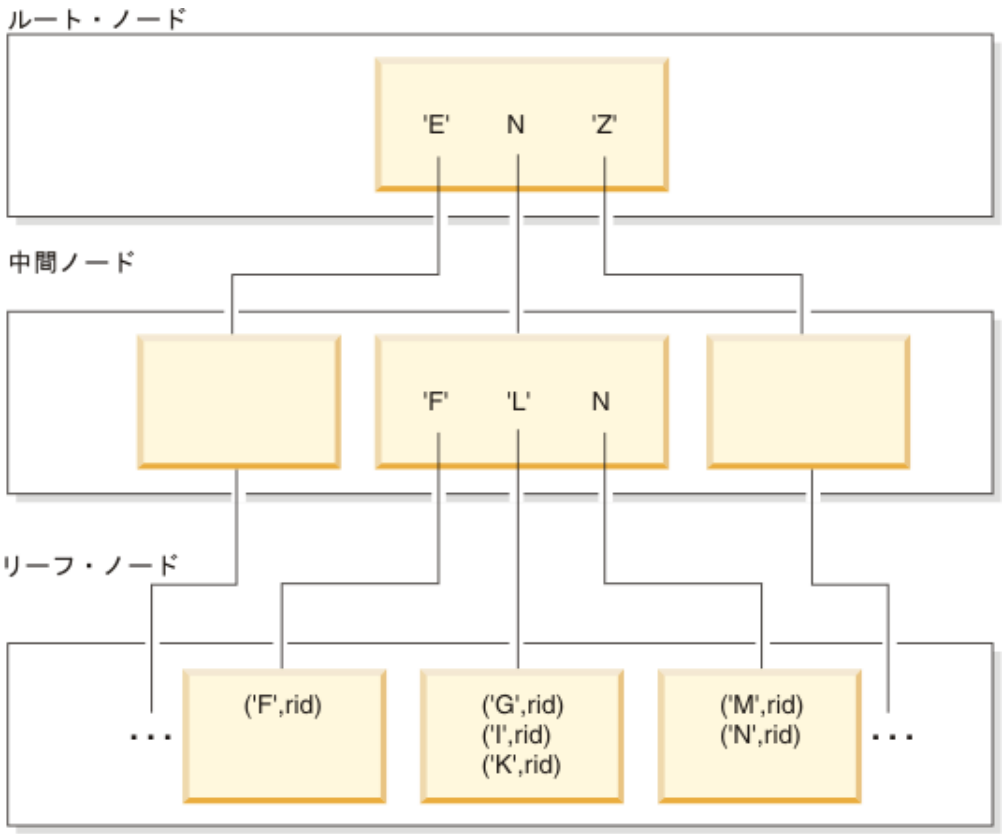


図 11. B+ ツリー索引の構造

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ーが保管されます。ルート・ノードとリーフ・ノードの間のレベルは、*中間ノード* と呼ばれます。

特定の索引キー値を検出するとき、索引マネージャーは、ルート・ノードから開始して、その索引ツリーを検索します。ルート・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検索します。ルート・ノード・キーは特定の中間ノードを指しています。索引マネージャーは必要とする索引キーを持つリーフ・ノードが見つかるまで、各中間ノードでこの手順を行います。

70 ページの図 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ーが入っている場合もあります。こうしたポインターによって、索引マネージャーはどちらの方向にでもリーフ・ノードをスキャンして、範囲内の 1 つの値を見つけた後、値の範囲を検索することができます。いずれの方向にもスキャンを実行する機能は、ALLOW REVERSE SCANS 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。索引内の可変長列ごとに、1 つの追加のバイトが列の値の実際の長さを保管します。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。

索引のクリーンアップおよび保守

索引の作成後、索引をコンパクトに保ち、十分に整理しないと、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、以下に説明するように、キーが削除済みとマークされた時点で物理的に除去されます。
- REORGCHK 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 ONLY 節を使っていつ 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。

  クリーンアップ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。データ・パーティション表の場合、REORG INDEXES...ALL コマンド上に ALLOW WRITE ACCESS 節を指定できるのは CLEANUP ONLY 節も指定する場合です。

  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 節を使用してデータ・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。

以下の状況では、削除済みとマークされた索引キーがクリーンアップされます。

- その後の挿入、更新、または削除アクティビティー中

  キーの挿入中、クリーンアップすればページ分割の必要がなくなり、索引サイズの増加を防げるような場合には、削除済みとマークされてコミット済みと認識されたキーがクリーンアップされます。

  キーの削除中、ページ上のすべてのキーが削除済みとマークされた場合には、すべてのキーが削除済みとマークされ、それらの削除がすべてコミット済みになった別の索引ページを見つけるよう試行されます。そのようなページが見つかると、索引ツリーから削除されます。キーを削除するとき、表に対する X ロックが存在すれば、キーは単に削除済みとマークされる代わりに、物理的に削除されます。物理的削除の際、削除済みとマークされてコミット済みと認識されたキーが同じページ上に存在すれば、それらもすべて除去されます。
- CLEANUP オプションを使って REORG INDEXES コマンドを実行するとき

  CLEANUP ONLY PAGES オプションを指定すると、すべてのキーが削除済みとマークされてコミット済みと認識されている索引ページが検索されて、解放されます。

  CLEANUP ONLY ALL オプションを指定すると、すべてのキーが削除済みとマークされてコミット済みと認識されている索引ページが解放されるだけでなく、未削除のレコード 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」(論理的デタッチ済み) 、「D」 (マテリアライズ照会表のようなデタッチ従属表を持つデ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とき (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とき)

  索引を再作成するユーティリティーには、以下のものがあります。
  - REORG INDEXES (いずれの CLEANUP オプションも使用しない)
  - REORG INDEXES (ON DATA PARTITION 節も使用する)
  - REORG TABLE (ON DATA PARTITION 節も使用する)
  - REORG TABLE (INPLACE オプションを使用しない)
  - IMPORT (REPLACE オプションを使用する)
  - LOAD (INDEXING MODE REBUILD オプションを使用する)

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ので、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ー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、ユーティリティー ID が同じになる場合がある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に、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、SYSPROC.ADMIN_GET_TAB_INFO_V95 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 DETAILS 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・ページにある使用済みスペースの最小量としてユーザー定義可能な限界値により、オンライン索引デフラグが使用可能になります。

リーフ・ページから索引キーが削除された場合にこの限界値を超えていると、索引の隣接するリーフ・ページがチェックされ、2 つのリーフ・ページをマージできるかどうか判別されます。ページ上に十分なスペースがあり、2 つの隣接するページをマージできるのであれば、バックグラウンドで即時にマージが行われ、生成された空の索引リーフ・ページが削除されます。

既存の索引が、オンラインでのマージ機能を必要とする場合、その索引をいったんドロップしてから CREATE INDEX ステートメントに指定した MINPCTUSED 節を使って再作成することが必要です。隣接する索引のリーフ・ページをマージすることが目標なので、MINPCTUSED の値は 50 より小さくすることをお勧めします。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。

索引のノンリーフ・ページは、オンライン索引デフラグ中にマージされません。しかし、空のノンリーフ・ページは削除され、同じ表上の他の索引による再利用が可能になるようにされます。これらのノンリーフ・ページをデータベース管理スペース (DMS) ストレージ・モデルの他のオブジェクト用に解放したり、システム管理スペース (SMS) ストレージ・モデルのディスク・スペースを解放したりするには、表または索引の完全な再編成を実行します。それにより、可能な限り小さい索引が作成されます。索引内のレベル数は、オンライン索引デフラグ時には減りません。

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が効果的です。しかし、キー削除中に表に X ロックがない場合、キーは削除のマークを付けられますが、物理的には索引ページから除去されず、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 ONLY ALL オプションを指定して呼び出します。マージされたページに少なくとも PCTFREE フリー・スペースが残る場合、2 つの隣接するリーフ・ページがマージされます。 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。

照会オプティマイザー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には、runstats ユーティリティーを実行して、新しい索引統計を収集してください。runstats ユーティリティーは定期的なインターバルで実行し、統計を現行のものに保持する必要があります。索引についての最新の統計がないと、オプティマイザーは照会に対する最適のデータ・アクセス・プランを判別することができません。

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、db2advis コマンドを使用して設計アドバイザーを起動します。

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。

オプティマイザーは、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て済みます。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、オプティマイザー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、オプ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 (db2advis 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を意味します。
  - 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 列が多くの重複値を含んでいる場合を除き、WORKDEPT に対する索引があれば 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ャンのほうが、逆方向のスキャ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。
    - 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ー (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ー・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、バッファー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ーも少ないため、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ー化できないことに注意してください。クラスタリング索引を作成した後で、追加の索引を作成してください。

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ー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、範囲クラスター表の場合を除き、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 ユーティリティーを使用します。

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。

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

パーティション表上の索引は、非パーティション表上の索引と同様に作動しますが、パーティション化索引か非パーティション化索引かに応じて、異なるストレージ・モデルを使用して保管されます。

通常の非パーティション表の索引はすべて、共用索引オブジェクトにありますが、パーティション表上の*非パーティション化索引* 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、単一の表スペース内の独自の索引オブジェクトに作成されます。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。

パーティション化索引 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・パーティションのすべての索引パーティションは、同じ索引オブジェクトにあります。

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。

非パーティション化索引の利点には、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。

パーティション化索引の利点には、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の B ツリー構造により、以下の結果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なときに、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化索引と非パーティション化索引の両方のスキャンに使用できますが、より効果的なのはパーティション化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、対応するデータ・パーティション専用のキーが含まれているためです。これにより、非パーティション化索引へ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て済む場合があります。

非パーティション化索引は索引列上の順序を常に保持しますが、パーティション化索引は、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。索引の作成中に表に行われた変更は、そのような索引が作成された後で、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。パーティション化索引の場合、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。

パーティション化索引サポートは、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。非パーティション化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。これは非パーティション化索引の保守、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。非パーティション化索引の保守は、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。パーティション化索引を使用して、この保守のコストを回避してください。

86 ページの図 12 は、1 つのパーティション表上の 2 つの非パーティション化索引を示しており、それぞれの索引は別々の表スペースにあります。

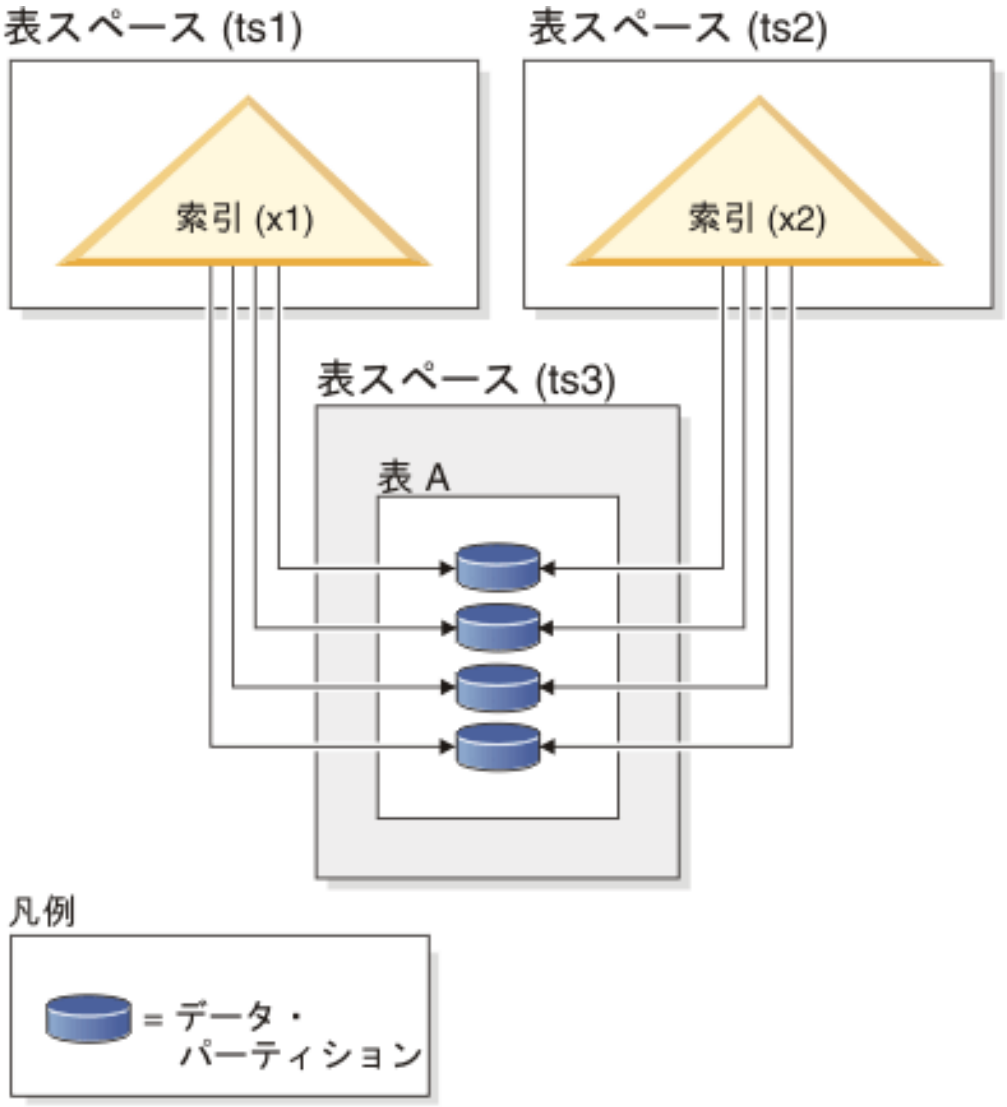


図 12. パーティション表上の非パーティション化索引

87 ページの図 13 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、パーティション表上のパーティション化索引を示しています。

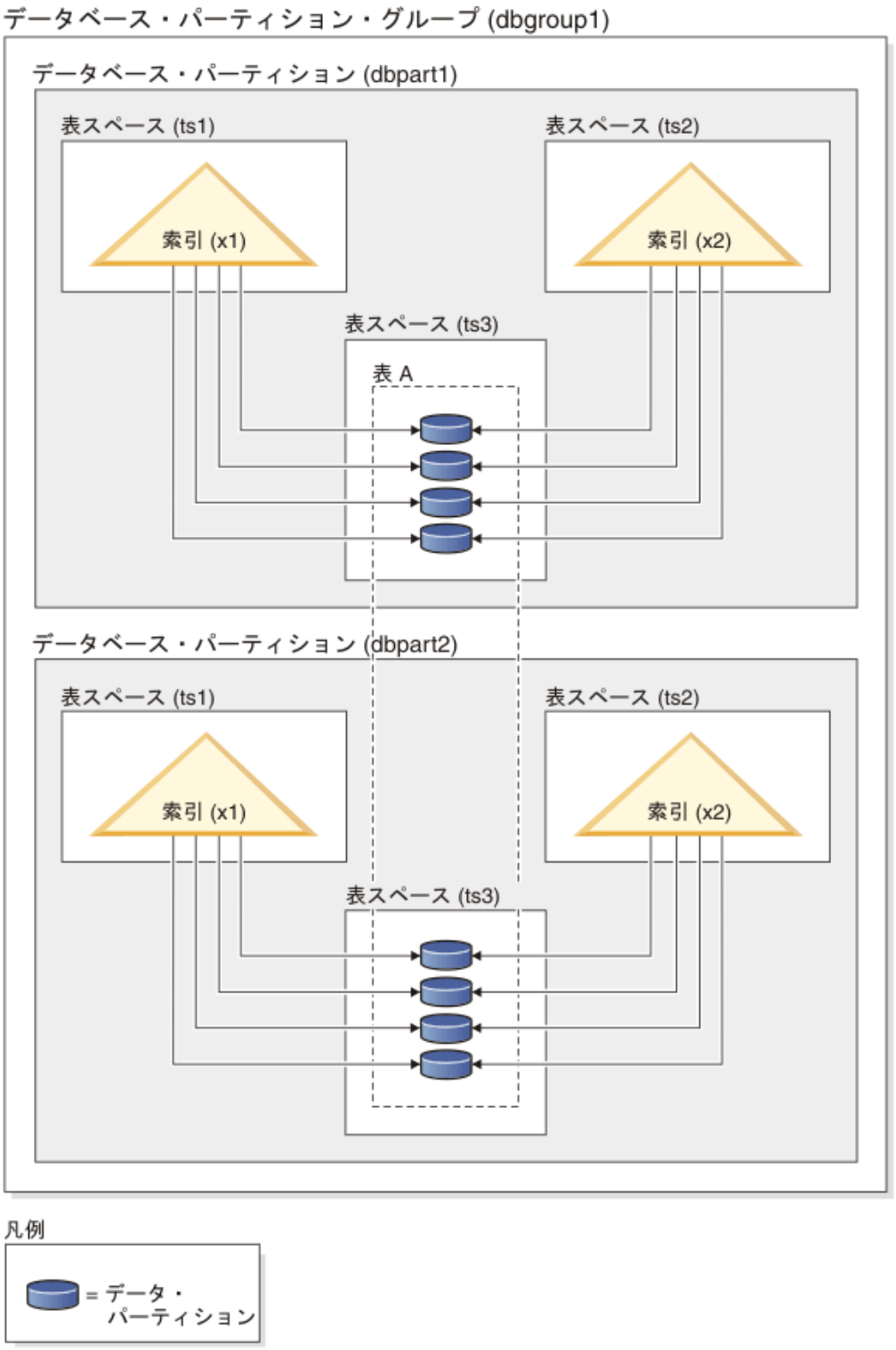


図 13. 分散とパーティション化の両方が行われている表における非パーティション化索引

88 ページの図 14 は、パーティション表上のパーティション化索引と非パーティション化索引の混合を示しています。

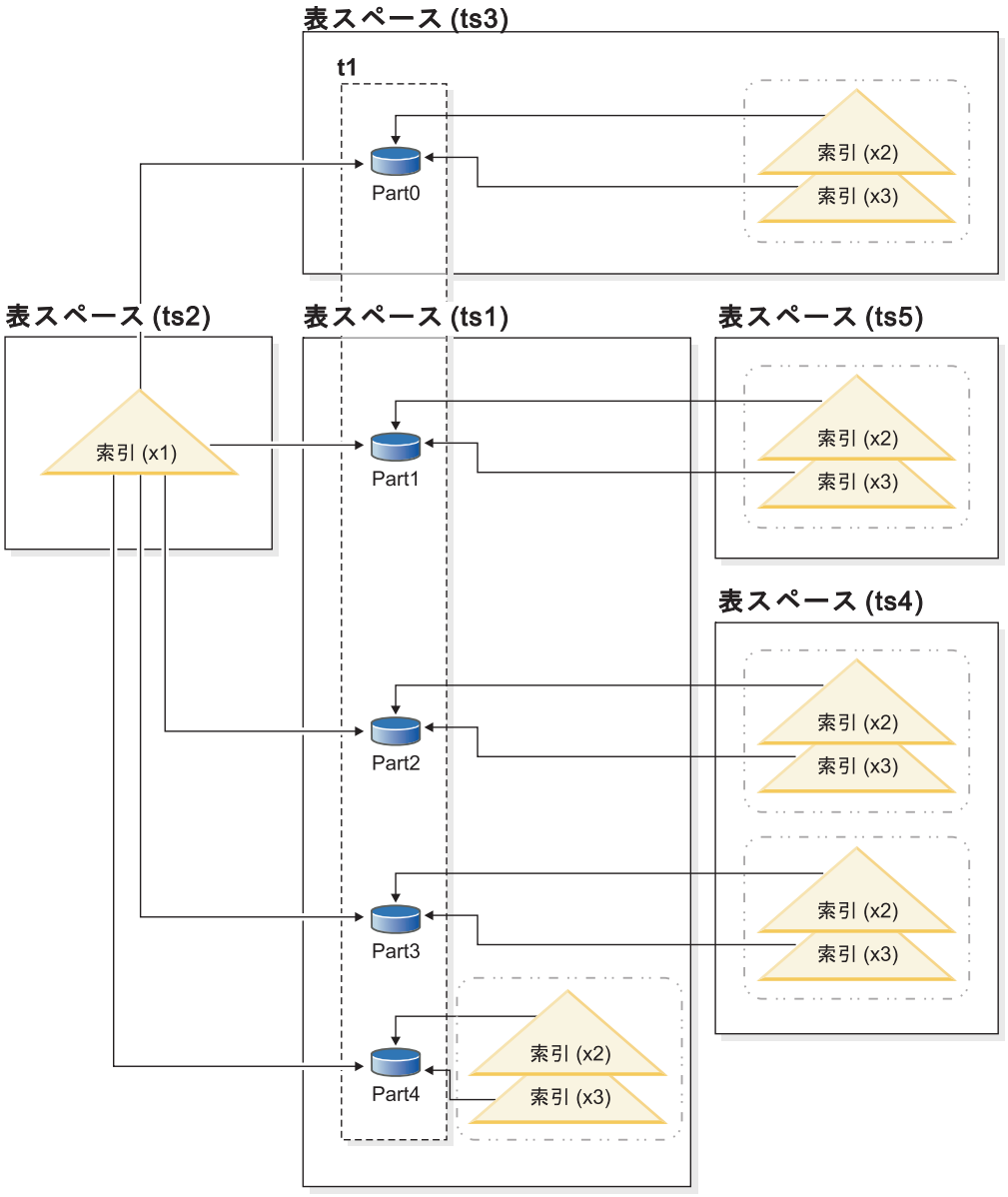


図 14. パーティション表上のパーティション化索引と非パーティション化索引

非パーティション化索引 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、パーティション化索引 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用する索引パーティションも示しています。これは、パーティション化索引のデフォルトです。

方法はそれぞれ異なりますが、非パーティション化索引およびパーティション化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。非パーティション化索引では、索引を作成するときに表スペースを指定できます。パーティション化索引の場合、表を作成するときに、どの表スペースに索引パーティションが保管されるかを決定する必要があります。

非パーティション索引

非パーティション化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、それにより代わりの索引の表スペース位置を指定することができます。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ると停止して、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 *tbspace* 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 *tbspace* 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化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。パーティション化索引を作成するときに INDEX IN 節を使用しようとすると、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 もパーティション化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: パーティション化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化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してもあります。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ー化しようとします。クラスター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。 91 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。

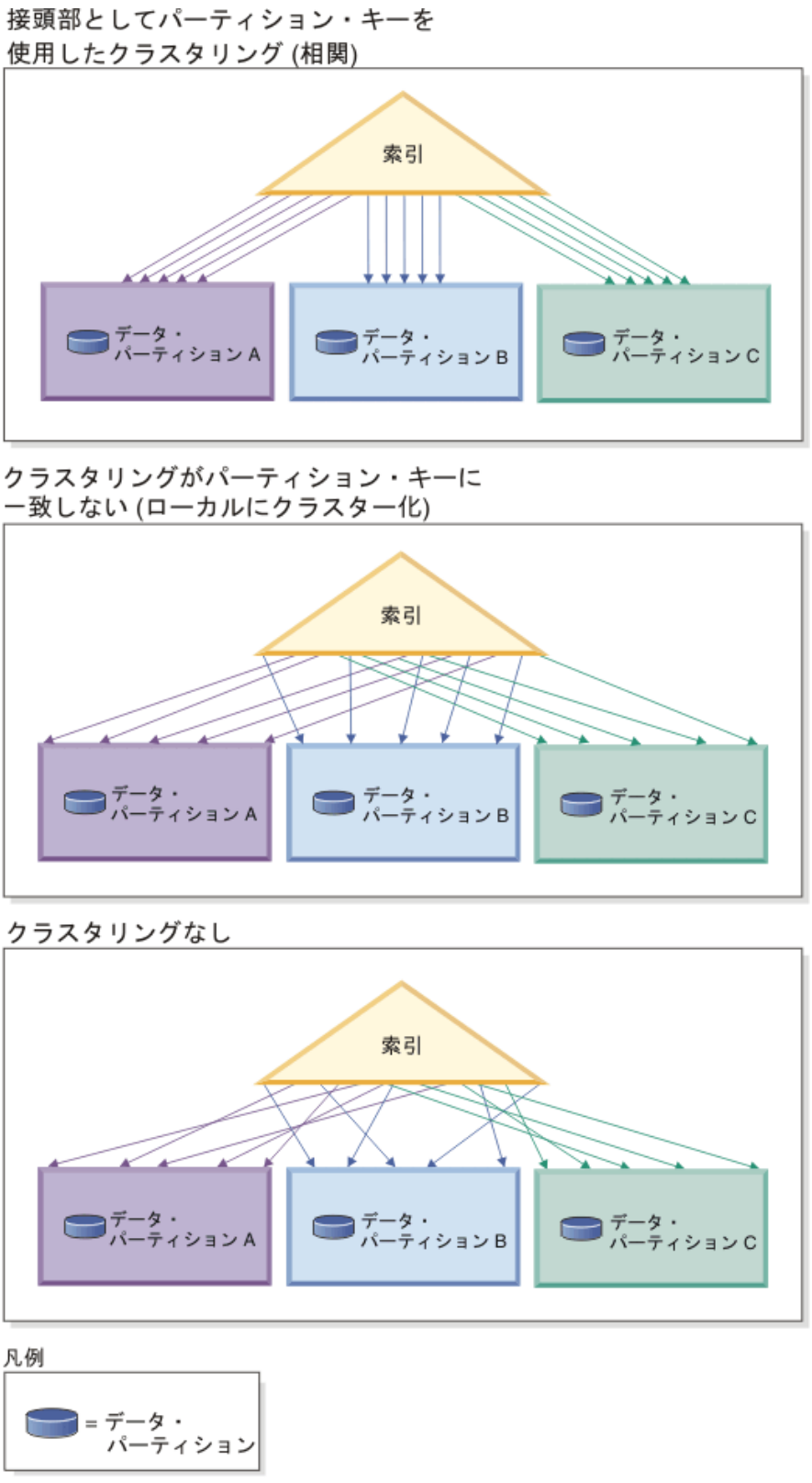


図 15. パーティション表でのクラスター化された索引の見込まれる効果。

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ー・プール内になければならないという意味です。対照的に、表がクラスター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。バッファー・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も遅くなります。

クラスタリング・キーが表パーティション・キーと相関していないものの、データがローカルにクラスター化される場合、バッファー・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、クラスター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 (91 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ー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に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・プロセッサー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・マネージャー共用メモリー・セットがあることを覚えておいてください。

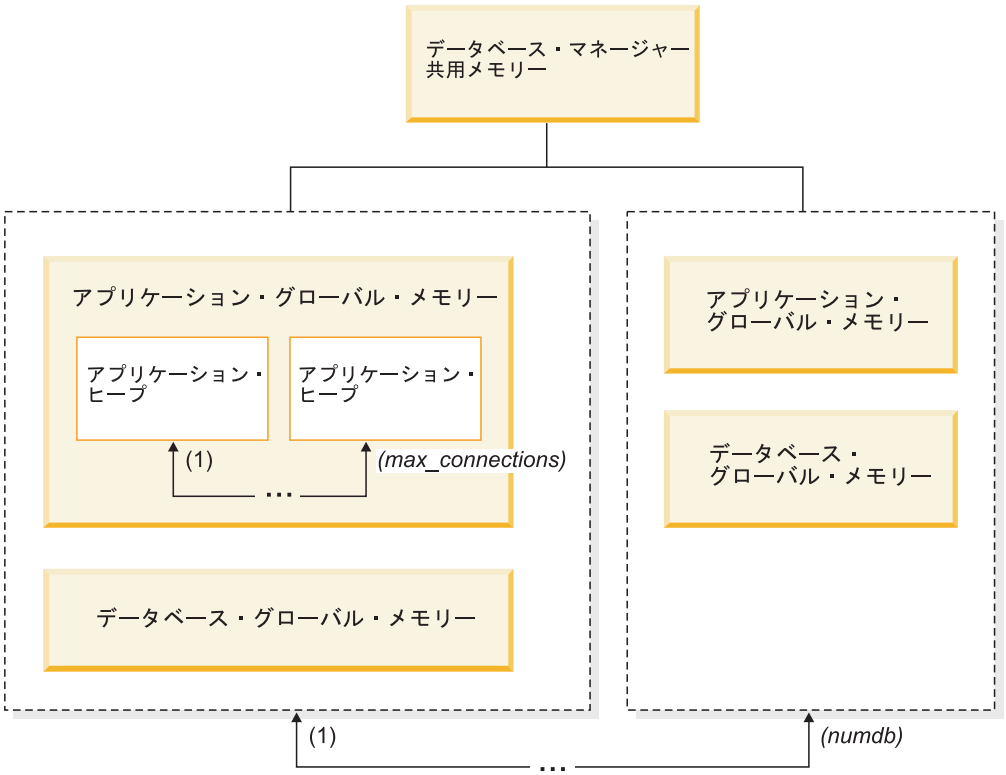


図 16. データベース・マネージャーによって割り振られるメモリーのタイプ

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*共用メモリー* (*インスタンス共用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・マネージャーの共用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とき

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。

- バッファー・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・バッファーを含む)
- 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
- 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。

共用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用メモリーの量は、**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とき

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す。

db2mtrk コマンドで起動されるメモリー・トラッカーによって、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_DBP_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。 GET SNAPSHOT コマンドを使用すると、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。

データベース・マネージャー共用メモリー

データベース・マネージャーの共用メモリーは、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。

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

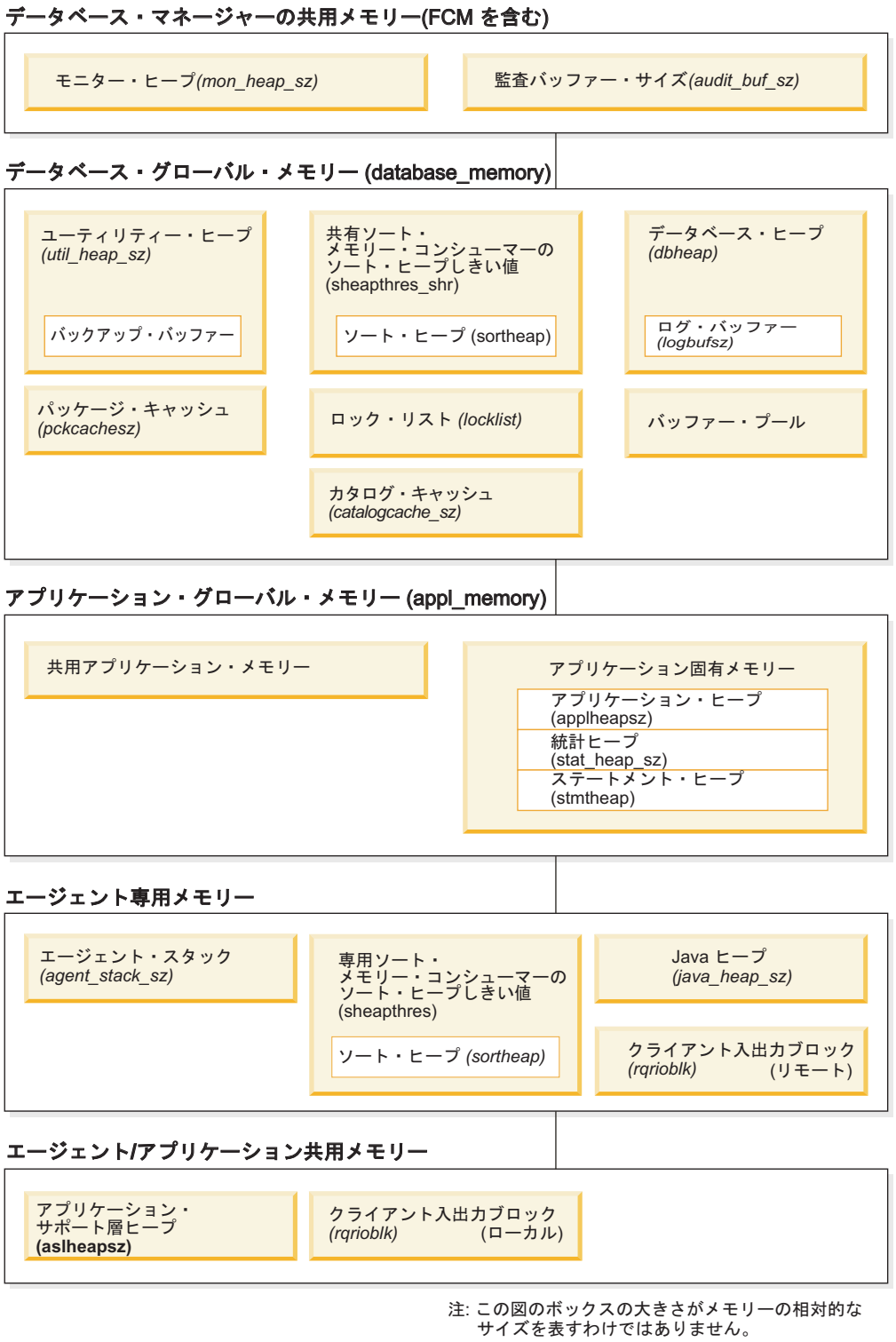


図 17. データベース・マネージャーでのメモリーの使用方法

モニター・ヒープ

このメモリー領域はデータベース・システムのモニター・データに使用されます。この領域のサイズは、データベース・マネージャーの **mon_heap_sz** 構成パラメーターによって決まります。

監査バッファー

このメモリー領域はデータベースの監査アクティビティーで使用されます。このバッファーのサイズは、データベース・マネージャーの **audit_buf_sz** 構成パラメーターによって決まります。

高速コミュニケーション・マネージャー (FCM) のバッファー・プール

パーティション・データベース・システムでは、 **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 バッファー・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、バッファー・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**
- **stmtheap**

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用メモリー

ローカル・クライアントの場合には、エージェント/アプリケーション共用メモリー・セグメントの合計数は、次の 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用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 バッファー・プールとメモリーの要件

パーティション・データベース・システムでは、高速コミュニケーション・マネージャー (FCM) バッファー共用メモリーはデータベース・マネージャーの共用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。

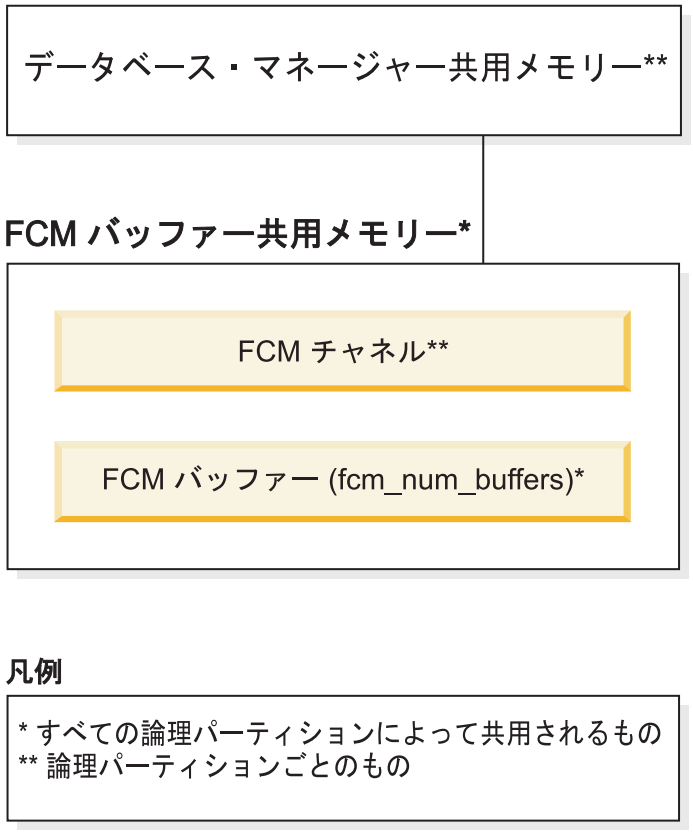


図 18. 複数の論理パーティションが使用されている場合の FCM バッファー・プール

各データベース・パーティションの FCM バッファーの数は、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 2 GB まで) のシステム・メモリーを FCM バッファーおよびチャ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**

オペレーティング・システムの中には、メモリーをスワップ・スペースにページアウトする必要があるときではなく、プロセスがメモリーを割り振る時点で常にスワ

ップ・スペースを割り振るものがあります。そのようなシステムの場合、システム上のメモリーの合計の少なくとも 2 倍の大きさのページング・スペースを確保することが通常は推奨されます。

有効なパラメーター範囲については、各パラメーターの詳細情報を参照してください。

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 - データベース共用メモリー・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 - 共用ソートのソート・ヒープのしきい値
- sortheap - ソート・ヒープ・サイズ

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用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、システムに十分の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用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、データベース共用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 **sortheap** 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。

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. バッファー・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、バッファー・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、そのバッファー・プールのいかなる項目も存在するべきではありません。

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。

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。

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。

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。

- コマンド行から構成パラメーターの設定を表示するには、SHOW DETAIL オプションを指定して 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 を使用することによってチューニングが使用可能かどうかを判別することもできます。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。

バッファー・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ー・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
  ```

  バッファー・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ー・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ー・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ー・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ー・プールのサイズを指定する ALTER BUFFERPOOL ステートメントは、そのバッファー・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)。バッファー・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ー・プール・サイズが AUTOMATIC に設定されていると、そのバッファー・プールはセルフチューニング操作に関与しません。

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。

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では、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ー・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 プロシージャー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 プロシージャー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、次回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ー・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ー・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ー・プールのサイズを指定する ALTER BUFFERPOOL ステートメントは、そのバッファー・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)。バッファー・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ー・プール・サイズが AUTOMATIC に設定されていると、そのバッファー・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、バッファー・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。メモリー要求量がデータベース・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、データベースが非活動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) を調べることもできます。

アクティビティーのバッファー・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・ページのバッファー・プール管理

バッファー・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ー・プールから明示的にパージされる (例えばオブジェクトのドロップ時のパージ) まで、バッファー・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

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ー・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連係しておらず、必要な場合にしか機能しないため、*非同期ページ・クリーナー* や*非同期バッファー書き込み機能* と呼ばれることもあります。

更新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ー・プールの管理作業の分担方法を示しています。

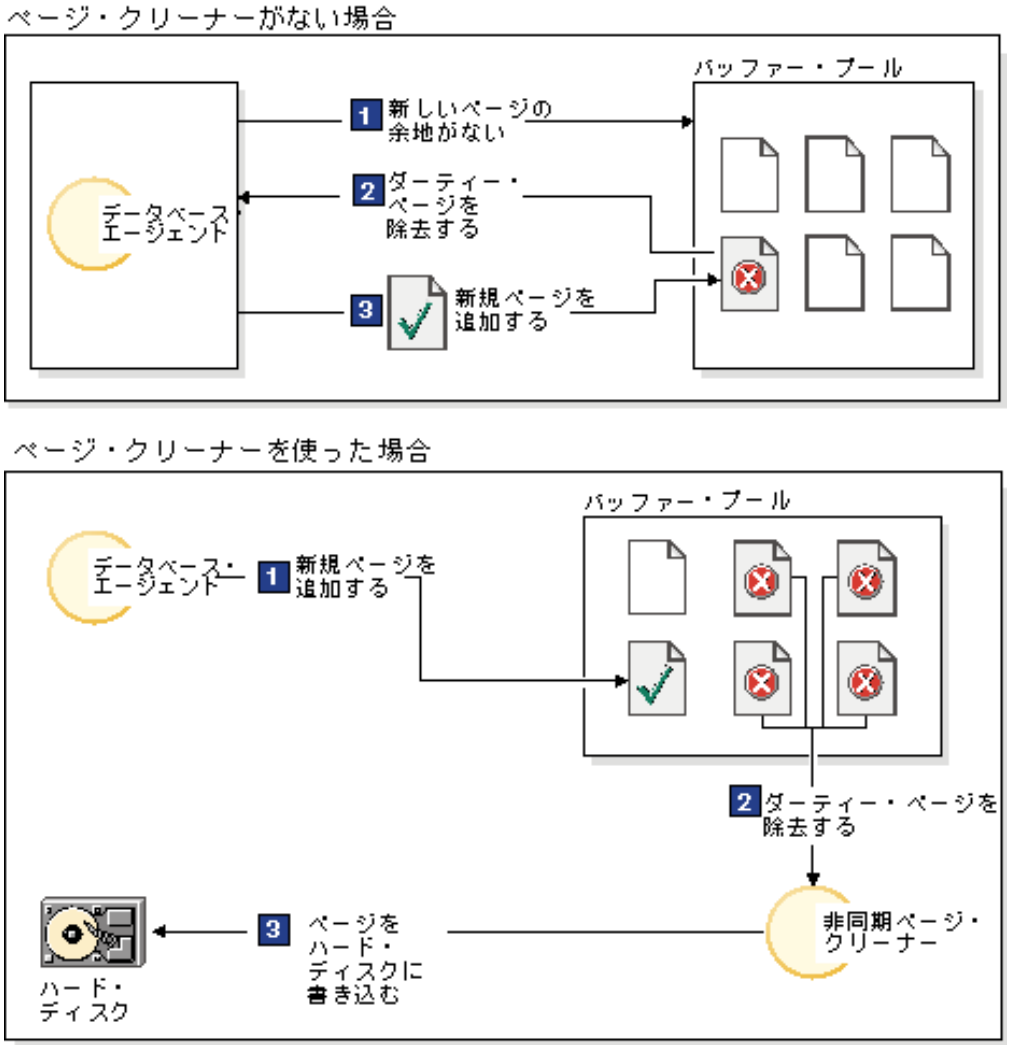


図 19. 非同期ページ・クリーニング： ダーティー・ページがディスクに書き出されます。

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ー・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
- バッファー・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、ダーティー・ページをディスクに書き込みます。

```
logfilsiz * 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ー・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・バッファー・プールの管理

各データベースは最低 1 つのバッファー・プールを必要としますが、複数のページ・サイズの表スペースを持つ 1 つのデータベースに対して、サイズやページ・サイズの異なる複数のバッファー・プールを作成することもできます。

ALTER BUFFERPOOL ステートメントを使ってバッファー・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・バッファー・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ー・プールに割り当てていない場合、表スペースはデフォルトのバッファー・プールに割り当てられます。デフォルトのバッファー・プールのサイズ変更と、その属性の変更は可能ですが、それをドロップすることはできません。

バッファー・プールのページ・サイズ

データベースを作成またはアップグレードした後で、追加のバッファー・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・バッファー・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、バッファー・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ー・プールに表スペースを割り当てることはできません。

注: 4 KB より大きいページ・サイズ (8 KB、16 KB、32 KB など) で表スペースを作成する場合、同じページ・サイズを使用するバッファー・プールに割り当てる必要があります。このバッファー・プールが現在アクティブでない場合、DB2 は、同じページ・サイズを使用する別のアクティブ・バッファー・プール (もしあれば)、または最初のクライアントがデータベースに接続した際に DB2 が作成するデフォルトのシステム・バッファー・プールに一時的に表スペースを割り当てようとします。データベースが再度アクティブにされ、最初に指定されたバッファー・プールがアクティブである場合、 DB2 は表スペースをそのバッファー・プールに割り当てます。

CREATE BUFFERPOOL ステートメントでバッファー・プ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・プール・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・プール・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・プールのデフォルト定義は、全データベース・パーティションに対して同一になっています。ただし、それ以外の定義が CREATE BUFFERPOOL ステートメントで指定された場合、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・プールのサイズが ALTER BUFFERPOOL ステートメントによって変更された場合は除きます。

大容量バッファー・プールの利点

大容量バッファー・プールには次の利点があります。

- 頻繁に要求されるデータ・ページをバッファー・プール内に保持しておけるので、迅速にアクセスができるようになります。入出力操作を減らすと入出力の競合も減るので、結果として、応答時間が短縮したり、入出力操作に必要なプロセッサー・リソースを減らすことになります。
- 同じ応答時間で、より高いトランザクション率を達成する可能性があります。
- 頻繁に使用するディスク・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・プールが多数ある場合の利点

使用しているシステムに以下の条件のいずれかが当てはまる場合には、バッファー・プールを 1 つだけ使用してください。

- 合計バッファー・プール・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・プ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・プ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・プールに割り当てることを検討してください。このバッファー・プールのサイズが適切ならば、ページが見つかる可能性が高くなるので、応答時間を短縮したりトランザクション・コストを下げるのに役立ちます。
- データを別のバッファー・プールに分離し、特定のアプリケーション、データ、索引を特別扱いにすることができます。例えば、頻繁に更新される表や索引を入れるバッファー・プールを、頻繁に照会されるけれども更新は多くない表や索引のバッファー・プールとは別々に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・プールを使用することができます。特に、非常に大きな表に対して非常にランダムなアクセスを要求するアプリケーションの場合は有効です。このような場合には、1 回の照会分より長い時間、バッファー・プール内にデータを保持しておく必要はありません。この種のデータには小さなバッファー・プールを用意して、余ったメモリーを他のバッファー・プ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・プ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・プ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・プール・メモリーの割り振り

バッファー・プールを作成するか、バッファー・プールを変更する場合は、データベースの開始時にすべてのバッファー・プールの割り振りができるように、すべてのバッファー・プールで必要なメモリーの合計量がデータベース・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・プールを作成または変更する場合、データベース・グローバル・メモリーの追加メモリーが使用できるようになっている必要があります。新規バッファー・プールを作成する際や既存のバッファー・プ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・プール (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が、データベースへの接続、バッファー・プール・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・バッファー・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、使用可能メモリー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、**chngpgs_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許容値よりも低いと、ページ・クリーナーがバッファー・プール全体を検索し、スワップアウト可能なページを書き出し、それらのページの位置をエージェントに通知します。
- ページ・クリーナーはロガーの発行するログ・シーケンス番号 (LSN) ギャップ・トリガーに反応しない。

  バッファー・プール内の最も古いページを更新したログ・レコードから現行のログ位置までのログ・スペースの量が、**softmax** 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ガーがページ・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ー状態とになります。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に、DB2 のログ・スペースが不足してしまうこともあり得ます。

  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・ページの数が極端に低くなっている場合。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。
  - 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ーブしておくということです。

索引ページおよびデータ・ページをバッファー・プールにプリフェッチすると、入出力の待ち時間が減るので、パフォーマンスは向上します。加えて、並列入出力によってもプリフェッチの効率は拡張されます。

プリフェッチには、2 つのカテゴリーがあります。

- *順次プリフェッチ* 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。
- *リスト・プリフェッチ* (*リスト順次プリフェッチ* とも呼ばれる) 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。

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ー・プールに読み込むと、アプリケーションのオーバーヘッドは大幅に短縮されます。

プリフェッチが開始されるのは、データベース・マネージャーが、順次入出力が適切であり、プリフェッチを行うとパフォーマンスが向上すると判断したときです。表スキャンや表ソートなどの場合、データベース・マネージャーは、自動的に順次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
```

順次検出

順次プリフェッチがパフォーマンスの向上につながるかどうかが、瞬時には判断しにくい場合があります。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。 **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・マネージャーが順次検出を実行する必要があるかどうか制御できます。

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ー化されていると、データページの読み取りはほぼ順次になるので、プリフェッチによってパフォーマンスが向上する可能性があります。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・コンテナーの数、各コンテナーの物理ディスクの数 (RAID 装置を使用している場合)、および表スペースの EXTENTSIZE の値 (データベース・マネージャーが別のコンテナーを使用する前にコンテナーに書き込むページ数) 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ーを 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。各コンテナーに 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、バッファー・プールの使用をモニターしているので、プリフェッチしたために、別の作業単位で必要なページが、バッファー・プールから除去されることはありません。データベース・マネージャーは、問題を避けるために、プリフェッチするページ数を、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** (バッファー・プール非同期データ読み取り) データ・エレメントを調べることにより)

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ーにある場合には、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、コンテナーが別個の物理装置を使用するように構成します。

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。エクステントは、コンテナー間で表スペースがストライピングされるときの細分度です。ブロック・サイズが等しく定義された 1 つのバッファー・プールに、エクステント・サイズの異なる複数の表スペースをバインドできます。このため、エクステント・サイズとブロック・サイズを活用すれば、バッファー・プール・メモリーを効率的に使用することができます。バッファー・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、バッファー・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ー・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ー・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ー・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ー・プールにバインドしてください。エクステント・サイズがブロック・サイズより大きい場合にも良好なパフォーマンスが得られますが、ブロック・サイズよりも小さい場合にはパフォーマンスが低下します。

ブロック・ベースのバッファー・プールを作成するには、CREATE BUFFERPOOL または ALTER BUFFERPOOL ステートメントを使用します。

注: ブロック・ベースのバッファー・プールの用途は、順次プリフェッチです。アプリケーションが順次プリフェッチを使用しない場合、バッファー・プールのブロック・エリアは無駄になります。

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す。

プリフェッチと並列処理のための入出力サーバー構成:

プリフェッチを使用可能にするため、データベース・マネージャーは、*入出力サーバー* と呼ばれる別の制御スレッドを開始してデータ・ページを読み取ります。

その結果、照会処理は 2 つの並列のアクティビティー、つまりデータ処理 (CPU) とデータ・ページ入出力に分けられます。入出力サーバーは、CPU のアクティビティーからプリフェッチ要求が出るまで待機します。このプリフェッチ要求には、照会を満たすために必要な入出力記述が含まれます。

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。

122 ページの図 20 にこのプロセスが示されています。

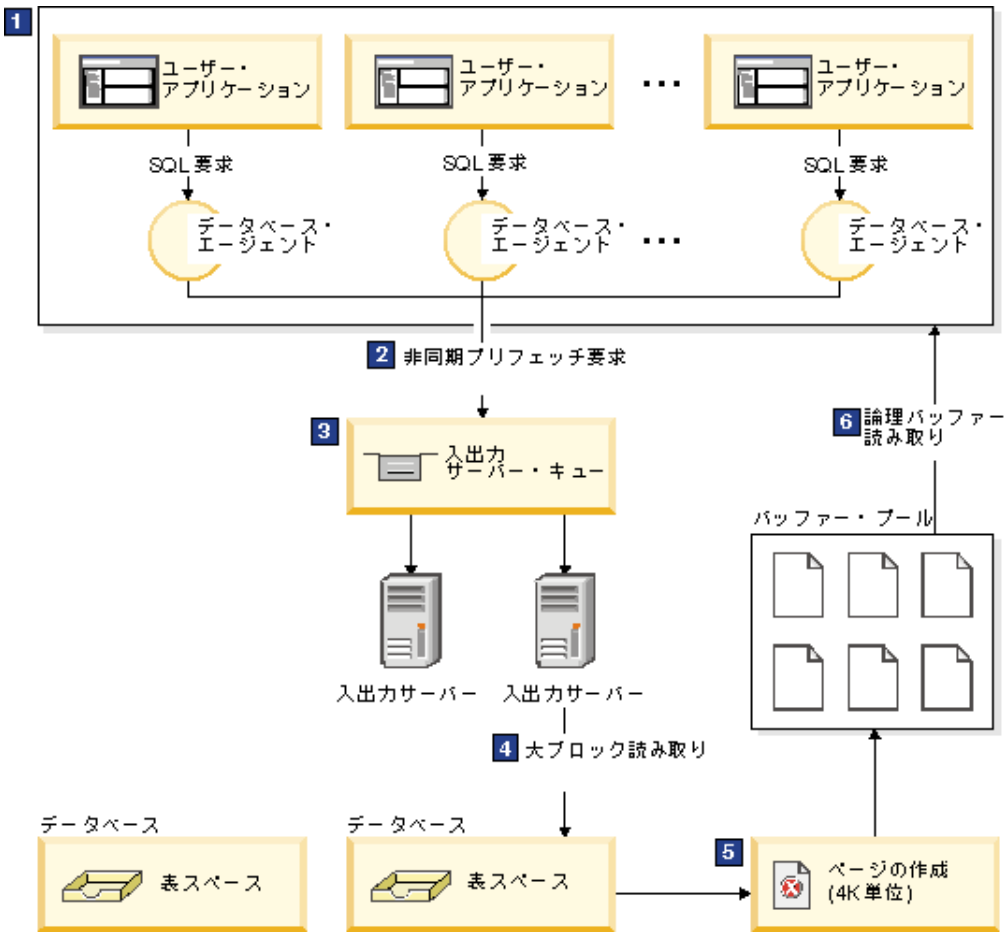


図 20. 入出力サーバーを使用したデータのプリフェッチ

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ー・プールに読み込みます。表スペースから同時にデータを取り出せる入出力サーバーの数は、キューに入っているプリフェッチ要求の数と **num_ioservers** データベース構成パラメーター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、バッファー・プール内のデータ・ページに対して必須の走査実行し、その結果をユーザー・アプリケーションに戻します。

並列入出力の管理:

1 つの表スペースに対して複数のコンテナーが存在する場合、データベース・マネージャーは*並列入出力* を開始することができます。データベース・マネージャーはこの並列入出力において複数の入出力サーバーを使用して、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ーごとに入出力ワークロードを処理するため、複数のコンテナーを並列で読み取ることができます。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ー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の数は、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。
- *リスト・プリフェッチ* の場合、ページの各リストは、データ・ページが保管されるコンテナーにしたがって、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・バッファー・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ーの数) 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。データはバッファー・プールにリストアされません。リストア・バッファー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・コンテナー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ーの数より少し多めに指定します。
- エクステント・サイズおよびプリフェッチ・サイズが表スペースに適していること。バッファー・プールの使い過ぎを避けるために、プリフェッチ・サイズを大きくし過ぎないでください。理想的なサイズは、エクステント・サイズ、各コンテナーの物理ディスクの数 (RAID 装置を使用している場合)、および表スペース・コンテナーの数を掛け合わせた値です。エクステント・サイズはかなり小さくする必要があり、適した値は 8 から 32 ページの範囲です。
- コンテナーが、別々の物理ドライブに存在すること。
- 並列処理の多重度に一貫性を持たせるために、すべてのコンテナーが同じサイズであること。

他のコンテナーより小さいコンテナーが 1 つ以上ある場合には、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ーが満杯になると、それから先のデータは残りの、他のコンテナーに保管されることになるので、コンテナー間の均衡がとれなくなります。コンテナーが不均衡になると、データのプリフェッチができるコンテナーの数が、コンテナーの合計数より少なくなるので、並列プリフェッチのパフォーマンスが低下します。
- 他より小さいコンテナーが後で追加されたときにデータの均衡が再度図られた場合には、小さなコンテナーには他のコンテナーより少ないデータが入れられることになります。他のコンテナーに比べて少量のデータでは、並列プリフェッチは最適化されません。
- ある 1 つのコンテナーが他より大きいときに、他のコンテナーがすべて満杯になったとすると、その大きいコンテナーが追加のデータを保管するための唯一のコンテナーになります。したがって、データベース・マネージャーは、この追加データにアクセスする際には、並列プリフェッチを使用することができません。

- 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、複数のプロセッサー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。各プロセッサーを十分に活用するには、十分な入出力容量が必要です。通常、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、およびプロセッサーの速度によって異なります。

AIX 上での IOCP の構成:

AIX 5.3 TL9 SP2 および AIX 6.1 TL2 には、基本インストールの一部として組み込まれた、入出力完了ポート (IOCP) のファイル・セットがあります。ただし、最小のオペレーティング・システム要件を適用するために、新規オペレーティング・システムのインストールを使用するのではなく、オペレーティング・システムのアップグレードを使用した場合、入出力完了ポート (IOCP) を個別に構成する必要があります。

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。

   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活動化動作

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活動化は、*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活動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、活動化または非活動化されます。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、データベースは非活動化されます。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) の活動化および非活動化ごとに生じます。

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活動化するには、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活動化します。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**)、および共用ソートのソート・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、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。

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。

表または索引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、論理上の順次データが順序どおりになっていないデータ・ページ上に置かれる場合があるので、データベース・マネージャーはデータにアクセスするために追加の読み取り操作を実行する必要があります。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ーが発生した。
- 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、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。

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・プロシージャー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 プロシージャー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ー化されている。
- 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ー化するために、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 1. 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	大きくなることがある
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
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 2. 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
範囲クラスター表 (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

表 2. 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 (続き)

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
注: 1. クラスター化は 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。 2. RCT の範囲域は常にクラスター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。

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。

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。

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、以下のようにします。

   ```
   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、以下のようにします。

   ```
   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 プロシージャーを使用して 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。

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) 表スペースの間で相違点があります。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・ユーティリティーでは、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の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ー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、バッファー・プール内のダーティー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。

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ト・サイズで、最小は 32 です。

2. 範囲を空ける

   REORG ユーティリティーは、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ー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 ユーティリティーは 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。 NOTRUNCATE 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。

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。

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。

1. 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、単に表の名前および INPLACE オプションを指定します。例えば、

   ```
   reorg table employee inplace
   ```

2. 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 プロシージャーを使用して REORG TABLE コマンドを指定します。例えば、

   ```
   call sysproc.admin_cmd ('reorg table employee inplace')
   ```

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オンラインで表を再編成するには、db2Reorg API を呼び出します。

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・フルやロギング・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 オプションを指定すると、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 オプション、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。

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。

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 オプション、および PAUSE オプションを指定します。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 オプションを指定します。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 オプションが現在の 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 オプションは無視されます。

   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 ユーティリティーから参照できず、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・アクティビティー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。

- 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。

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ーの呼び出し元 (アプリケーションまたはコマンド行プロセッサー) 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しなくなり、*不良クラスター索引* になる。リーフ・ページが不良な状態にクラスター化されていると、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。
- 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
- 索引の現行サイズと等しい量の、索引保管先表スペース内のフリー・スペース。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 ユーティリティーは、そのアクティビティー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。
- 表とその索引の両方をオフラインで再編成できるようにするオプションを指定して、REORG TABLE コマンドを使用します。
- REORG INDEXES コマンドを使用してオンラインで索引を再編成します。この方法は、実稼働環境で選択することもできます。なぜなら、この方法では、索引の再ビルド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。

オンライン索引再編成

ALLOW WRITE ACCESS オプションを指定して REORG INDEXES コマンドを使用すると、表への読み取り/書き込みアクセスが継続しながら、指定された表の索引すべてが再ビルドされます。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、またはマルチディメンション・クラスタリング (MDC) 表では、ALLOW WRITE ACCESS モード (CLEANUP ONLY 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はサポートされていません。

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。

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。

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必要としている可能性があります。

- 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 x 253) = 28905 あることになります。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してください。

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。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、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を影響を与える可能性があります。データベース・マネージャーはポインターに従って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す。この値が小さいということは、プリフェッチャーが本来の効果性を発揮できないということです。これは、表スペースの PREFETCHSIZE 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) に近づくはずです。

  – AVERAGE_RANDOM_FETCH_PAGES 列は、索引を使用して表の行をフェッチするときの、順次ページ・アクセスの間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、適切でない順序で挿入が行われることにより発生します。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 列に保管されます。

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 ONLY 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、クラスター率が高い方が良い可能性があります。クラスター率が低いと、この種のスキャンでは入出力が増えます。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、クラスター率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。

  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。

- リーフ・ページの数

  索引が使用するリーフ・ページの数を調べるには、SYSCAT.INDEXES 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ーによりページ分割が行われ、索引のサイズが大きくなることがあります。表の 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計算するには、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ー・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 エクステント・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 <= (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) * (NPARTITIONS * 1 エクステント・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

表または索引の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のは避けられないため、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れらを考慮する必要があります。

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。

- ユーティリティーを実行する際の処理時間
- REORG ユーティリティー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 ユーティリティー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ー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 ユーティリティーは頻繁に内部 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ーが 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ー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、以下のようにします。

- 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 ユーティリティーは MDC 表の再編成を依然として勧める場合があります。
- 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、表のページがいっぱいになると、表の最後にレコードが追加され、クラスター特性は徐々に失われます。

クラスタリング索引を作成した後に表の 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 ユーティリティーによって収集された CLUSTERRATIO または CLUSTERFACTOR 統計は改善されます。

索引の再編成の必要性を少なくする

索引の再編成の必要性を少なくするには、以下のようにします。

- PCTFREE または LEVEL2 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。
- 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 ONLY 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。

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 ユーティリティー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 ユーティリティー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。

自動再編成フィーチャー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。ただし再編成操作は、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 表からのエクステントの解放は、DMS 表スペースの MDC 表でのみサポートされています。MDC 表からのエクステントの解放を、データベースの自動保守アクティビティーの一部にすることができます。

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) が表で定義されており、再編成が必要なパーティションが 1 つだけの場合、再編成が必要なパーテ

ィションを指定するために ON DATA PARTITION 節を使用して、自動再編成によって REORG TABLE 操作が実行されます。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) が表で定義されていない場合、再編成が必要な各データ・パーティションで ON DATA PARTITION 節を使用して、自動再編成によって REORG TABLE 操作が実行されます。

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にすみます。

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。そうなると、データベース・マネージャーは、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。

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**

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に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しか影響しません。

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はあります。

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。

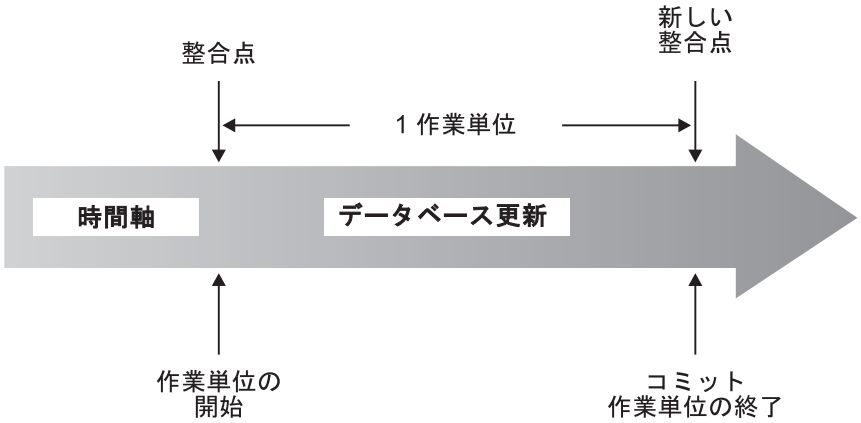


図 21. COMMIT ステートメントの作業単位

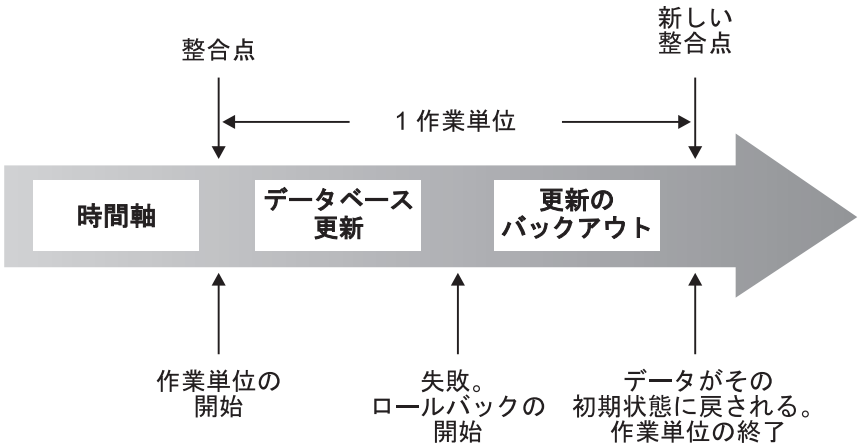


図 22. ROLLBACK ステートメントの作業単位

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用することです。データベース・マネージャーはこのアクセスを制御し、次のような望ましくない結果を防ぎます。

- **更新の消失**。 2 つのアプリケーション A および B が、両方とも同じ行を読み取り、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になります。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ん。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ー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施行され、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、ロックされることはありません。

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。

共用 (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。

- 154 ページの『反復可能読み取り (RR)』
- 155 ページの『読み取り固定 (RS)』

- 155 ページの『カーソル固定 (CS)』
- 155 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、10 000 個の行をスキャンしてそれらに述部を適用する場合、たとえ 10 行しか適格でなくても、それら 10 000 個の行すべてにロックがかけられます。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、外部キー列のみが含まれる索引を作成し、参照整合性スキャンでそれを代わりに使用できるようにします。

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 により、ある UOW で適格とされて読み取られた行は、その UOW が完了する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くなり、また別のアプリケーション・プロセスによって変更されたすべての行は、そのプロセスによって変更がコミットされるまで読み取れなくなります。 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。

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 つの作業単位の間に同じ照会を 2 回以上発行しない、または 1 つの作業単位の間に同じ照会を 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。

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 3 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 3. 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

表 3. 分離レベルの比較 (続き)

	UR	CS	RS	RR
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

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、または削除の操作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. *ダーティー読み取り現象* 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になります。
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 currently committed 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 4 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 4. 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

表 4. 分離レベルのまとめ (続き)

分離レベル	コミットしていないデータへのアクセス	反復不能読み取り	幻像読み取り
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 5 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 5. 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決めるものなので、並行性の要件とデータ保全性の要件のバランスを取る分離レベルを選択する必要があります。

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。

- **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
  – Searched (検索条件付き) DELETE
  – INSERT
  – SELECT
  – SELECT INTO
  – Searched (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。sqlaprep 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・インターフェース (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ー (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。

currently committed セマンティクスによる並行性の改善

ロック・タイムアウトとデッドロックが生じる可能性があるのは、行レベルのロックがある CS 分離レベルです。特に、そ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得ます。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の際に発行されるロックの待機を許容できませんし、アプリケーションによっては、非コミット・データの処理を許容できないものの読み取りトランザクション用に非ブロッキング動作を依然として必要とするものがあります。

新しい *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、読み取り側は書き込み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。

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。これにより、すべてのアプリケーションが新しい動作を活用でき、アプリケーション自体に変更を加える必要はありません。新しいデータベース構成パラメーター **cur_commit** を使用すると、この動作をオーバーライドできます。こ

れが役立つ場合があります。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です。

同様に、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で、デフォルトで **cur_commit** が使用不可にされているアップグレード・データベースでは、必要に応じてこのパラメーターを後ほどオンにできます。

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、または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 はスキャン・レベルによって決定されるので、書き込み側のアクセス・プランには currently committed スキャンが組み込まれている場合があることに注意してください。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** が使用不可の場合には当てはまりません。

例

currently committed セマンティクスでデッドロックを回避する、以下のシナリオについて考慮します。このシナリオでは、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。そのとき各アプリケーションは、もう一方のアプリケーションが更新した表を読み取ろうとします (読み取り専用カーソルを使用)。

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、(いずれかのアプリケーションの) ステップ 2 における照会で、他のアプリケーションが現在更新しているデータを必要とする場合、そのアプリケーションはロックの解放を待機せず、デッドロックは生じま

せん。代わりに、データのこれまでのコミット済みバージョンが探索されて使用されます。

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有効な場合には、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 6. RID 索引単独アクセス

述部	非コミットの評価
なし	いいえ
検索引数述部	はい

表 7. 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
検索引数述部	はい

表 8. 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	検索引数述部	いいえ	いいえ
検索引数述部	なし	はい	いいえ
検索引数述部	検索引数述部	はい	いいえ

表 9. 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	検索引数述部	いいえ	はい
検索引数述部	なし	はい	いいえ
検索引数述部	検索引数述部	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す。

表 10. 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す。セッション 2 の照会では、セッション 1 の更新のロールバックがセッション 2 の照会を満たしているにもかかわらず、結果セットの最初の行が省略されます。

操作の順序を逆にすると、非コミットの評価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。デフォルトのロッキング動作では、セッション 2 で最初に行ロックが獲得され、セッション 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 の検索済み UPDATE が実行されないようにします。セッション 1 の検索済み UPDATE が最初に行を検査して、条件を満たしている場合にのみ行をロックしようとする場合は、セッション 1 の照会は非ブロッキングになります。

制約事項

- **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。
- 検索引数述部評価述部が存在します。
- 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) 表の場合は索引スキャンのためにブロック・レベル・ロッキングを据え置くことができますが、表スキャンの場合はブロック・レベル・ロッキング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、行レベルのロッキングはデータ・アクセスのために据え置かれるのではなく、表の中の行に移動する前に索引アクセス中に行がロックされます。

- 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、以下のようにします。

- 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い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:

式で結合述部を使用すると、結合方式がネスト・ループに制限されます。

加えて、カーディナリティー推定値が不正確になるおそれがあります。以下は、式を伴う結合の例です。

```
WHERE SALES.PRICE * SALES.DISCOUNT = TRANS.FINAL_PRICE
WHERE UPPER(CUST.LASTNAME) = TRANS.NAME
```

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、照会オプティマイザーは、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
```

照会オプティマイザーは、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)
```

DB2 Database for Linux, UNIX, and Windows バージョン 9.5 フィックスパック 1 における大/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たものです。 UCA500R1 照合名で _Sx 属性を使用して、照合の強度を制御できます。例えば、UCA500R1_LFR_S1 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。

結合列でのデータ・タイプの不一致の回避:

場合によっては、データ・タイプの不一致によってハッシュ結合の使用が妨げられます。

ハッシュ結合の場合、他の結合方法にはない、結合述部に関する追加の制約事項がいくつかあります。とりわけ、結合列のデータ・タイプは完全に一致していなければなりません。例えば、一方の結合列が FLOAT で、他方が REAL の場合、ハッシュ結合はサポートされません。加えて、結合列のデータ・タイプが CHAR、GRAPHIC、DECIMAL、または DECFLOAT である場合は、長さが同じでなければなりません。

述部でノーオペレーション式を使用してオプ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、力ず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えます。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、(F.PROD_KEY, F.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。)

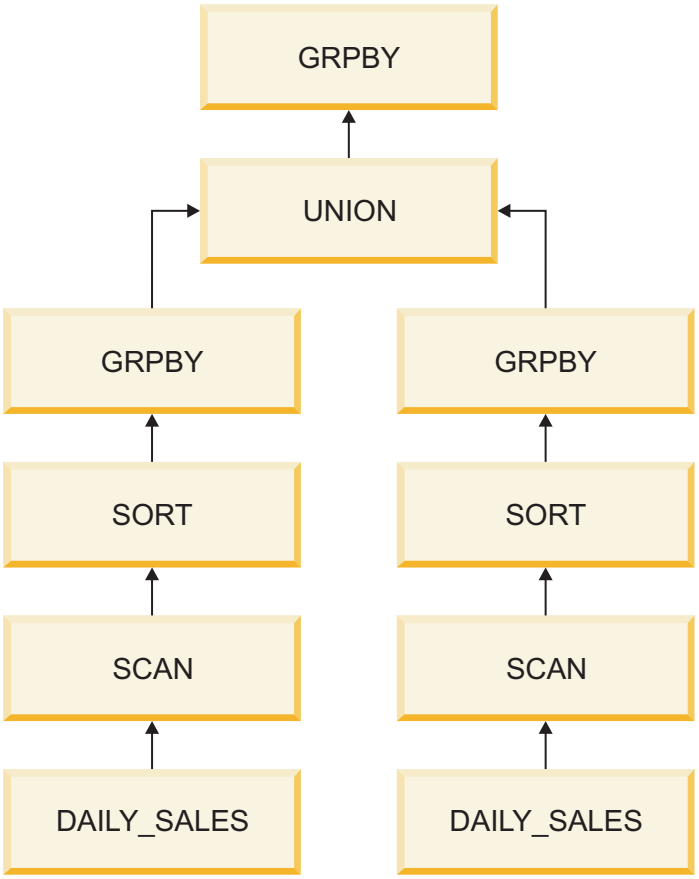
DISTINCT キーワードによる複数の集約の回避:

同じ副選択内で、複数の DISTINCT 集約を実行する照会を使用しないでください。実行コストが高いためです。

次の例を考慮してください。

```
SELECT SUM(DISTINCT REBATE), AVG(DISTINCT DISCOUNT)
  FROM DAILY_SALES
  GROUP BY PROD_KEY
```

別個の REBATE 値および別個の DISCOUNT 値の集合を特定するには、PROD_KEY 表からの入力ストリームを 2 回ソートしなければならない可能性があります。この照会の照会アクセス・プランは以下のようになるでしょう。



オプティマイザーは、元の照会を別個の集約に分けるように書き直して、それぞれに DISTINCT キーワードを指定してから、それら複数の集約を UNION キーワードを使用して結合します。内部で書き直されたステートメントは以下のとおりです。

```
SELECT Q8.MAXC0, (Q8.MAXC1 / Q8.MAXC2)
FROM
  (SELECT MAX(Q7.C0) AS MAXC0, MAX(Q7.C1) AS MAXC1, MAX(Q7.C2) AS MAXC2
    FROM
      (SELECT SUM(DISTINCT Q2.REBATE) AS C0, CAST(NULL AS INTEGER) AS C1,
        0 AS C2, Q2.PROD_KEY
      FROM
        (SELECT Q1.PROD_KEY, Q1.REBATE
          FROM DB2USER.DAILY_SALES AS Q1) AS Q2
      GROUP BY Q2.PROD_KEY
      UNION ALL
      SELECT CAST(NULL AS INTEGER) AS C0, SUM(DISTINCT Q5.DISCOUNT) AS C1,
          COUNT(DISTINCT Q5.DISCOUNT) AS C2, Q5.PROD_KEY
        FROM
          (SELECT Q4.PROD_KEY, Q4.DISCOUNT
            FROM DB2USER.DAILY_SALES AS Q4) AS Q5
        GROUP BY Q5.PROD_KEY) AS Q7
      GROUP BY Q7.PROD_KEY) AS Q8
```

複数の DISTINCT 集約を避けることができないなら、**DB2_EXTENDED_OPTIMIZATION** レジストリー変数を ENHANCED_MULTIPLE_DISTINCT オプションと共に使用することを考慮してください。このオプションを使用する結果、複数の DISTINCT 集約への入力ストリームが 1 回読み込まれ、次いで UNION の各アームのために再使用されます。データベース・パーティション数に対するプロセッサーの比率が低い場合 (例えば、1 以下の比率)、このオプションによって、こうしたタイプの照会のパフォーマンスが改善される可能性があります。この設定は、対称型マルチプロセッサー (SMP) を持たないパーティション・データベース環境で使用してください。この最適化拡張を使用しても、必ずしもすべての環境で照会のパフォーマンスが向上するわけではありません。それぞれの照会パフォーマンスが向上するかどうかを判別するために、テストを行う必要があります。

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必要で、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、照会オプティマイザーによって、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、オプティ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:

オプティマイザーは、スター型結合またはハブ型結合という、スター・スキーマ用の 2 種類の特殊結合方式を検討します。これは、パフォーマンスの大幅な向上に寄与します。

しかし、照会が以下の基準に適合していなければなりません。

- 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
  - 索引 ANDing において、非決定的な関数がないこと。これは、索引によってファクト表の述部を適用し、半結合を促進するためです。
  - 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
    - ただ 1 つの表とされていること
    - 少なくとも 2 つのディメンション表、またはスノーフレークという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、またはスノーフレーク形式で結合可能
  - ディメンション表またはスノーフレークについて
    - 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きます)

- ファクト表索引内の先行列を使用するファクト表に対して、結合述部を持つこと。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。

オプ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。

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、オプティマイザー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フレーク・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 プロセッサーは、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。

上記の述部は、以下に示す述部に変わります。ファクト表の列 ″SID_0CALMONTH″ について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、インフォメーショナル制約を使用する方がより効率的かもしれません。

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、インフォメーショナル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。

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、最初の入力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、入力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、またはレガシー JDBC ドライバーを使用する JDBC アプリケーションの場合、`db2cli.ini` 構成ファイルで REOPT キーワード設定を使用します。各値および対応するオプションは、以下のとおりです。
  - 2 - NONE
  - 3 - ONCE
  - 4 - ALWAYS
- JCC Universal Driver を使用する JDBC アプリケーションの場合、以下の方法の 1 つを使用します。
  - 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
```

- SQL PL プロシージャーの場合、以下の方法の 1 つを使用します。
  - SET_ROUTINE_OPTS ストアード・プロシージャーを使用して、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 プロシージャー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_PREPOPTS** レジストリー変数を使用して、SQL PL プロシージャー・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・プロシージャーを使用して設定された値は、**DB2_SQLROUTINE_PREPOPTS** 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、オプ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に要求することができます。使用される最適化手法の数を削減する場合、最適化の際に使用される時間およびリソースも削減されます。

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プ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、オプ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貪欲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ってはなりません。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ー域が確立されます。

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、クラスター索引操作の際に適切なページにフリー・スペースが検出される可能性が高くなります。

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ー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。
- 位置指定更新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。
- 検索済み更新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いてはならず、表またはビューの 1 つの列を識別している必要があります。

以下の場合は、FOR UPDATE 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

DB2 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。

DB2 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になります。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 *貪欲型結合列挙* 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。貪欲型結合列挙方法を使用すると、最初の数行の素早い検索には役に立たない複数表結合のアクセス・プランになることがあります。

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

データがニックネームから選択されるとき、結果はデータ・ソース・サポートによって異なります。ニックネームによって参照されるデータ・ソースが OPTIMIZE FOR 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りなく、*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。

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。

DB2 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 (DB2 CLI または ODBC) を使用するパッケージ・アプリケーションがある場合は、db2cli.ini 構成ファイルにある **CURSORHOLD** キーワードを用いて、DB2 CLI にすべての宣言されているカーソルについて WITH HOLD 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・マネージャーのオーバーヘッドを削減します。

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トに物理的にマップされるマルチディメンション (MDC) 表ブロックではありません。

行ブロッキングは、BIND 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ー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キ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キ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。

行ブロッキングを指定するには、次のようにします。

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

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

     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。

     ```
     Rows per block = rqrioblk / 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ー (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ー (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ーがアクセスでき、複雑な SQL 要求の実行を複数のプロセッサー間で分け合って実行できます。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 特殊レジスターまたは 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。プロセッサーの数と並列処理の多重度として選択された値との間には、厳密な関係はありません。マシン上のプロセッサー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。ユニプロセッサー・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ーのオーバーヘッドが増えるという点にご注意ください。

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

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

照会で DEGREE = ANY を指定してコンパイルすると、データベース・マネージャーによってパーティション内並列処理の多重度が選ばれます。多重度は、プロセッサー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ー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。

オプ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、ユニプロセッサー・マシンで入出力制約の照会を実行する場合でも、2 度以上を宣言しておいた方が有利です。この場合、プロセッサーは入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キングおよび表レベルのロッキングを適宜使用します。

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

  これらの異なる分離レベルは、非コミット・データへのアクセス、更新消失の防止、データの反復不能読み取りの許可、および幻像読み取りの防止を制御するために使用されます。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。
- ロッキング専用のメモリーの量は、**locklist** データベース構成パラメーターにより制御されます。ロック・リストが満杯になると、ロック・エスカレーションが発生して、データベース内の共用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。

バッファー・プールの作成、変更、またはドロップ時には、バッファー・プールのロック (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、バッファー・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、行レベルのロッキングが使用されます。

- 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
- 検索型の更新または削除操作にディメンション列の述部のみが関係する場合

行ロッキングの期間は、使用されている分離レベルによって異なります。

- UR スキャン。行データが変更されていない限り行ロックは保持されません。
- CS スキャン。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) 表にはブロック・ロックを設定することもでき、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 11 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 11. 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はできますが、更新はできません。 他のアプリケーションは、その表を読み取ったり更新したりできます。
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はできますが、更新はできません。このロックは、S ロックの代わりに、表の行に対して獲得されます。この場合、アプリケーションの分離レベルは、RS か CS のいずれかです。
NW (次キーの弱い排他)	行	行が索引に挿入されるとき、NW ロックが次の行に獲得されます。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はできますが、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はできますが、更新はできません。
X (排他)	行、ブロック、表、バッファー・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とき、表の索引が作成またはドロップされるとき、あるいは表の一部のタイプが再編成されるときなど、特定の状況においてです。同時に実行される他のアプリケーションは、その表を読み取ったり更新したりできません。

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、共用ロック (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、共用および更新ロック (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。

範囲クラスター表のロックインの作業は、標準キー・ロッキングの作業とは若干異なります。範囲クラスター表内で行範囲にアクセスする場合、範囲指定した行の一部が空であっても、範囲内のすべての行がロックされます。標準キー・ロッキングの場合、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、それによりロッキングのシナリオがさらに複雑になることを示唆しています。ロック獲得のタイミングおよびロックの持続性は、分離レベルに依存します。反復可能読み取り (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 12 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 12. ロック・タイプの互換性

	保持されているリソースの状態										
要求されている状態	**なし**	**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 12. ロック・タイプの互換性 (続き)

	保持されているリソースの状態										
要求されている状態	**なし**	**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とき、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化索引の場合、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. ブロック・レベルのロックも、マルチディメンション・クラスタリング (MDC) 表で使用可能です。
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 13. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 14. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 15. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 16. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 17. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 18. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S	IX/S	IX/X	IX/S	IX/X

表 18. 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 (続き)

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 19. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S	IX/S		X/-	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

表 20. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 21. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S	IX/S		IX/S	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

表 22. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
RR	IN/-	IX/S	IX/X	IX/S	IX/X

表 22. 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 (続き)

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 23. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S	IX/S		IX/X	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 RID 索引スキャンのロック・モード

表または RID 索引スキャン中に、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 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 25. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 26. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 27. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 28. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 29. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 30. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 31. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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

表 31. 索引述部のみでの RID 索引スキャンのロック・モード (続き)

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 32. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 33. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 34. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 35. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 36. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 37. 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

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 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: 開始述部と停止述部のみの RID 索引スキャン後

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 ブロック索引スキャンのロック・モード

ブロック索引スキャン中に、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ャ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索引スキャンのロック・モード
- 表 3. 開始述部と停止述部のみでの索引スキャンのロック・モード
- 表 4. 述部での索引スキャンのロック・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ャンのロック・モード: 述部なしでのブロック索引スキャン
- 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ャンのロック・モード: 述部なしでのブロック索引スキャン後
- 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ャンのロック・モード: ディメンション列上の述部のみのブロック索引スキャン
- 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: ディメンション列上の述部のみのブロック索引スキャン後
- 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: 開始述部と停止述部のみのブロック索引スキャン
- 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: 開始述部と停止述部のみのブロック索引スキャン後
- 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: 他の述部 (sargs、resids) でのブロック索引スキャン
- 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: 他の述部 (sargs、resids) でのブロック索引スキャン後

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 39. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 40. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 41. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 42. 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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 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ャンのロック・モード: 述部なしでのブロック索引スキャン

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 44. 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: 述部なしでのブロック索引スキャン後

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 45. 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: ディメンション列上の述部のみでのブロック索引スキャン

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/--	IX/IX/--		IX/S/--	

表 45. 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: ディメンション列上の述部のみでのブロック索引スキャン (続き)

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

表 46. 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: ディメンション列上の述部のみでのブロック索引スキャン後

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 47. 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ブロック索引スキャン

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 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ャンのロック・モード: 開始述部と停止述部のみのブロック索引スキャン後

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 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ャンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキャン

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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 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ャンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキャン後

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		検索型更新または削除	
		スキャン	現在の場所	スキャン	更新または削除
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) の索引のスキャンは、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用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用、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化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

パーティション化索引の場合、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・ユーティリティー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 (意図的排他共用) 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もあります。索引アクセスによって行が読み取られ、S としてロックされる場合、行を含む表には、目的ロックが含まれています。ただし、ロック・タイプが IX ではなく IS である場合、後で行が変更されると、表ロックは IX に変換され、行ロックは X に変換されることになります。

ロック変換は、通常、照会が実行されるときに暗黙的に行われます。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ケーションがコミットできるようにロックを解放するということをせずに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。ロック要求がペンディングにされている時間が **locktimeout** 値より長くなると、要求しているアプリケーションにエラーが戻り、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、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。

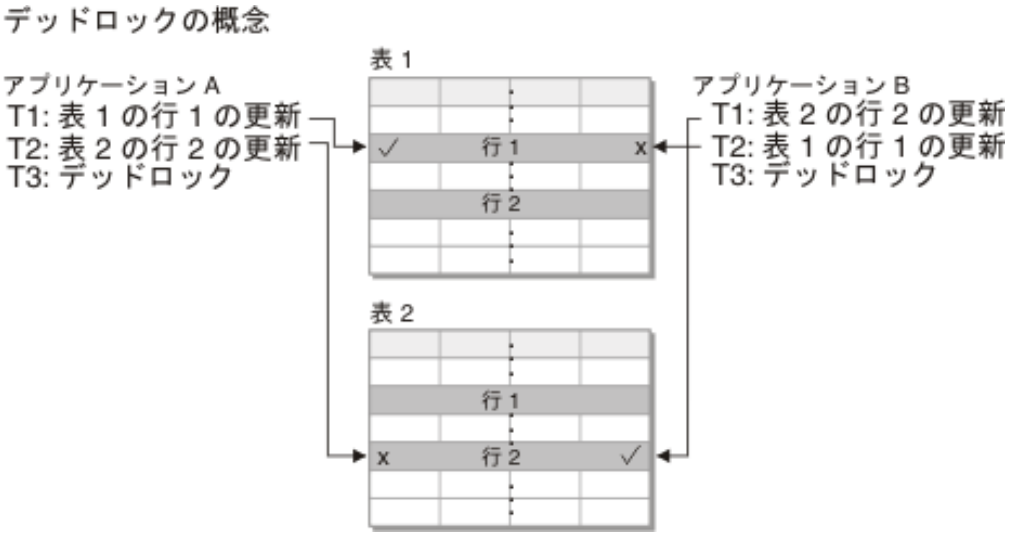


図 23. アプリケーション間のデッドロック

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、 212 ページの図 24 と以下の説明文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に該当することに注意してください。

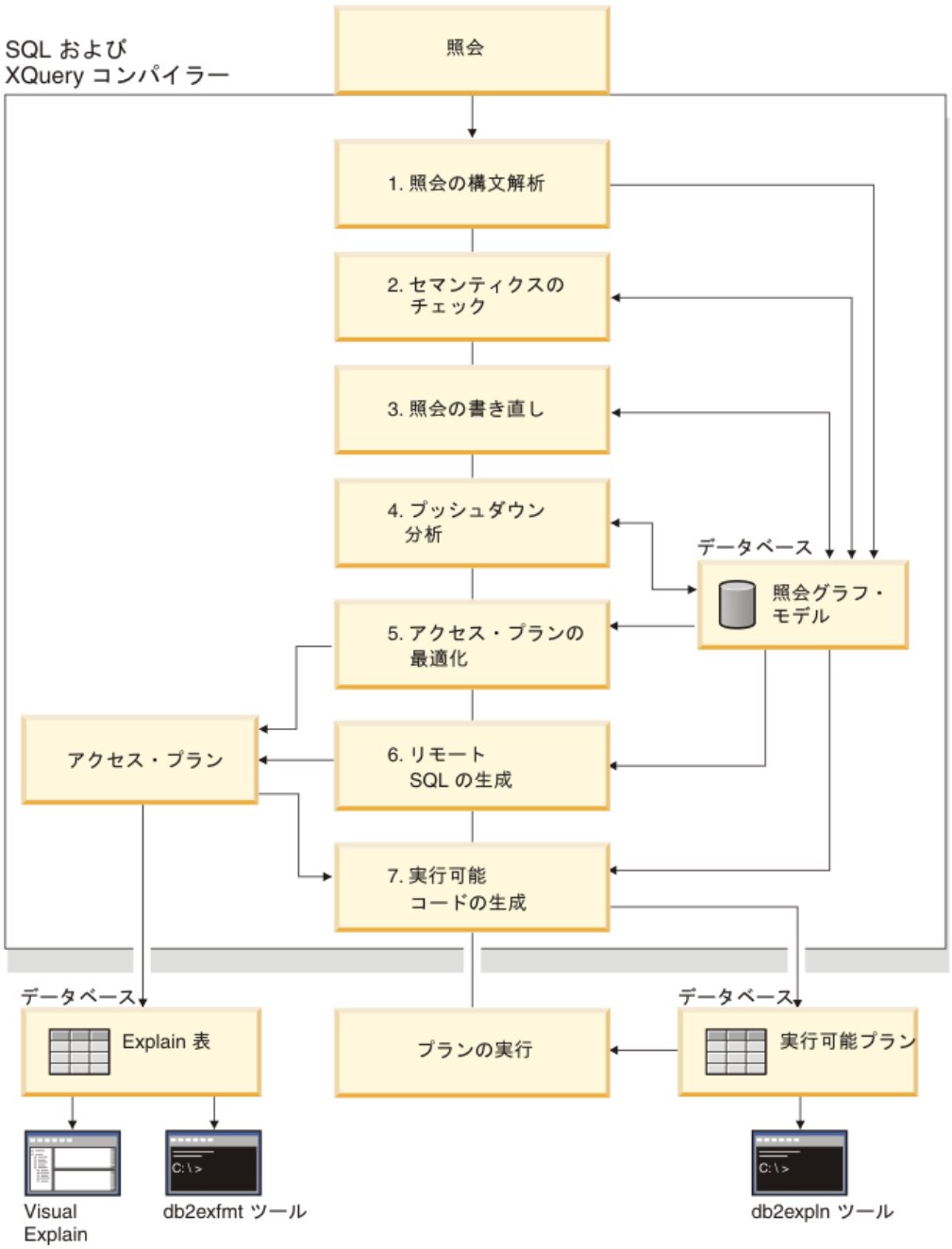


図 24. SQL および XQuery コンパイラーによって実行されるステップ

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、以下の照会操作では、計算量が多くなります。

   - 集約
   - 行の再配分
   - 副照会の外側にある表の列への参照を含む副照会である相関副照会

   パーティション・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・サイズの選択時に、別のサイズのバッファー・プールが存在するかどうかも考慮されます。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ー (SMP) 環境で照会内並列処理の可能性があるかどうかが考慮されます。この情報は、オプティマイザーが照会にとって最適のアクセス・プランを選択するのに使用されます。

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)

   オプティマイザーで選択した最終プランは、リモート・データ・ソースに対して実行される一連のステップで構成されています。リモート SQL 生成のステップでは、データ・ソースごとに実行される操作のために、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、コンパイラーはアクセス・プランと照会グラフ・モデルを使用して、照会の実行可能なアクセス・プラン、つまりセクションを作成します。このコード生成ステップでは、照会グラフ・モデルの情報を使用して、1 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、特殊レジスター、またはパラメーター・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、特殊レジスター、またはパラメーター・マーカーを含んだステートメントのアクセス・パスは、コンパイラー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・マネージャーはシステム・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、db2expln 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。オプティマイザー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オプティマイザーが想定している) 場合には、オプティマイザーは、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、照会コンパイラーは 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 機能または Visual Explain 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 - 共用集約

    照会がさまざまな関数を使用している場合は、書き直しによって、実行する必要のある計算の数を減らすことができます。

- 操作の移動

  照会を最小限の操作数と述部数で構成するため、コンパイラーは照会を書き直して照会操作を移動します。以下の例に、移動可能な操作をいくつか示します。

  – 例 - DISTINCT の除去

    照会書き直し時には、オプティマイザーで DISTINCT 操作の位置を移動して、その操作のコストを少なくすることができます。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、オプティマイザー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

    照会の書き直し時に述部をその照会に追加することによって、オプティマイザーが照会の最適アクセス・プランを選択するときに、追加の表結合についても検討できるようになります。

  – 例 - OR から IN への変換

    照会書き直し時には、より効率的なアクセス・プランのために、OR 述部を IN 述部に変換することができます。また、IN 述部を OR 述部に変換すれば一層効率的なアクセス・プランが作成されるのであれば、 SQL 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, firstname, 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ても、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 - 共用集約

照会の中で複数の関数を使用すると、複数の計算が生成されますが、これは場合によってはかなり時間を要します。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りません。

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_prog a
  where
    p.empno = e.empno and
    p.projname like '%PROGRAMMING%' and
    p.projno = a.projno and
    e.salary+e.bonus+e.comm < a.avg_comp
```

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。

コンパイラー書き直しの例: 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                    Document 2

<customer>                    <customer>
  <name>John</name>             <name>Michael</name>
  <lastname>Doe</lastname>      <lastname>Miller </lastname>
  <date_of_birth>               <date_of_birth>
   1976-10-10                   1975-01-01
  </date_of_birth>              </date_of_birth>
  <address>                     <address>
    <zip>95141.0</zip>            <zip>95142.0</zip>
  </address>                    </address>
<volume>80000.0</volume>      <volume>100000.00</volume>
</customer>                   </customer>
<customer>                    <customer>
  <name>Jane</name>             <name>Michaela</name>
  <lastname>Doe</lastname>      <lastname>Miller</lastname>
  <date_of_birth>               <date_of_birth>
    1975-01-01                   1980-12-23
  </date_of_birth>              </date_of_birth>
  <address>                     <address>
    <zip>95141.4</zip>            <zip>95140.5</zip>
  </address>                    </address>
<volume>50000.00</volume>     <volume>100000</volume>
</customer>                   </customer>
```

例 - 整数述部のプッシュ

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、不要な人物を事前にフィルターに掛けるためには、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 - 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、不要な人物を事前にフィルターに掛けるためには、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。

```
T.XMLCOL/customer/lastname[. > rtrim("Miller") and . < blank_padd("Miller",
  max(20,length("Miller"))]
```

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、末尾ブランクを含め、すべての考えられる『Miller』ストリング以上でなければなりません。元のストリングはその列の最大長までブランクで埋められるか、比較ストリングの方が長ければその長さまでブランクで埋められます。

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 double 索引を使用し、不要な人物を事前にフィルターに掛けるためには、volume = 100000.00 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/volume[.=100000.00]
```

この述部は範囲フィルターにトランスフォームする必要はありません。キャスト制約によって XML 値はターゲット SQL タイプと同じ精度で、同じ小数部分の長さを持つようにされるからです。この制約に違反すると、エラーが戻ります。精度は、DOUBLE 値が DECIMAL(*x*,*y*) にキャストされても減少することはありません。比較値の丸めまたは切り捨ては不要です。

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に、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 つのエレメントです。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。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すと、次のリストのようになります。

1. 範囲区切り述部
2. 索引 SARGable 述部
3. データ SARGable 述部
4. Residual 述部

SARGable (検索引数述部) は、検索引数 (*s*earch *arg*ument) として使用できる述部です。

表 51 は述部カテゴリーの特性の要約です。

表 51. 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ャ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変わりません。

データ SARGable 述部

索引マネージャーによっては評価できないものの、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す。

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。

関数がリモート・データ・ソースにプッシュダウンできない場合、その関数は、照会のパフォーマンスに大きな影響を与える可能性があります。選択述部をデータ・ソースで評価したときではなく、ローカルに評価したときの影響を考慮してください。このような評価では、DB2 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が、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・マーカーをリモート SQL ステートメントへの値にバインドする必要があります。したがって、パラメーター・マーカーの制限を調べ、各データ・ソースがそのようなバインド機構をサポートできることを確かめる必要があります。ある関数の値をバインドする適切な方法を DB2 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うと、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もあります。
- 関数は、リモート側で評価されると違う値を戻すことがあります。この状況の一例として、より大きい ('>') 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。その後コンパイラー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、異なってはい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 52 は、これらのオプションについて説明しています。

表 52. 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	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。 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。	N
VARCHAR_NO_TRAILING_BLANKS	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。 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。	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 ユーティリティーにより、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。 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・プロパティーを変更し、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 演算子への入力はリモートに評価されますか? 評価されない場合、入力を調べてください。
  – そのデータ・ソースには、この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 演算への入力はリモートに評価されますか? 評価されない場合、入力を調べてください。
  - ORDER BY 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。

オプティマイザーは、プッシュダウン分析の出力を使用して、各操作が DB2 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伝送するときのコストも含まれています。

目標は最適化された照会を生成することですが、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、プロセッサー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)*/
    col1
  from table1
```

プラン・ヒントは、ストリング `/*+ INDEX (table1, t1index)*/` です。

- DB2 オプティマイザー・ナレッジ・ベースでの情報

  DB2 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 カタログ・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 ユーティリティーにより、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・オプ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。そのプランに関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に、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。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。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ー・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化索引 (特定の場合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がテストされます。
- 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・マーカー、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なります。

- 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
```

- 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

  索引スキャンの範囲を制限する場合、広義の不等比較述部が含まれ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。

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。範囲クラスター表の場合は、データ・レコードの位置を計算する「仮想」索引を介してデータにアクセスできます。

索引のみのアクセス

場合によっては、表にアクセスせずに、索引からすべての必須データを検索できることがあります。これは、*索引のみのアクセス* と呼ばれます。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、必要な列が索引内に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ブを行うと、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ーがあるか、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に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、オプティマイザーが想定した値よりずっと少ないメモリーで作動しなくてはなりません。これを解決するには、**sheapthres** の値を **sortheap** と比較して増加させます。

オプティマイザーは、単一の表にアクセスするのに、索引 ANDing 操作と索引 ORing 操作を組み合わせることはしません。

範囲クラスター表での索引アクセス

標準的な表とは異なり、(従来の B ツリー索引のような) 行にキー値をマップする物理索引は、範囲クラスター表には不要です。その代わりに、列ドメインが持つ順次性を利用し、表内の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。

オプティマイザー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。範囲クラスター表には元のキー値配列が保持されるので、表内の行のクラスター化は保証されます。

索引アクセスとクラスター率

アクセス・プランを選択するとき、オプティマイザーはディスクからページをバッファー・プールに取り出すのに必要な入出力の数を見積もります。この見積もりには、バッファー・プール使用率の予測も含まれています。既にバッファー・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、オプティマイザーは、システム・カタログの情報を使用して、データ・ページをバッファー・プール内に読み込むための入出力コストを見積もります。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・ページがバッファーにある内にそのページから読み取ることができます。この列の値が -1 の場合、オプティマイザーは、使用可能なら PAGE_FETCH_PAIRS および CLUSTERFACTOR 情報を使用します。
- PAGE_FETCH_PAIRS 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ー・プールに読み込むのに必要な入出力の数をモデル化するための数値の対がいくつか含まれています。これらの列のデータは、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。

索引のクラスタリング統計が使用不可である場合、オプティマイザーはデフォルト値を使います。この値は、索引に関するデータのクラスタリング率が低いことを想定しています。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るため、定期的に表を再編成することが必要な場合があります。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、バッファー・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、既にバッファー・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。 244 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。

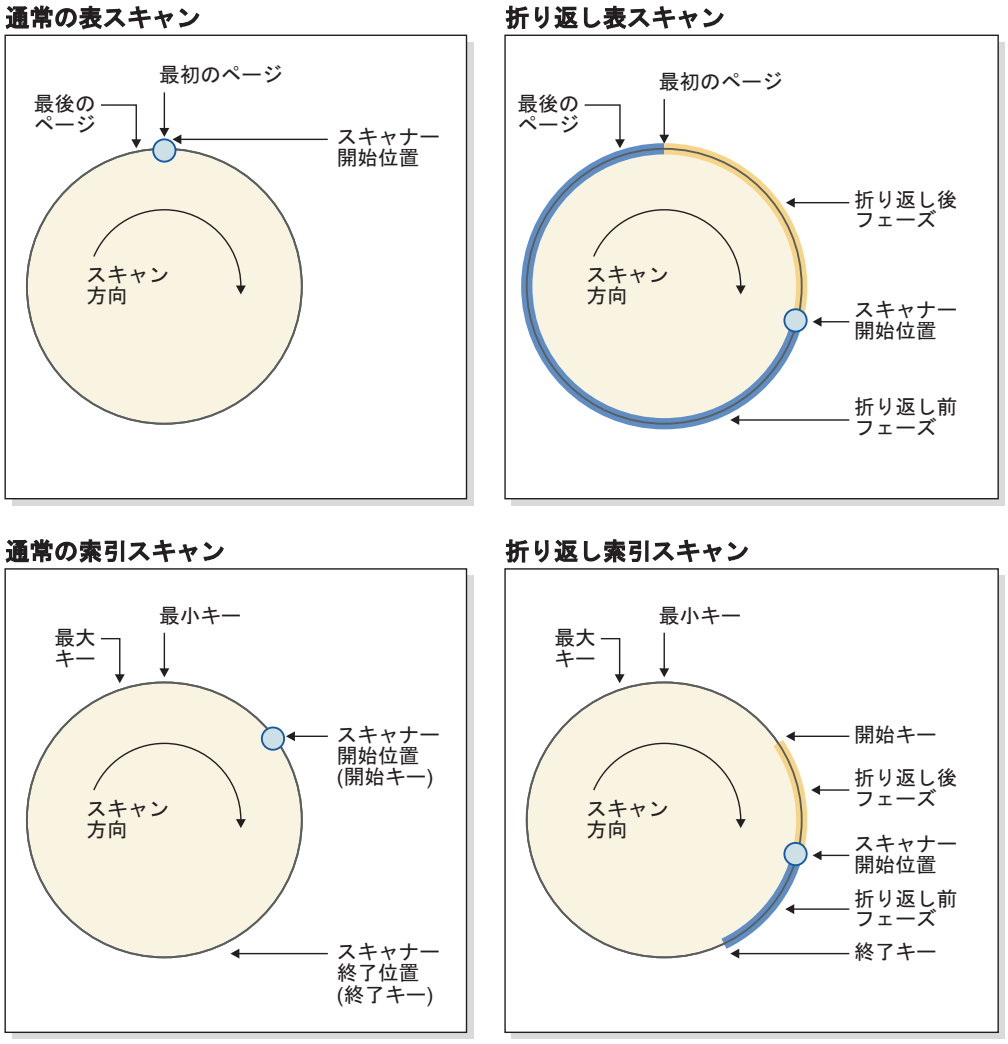


図 25. 通常のスキャンとラッピング・スキャンの概念ビュー

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ー・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、バッファー・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ー・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、バッファー・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。

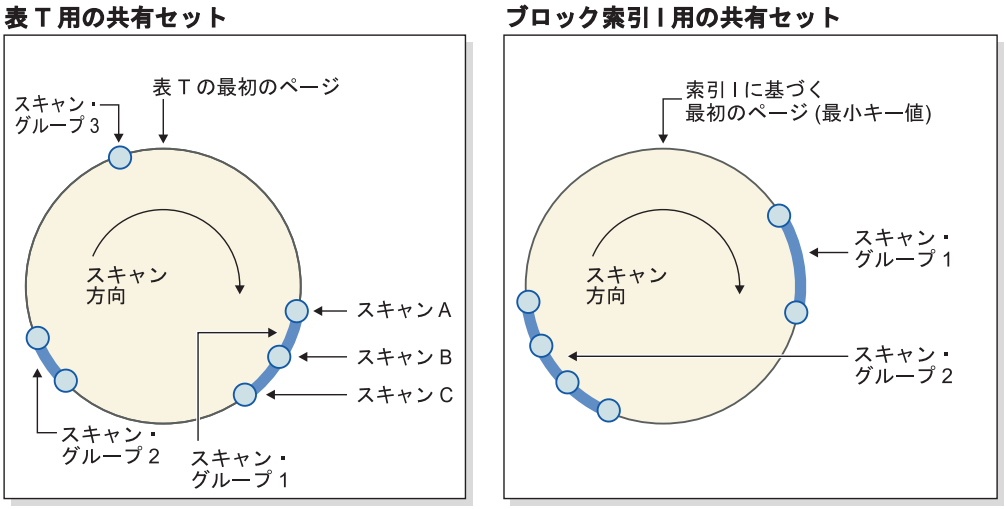


図 26. 表とブロック索引のスキャン・シェアリングの共有セット

またこの図は、バッファー・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 がバッファー・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。

オプティマイザー・プロファイルを使用すると、コンパイラーがスキャン・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にのみ使用します。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 (ソートを起動する場合がある) 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。

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

PROJ	PROJ_ID
A	1
B	2
C	3
D	4

TABLE2

PROJ_ID	名前
1	Sam
3	Joe
4	Mary
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、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最小限にとどめることができます。

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* とも呼ばれる) 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` の形式の述部が 1 つ以上必要で、これらの列のタイプは同じでなければなりません。タイプが CHAR の列の場合は、長さが同じでなければなりません。タイプが DECIMAL の列の場合は、精度と位取りが同じでなければなりません。タイプが DECFLOAT の列の場合は、精度が同じでなければなりません。列を LONG フィールド列や LOB 列にすることはできません。

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、初期外部結合、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、貪欲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 が使用する方法と似ています。

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、オプ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。デカルト積を計算するプランが見積もりで最低コストである場合は、そのプランがオプティマイザーによって選択されます。

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、外部キー索引に関係するシナリオもあります。ファクト表の外部キー列が単一列索引で、全ディメンション表をまたがって比較的高い選択可能性がある場合には、以下に示すようなスター・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。

初期外部結合

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、初期外部結合を選択できます。

初期外部結合は、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戻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。

初期外部結合は、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。

初期外部結合も、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とを戻します。

```
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

一対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、特に貪欲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、オプティマイザーは結合の内部表となる表の最大数に異なる制約を課します。複合内部結合は最適化クラス 5、7、および 9 で使用できます。

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 ユーティリティー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。

コロケーテ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テッド結合のために正しくパーティション化されます。他のディメンション表はコロケーテッド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 `dim1.c1 = 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 `dim2.c2 = 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もあります。

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ー間) 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、コロケーテッド結合、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。

コロケーテッド結合

コロケーテ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。オプティマイザーがコロケーテッド結合を処理するためには、結合される表は連結され、対応する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。 257 ページの図 27 に例を示します。

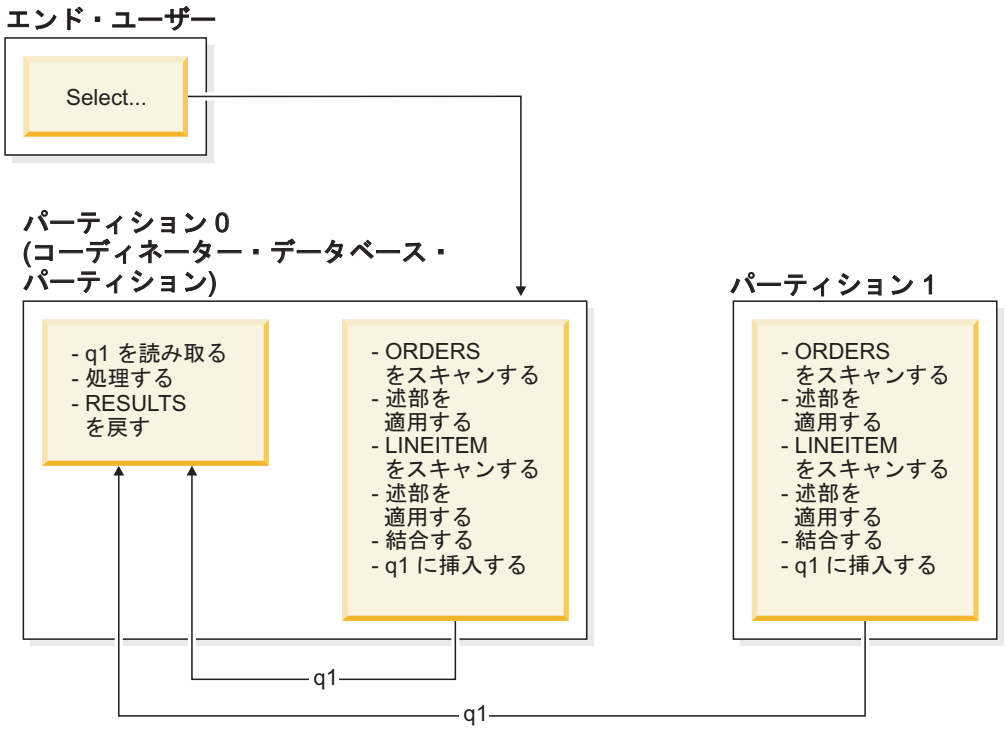


図 27. コロケーテッド結合の例

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) はコロケーテ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。 258 ページの図 28 に例を示します。

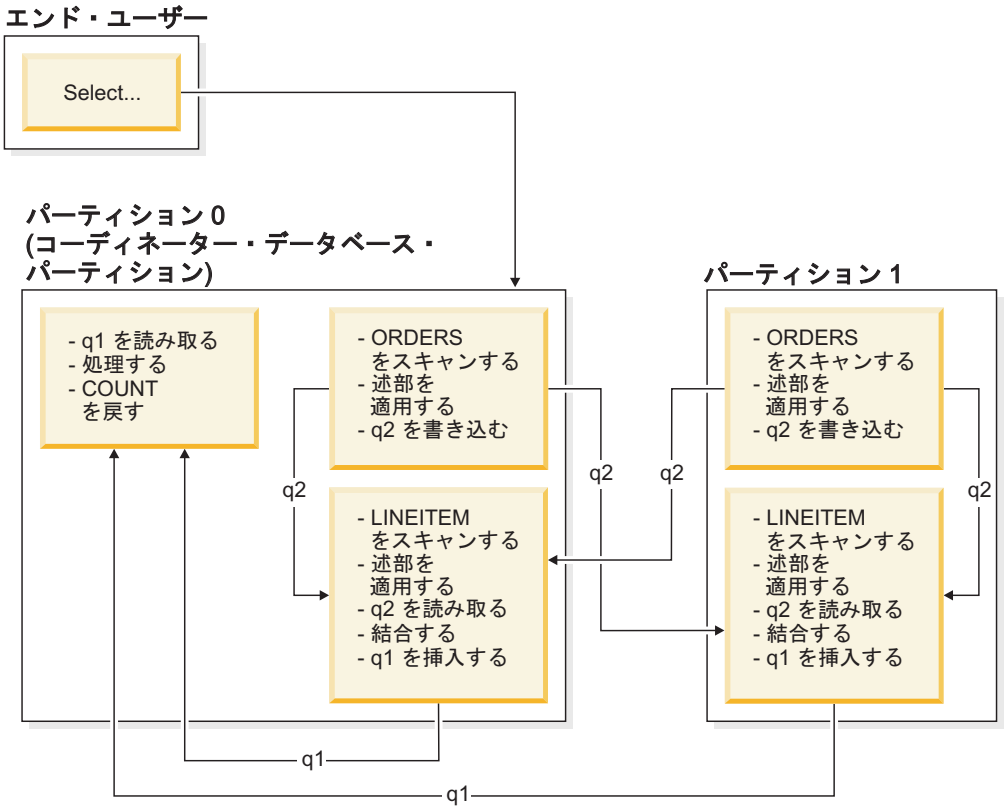
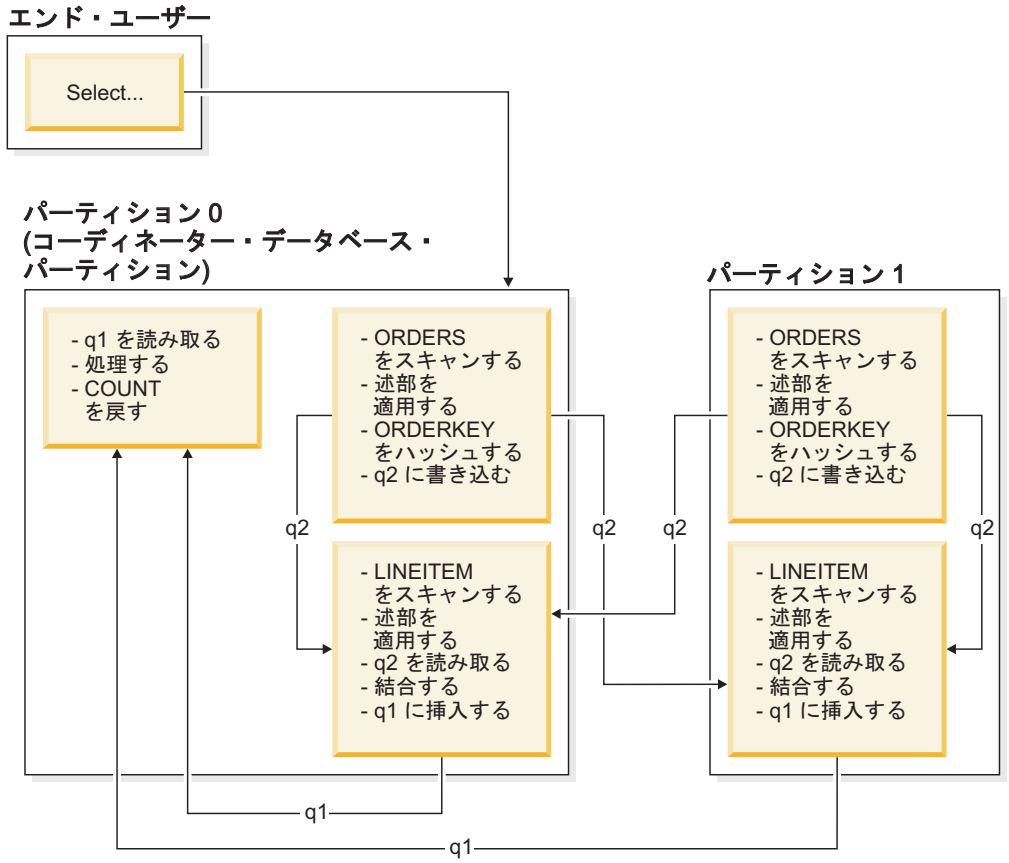


図 28. 外部表のブロードキャスト結合の例

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分に送ります。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。259 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。 260 ページの図 30 に例を示します。

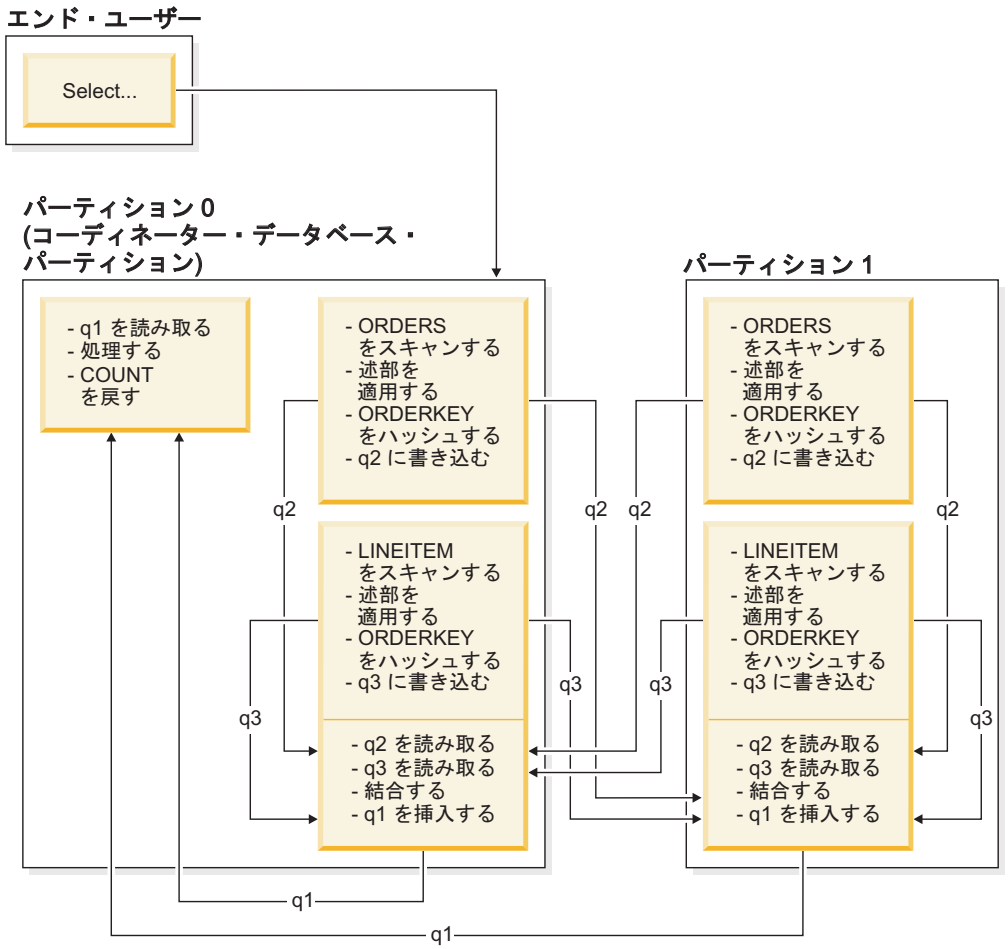
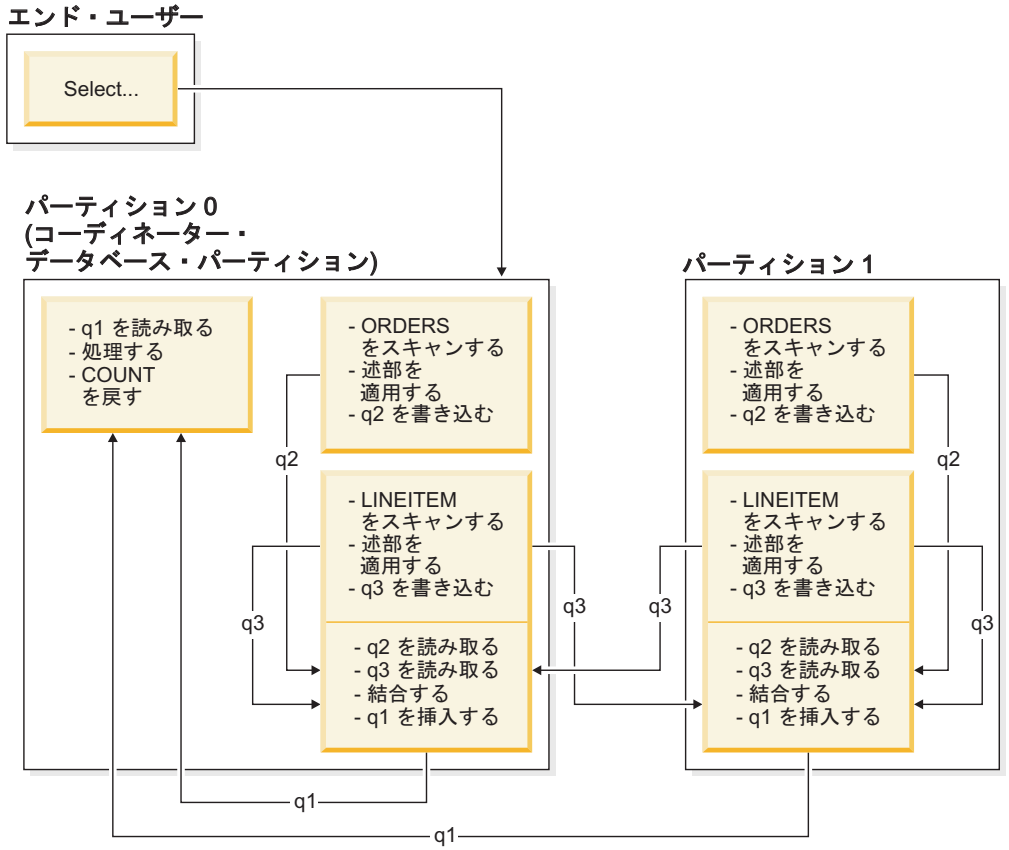


図 30. 内部表および外部表の指示結合の例

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

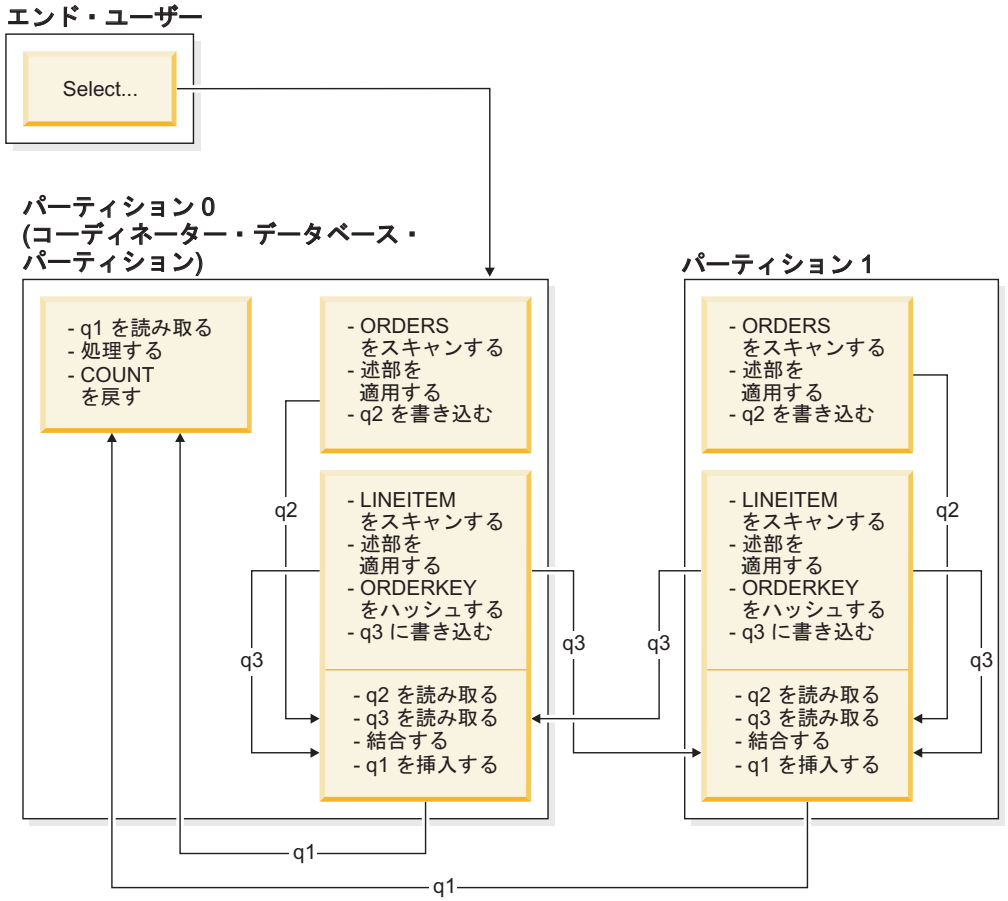
内部表のブロードキャスト結合方式では、内部表が外部表のすべてのデータベース・パーティションに対してブロードキャストされます。 261 ページの図 31 に例を示します。



ORDERS 表は、LINEITEM 表を持つデータベース・パーティションすべてに送られる。表キュー q3 は、外部表のデータベース・パーティションすべてにブロードキャストされる。
図 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。 262 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、ソートする行の数、キー・サイズ、および行の幅を含め、いくつかの要因によって違ってきます。ソートされる行が、ソート・ヒープ内で使用可能なスペースより多くのスペースを占める場合は、複数のソート・パスが実行されます。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。各パスはバッファー・プール内の一時表に記憶されます。バッファー・プールに十分のスペースがない場合、この一時表のページはディスクに書き込みます。すべてのソート・パスが完了すると、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の数が多い場合は有利です。ソート中に行われる何らかのグループ化により、ソートをディスクにスピルさせる必要がなくなっているか少なくなっている場合はさらに有利です。

ソート中の集約には、正しい結果を戻すために、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ジを終了するときに、似たようなケースがパーティション内並列処理で存在します。
- 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には、索引は、索引キー値およびキー値あたりの索引項目数に基づいて、レコード範囲に分割されます。並列索引スキャンは、並列表スキャ

ンと同様に、レコード範囲を割り当てられたサブエージェントを使用して行われます。サブエージェントには、現行の範囲での作業が完了した時点で新しい範囲が割り当てられます。

オプティマイザーは、スキャンの単位 (ページまたは行) と細分度を決定します。

並列スキャンは、サブエージェント間で均等になるように作業を分散します。並列スキャンの目標は、サブエージェント間の負荷を均衡させて、サブエージェントが同等に使用されるようにすることです。使用中のサブエージェントの数が使用可能なプロセッサーの数と等しく、ディスクが入出力要求で過度に作動しているということがない場合には、マシン・リソースは効率的に使用されていると言えます。

他のアクセス・プラン方式によっては、照会の実行時にデータの不均衡が生じることがあります。オプティマイザーは、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

オプティマイザーは、以下のいずれかの並列ソート方式を選択します。

- ラウンドロビン・ソート

  このソートは、*再配分ソート* とも呼ばれます。このソート方式では、共用メモリーを効率的に使用し、すべてのサブエージェントに対して可能な限り均一にデータを再配分します。このソートは、ラウンドロビン・アルゴリズムを使用して、均等な分散を行います。まず最初に、各サブエージェントごとに個々のソートを作成します。挿入フェーズでは、サブエージェントが、ラウンドロビン様式で個々のソートにそれぞれデータを挿入していくことによって、より均等なデータの分散を行います。

- パーティション・ソート

  このソートは、ソートが各サブエージェントごとに作成されるという点では、ラウンドロビン・ソートに似た働きをします。このソートでは、サブエージェントはハッシュ関数をソート列に適用して、行をどのソートに挿入するかを判別します。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。

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 DELETE であって、定位置 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格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、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ャンは、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(
  subsdate date, province char(2), accountid int)
  partition by range(subsdate) (
    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 subsdate between '1/1/2000' and '12/31/2000'
```

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。

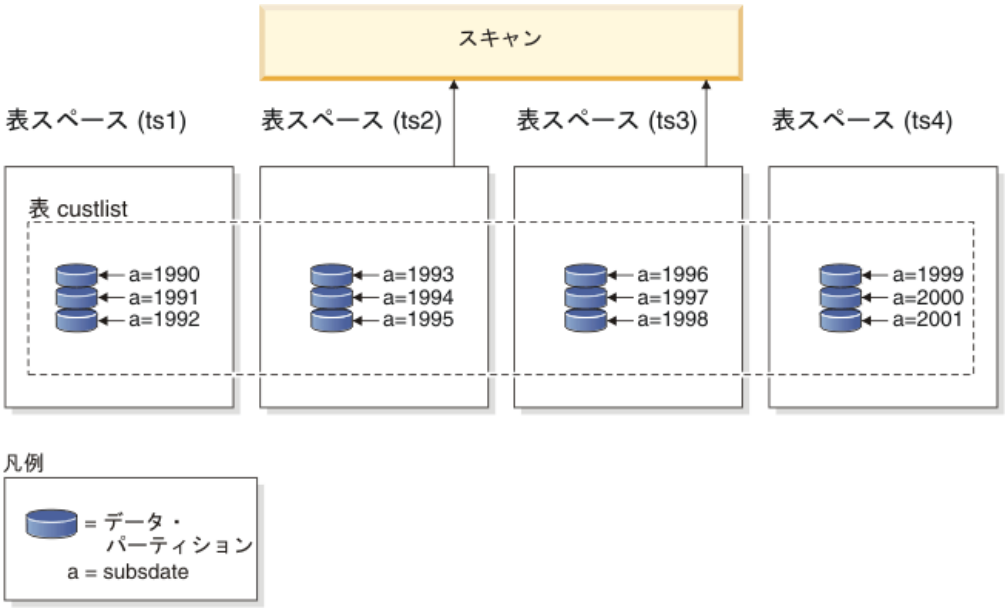


図 33. データ・パーティションの除去のパフォーマンス上の利点

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

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。

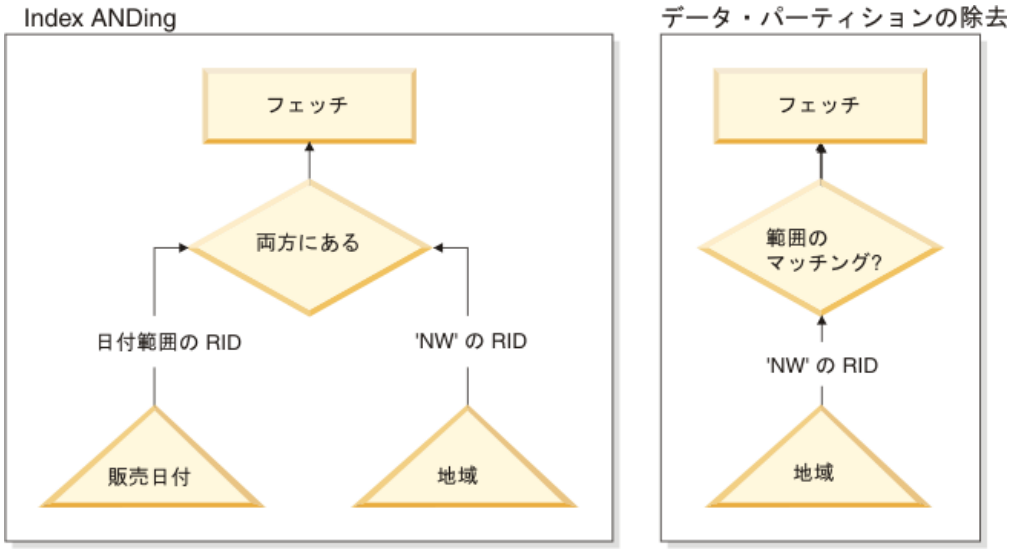


図 34. 表のパーティション化と索引 ANDing の両方のオプティマイザーの決定パス

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 subsdate 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・マーカー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 ユーティリティー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す。次の例は、製品グループと製品ラインのディメンション、市町村、都道府県、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 pgname,
      year(pdate) as year, month(pdate) as month,
      t.status,
      sum(ti.amount) as amount,
      count(*) as count
    from cube.transitem as ti, cube.trans as t,
      cube.loc as loc, cube.pgroup as pg, cube.prodline as l
    where
      ti.transid = t.id and
      ti.pgid = pg.id and
      pg.lineid = l.id and
      t.locid = loc.id and
      year(pdate) > 1990
    group by l.id, pg.id, loc.country, loc.state, loc.city,
      year(pdate), month(pdate), t.status, l.name, pg.name
  )
data initially deferred refresh deferred;

refresh table dba.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 が作成されるときに 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ーが選択するアクセス・プランについての、詳細な情報が提供されます。

その情報にはアクセス・プランを選択するために使用される決定の基準が記述されます。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 情報は以下のこと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。

キャプチャーされた情報には、次のものが含まれます。

- 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。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、コマンド行ツールまたは Visual Explain のいずれかを使用できます。使用するツールによって、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。例えば、 Visual Explain のみを使用する必要がある場合、スナップショット情報だけをキャプチャーする必要があります。 Explain 表に対して、コマンド行ユーティリティーのいずれかを使用するか、またはカスタム SQL または XQuery ステートメントを使用して、詳細な分析を実行する必要がある場合、すべての Explain 情報をキャプチャーしなければなりません。

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. この情報を表示するために Visual Explain にアクセスできない場合は、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実行するときに、この情報を収集することができます。静的 SQL ステートメントの場合は、バインド時にこの情報を収集することもできます。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。 Visual Explain または 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、コンテナーの追加、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 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 ALWAYS を使用してパッケージをバインドすると、ホスト変数、パラメーター・マーカー、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、Visual Explain が求める書式で EXPLAIN_STATEMENT 表の SNAPSHOT 列に保管されます。この書式は他のアプリケーションでは使用できません。 Explain スナップショットに関する付加的な情報は、Visual Explain 自体から使用できます。この情報には、データ・オブジェクトおよびデータ演算子に関する情報が含まれています。

SQL または XQuery ステートメントがコンパイルされ、Explain データが要求されていると、以下のように、Explain スナップショット・データがキャプチャーされます。

- 静的または追加バインド SQL 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になります)。

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・プロシージャーは、以下のリストに挙げるとおり、4 つあります。プロシージャーの違いは、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、アクティビティー ID、UOW ID、およびアクティビティー・イベント・モニター名を取ります。プロシージャーは、アクティビティー・イベント・モニターでこのアクティビティーに対応するセクションを検索します (SQL アクティビティー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。プロシージャーは、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

プロシージャーは、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、プロシージャーは、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。プロシージャー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 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 ユーティリティー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。

ストアード・プロシージャー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 プロシージャーに提供されなかった場合)、EXPLAIN_LEVEL が O 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。

- COLUMN_NAMES
- SINGLE_NODE
- PARTITION_COLUMNS
- SEQUENCE_SIZES

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
- PREDICATE_ID

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、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ー参照オブジェクト (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、アクティビティー・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 ストアード・プロシージャー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す。セクション実行時統計の使用可能化は、**DB2_SYSTEM_MONITOR_SETTINGS** レジストリー変数の SECTION_ACTUALS パラメーターを使って行います。セクション実行時統計を使用可能にするには、パラメーターを TRUE に設定します (デフォルト値は FALSE です)。例えば、

```
db2set DB2_SYSTEM_MONITOR_SETTINGS=SECTION_ACTUALS:TRUE
```

レジストリー変数の設定は動的です。レジストリー変数が TRUE に設定された後、セクション実行時統計はセクションの実行時に更新されます。

注: レジストリー変数を更新する前にアプリケーションによって実行されたステートメントはすべて、同じアプリケーション内で再実行される際、その元のセクション実行時統計の設定を保持します。つまり、セクション実行時統計が使用不可になっている間にアプリケーションがステートメントを発行し、レジストリー変数によってセクション実行時統計を使用可能にしてから同じステートメントを再発行した場合でも、そのステートメントの 2 度目の実行では、セクション実行時統計は使用不可のままです。ステートメントを初めて発行する他のアプリケーションでは、そのステートメントのセクション実行時統計は使用可能となります。

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、アクティビティー・イベント・モニターを使って行います。アクティビティー情報の収集が使用可能になっている場合、アクティビティー・イベント・モニターは、アクティビティーの実行完了時にアクティビティーの詳細を書き出します。アクティビティー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 (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ー・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、アクティビティー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ー・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。

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ーに関する情報をアクティビティー・イベント・モニターに送信するために WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーが使用される場合、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 ストアード・プロシージャーによってアクティビティー・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。
- 反応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 バージョン 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。
- アクティビティー・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 アクティビティー・イベント・モニター表 (特にアクティビティー表) を再作成しておく必要があります。アクティビティー論理グループに SECTION_ACTUALS 列が含まれていない場合、アクティビティー・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 プロシージャーを使って行えます。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ー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 プロシージャー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。

   ```
   db2set DB2_SYSTEM_MONITOR_SETTINGS=SECTION_ACTUALS:TRUE
   ```

2. SYSINSTALLOBJECTS プロシージャーを使って 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す。

3. 以下を発行して、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ー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ーのセクション・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
 GRANT USAGE ON WORKLOAD MYCOLLECTWL TO PUBLIC
```

注: 別個のワークロードを使用することを選択すると、アクティビティー・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。

4. 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ー・イベント・モニターを作成します。

```
CREATE EVENT MONITOR ACTEVMON FOR ACTIVITIES WRITE TO TABLE
```

5. 以下のステートメントを実行して、アクティビティー・イベント・モニター ACTEVMON を活動化します。

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

6. 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、アクティビティー・イベント・モニターによってキャプチャーされます。
7. 以下のステートメントを発行して、アクティビティー・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,
       ACTIIVTY_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 ...
```

8. 以下の CALL ステートメントで示されているように、アクティビティー ID 情報を EXPLAIN_FROM_ACTIVITY プロシージャー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
```

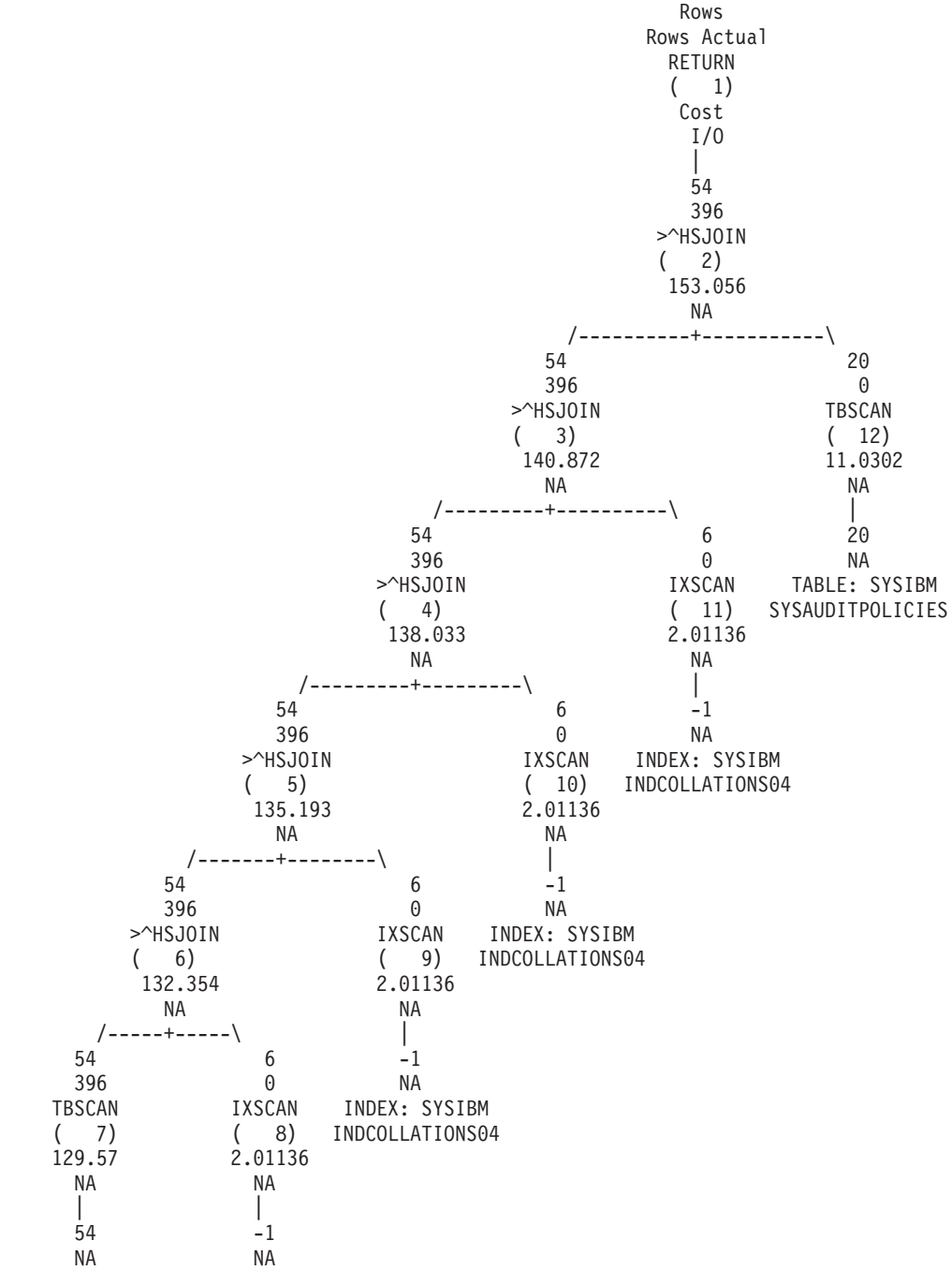
9. 以下のように、db2exfmt コマンドを使用し、EXPLAIN_FROM_ACTIVITY プロシージャーの出力として返された Explain インスタンス・キーを入力として指定することにより、Explain データをフォーマットします。

```
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:
-----------
        Total Cost:             154.035
        Query Degree:           1
```



```
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、オプ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、オプティマイザーは、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。この場合に取る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試行して、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および演算子の詳細が Explain 出力内のどこにあるかを説明します。

db2exfmt グラフ出力のセクション実行時統計

Explain で実行時統計が使用可能な場合、実行時統計はグラフ内の見積行数の下に表示されます。 Explain は演算子に関してのみ実行時統計をサポートし、オブジェクトについてはサポートしません。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。以下は、db2exfmt グラフ出力の例です。

```
        Rows
    Rows Actual
      RETURN
      (   1)
       Cost
        I/O
        |
      3.21948  << オプティマイザーで使用された見積行数
        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  << Explain はオブジェクトに関するセクション実行時統計はサポートしません
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すべてのデータベース・パーティションにおける平均カーディナリティーです。平均は、オプティマイザーによって見積もられる値なので、表示されます。実際の平均を提供することで、見積もりの平均と比較するための意味のある値が得られます。パーティション・データベース環境では、データベース・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で提供されます。ユーザーは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 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)。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、表示される実際のカーディナリティーは、パーティション・データベース環境の平均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 の下に個別に表示されます。セクション Explain Actuals 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、パーティション番号の横のデータベース・パーティションごとの値のリストに NA が表示されます。セクション Output Streams の Actual number of rows も NA になります。以下は、db2exfmt 出力の演算子詳細の例です。

```
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
```

Explain 情報の使用のガイドライン

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. この情報を表示するために Visual Explain にアクセスできない場合は、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実行するときに、この情報を収集することができます。静的 SQL 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、Visual Explain では、索引レベルの数 (NLEVELS 列) が、照会が最初にデータベースにバインドされたときよりもかなり大きくなっていることに気付きます。そこで、以下のアクションのいずれかを実行するように、選択することになります。

- 索引を再編成します。
- 表と索引の新規の統計を収集します。
- 照会を再バインドするときに Explain 情報を収集します。

アクションのいずれかを実行したあとで、アクセス・プランを再度調べます。索引が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、コンテナーの追加、新しいカタログ統計の収集などの、多数のアクションを行うことにより、照会パフォーマンスの向上に寄与することができます。

これらの領域のいずれかで変更を行ってから、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す。

  – 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ー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ー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になります。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。複数行が含まれる結合 (意思決定支援照会での結合など) は、通常、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作業方法を変更することになる酌量すべき状況があります。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 ユーティリティー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プティマイザーが選択するアクセス・プランについての詳細な情報を提供する、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プ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。オプ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 機能で使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 53. 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		

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す。

Explain 表および Explain 情報の編成

Explain インスタンスの概念について、すべての Explain 情報が編成されています。*Explain インスタンス* は 1 つまたは複数の SQL 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た Explain 情報には、コンパイル環境ならびにコンパイルされる SQL または XQuery ステートメントを実行するために選ばれたアクセス・プランが入っています。

例えば、Explain インスタンスは、以下のいずれかから成り立っています。

- 静的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) では、CALL、コンパウンド SQL (動的)、DELETE、INSERT、MERGE、REFRESH TABLE、SELECT、SET INTEGRITY、SELECT INTO、UPDATE、VALUES、および VALUES INTO ステートメントに関する Explain 情報をキャプチャーすることができます。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ことにより、またはセクション Explain インターフェースを使用することにより呼び出される 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たアクセス・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。これらを作成するには、sqllib サブディレクトリーの misc サブディレクトリーにある EXPLAIN.DDL スクリプトを実行します。

Explain 表は、SYSPROC.SYSINSTALLOBJECTS プロシージャーを使用して、作成、ドロップ、および妥当性検査することもできます。この表は、特定のスキーマと表スペースで作成できます。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。この表を 1 人のユーザーに対して定義してから、その定義済みの表を指す別名をそれぞれの追加ユーザーに対して作成することができます。またはその代わりに、Explain 表を SYSTOOLS スキーマ下で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 または XQuery ステートメントの場合)、またはステートメント許可 ID (静的 SQL または XQuery ステートメントの場合) の下に他の Explain 表または別名がない場合、Explain 機能のデフォルトは SYSTOOLS スキーマになります。共通の Explain 表を共用する各ユーザーは、それらの表に対する INSERT 特権を保持していることが必要です。

表 54. 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、照会コンパイラー、db2advis コマンド、またはユーザーによって入力されます。この表は、次の場合に使用します。 • 推奨索引を入手する • 提案された索引についての入力に基づき索引を評価する
ADVISE_INSTANCE	db2advis の実行に関する情報が入ります。これには開始時刻に関する情報も含まれます。 db2advis の実行ごとに 1 行が入ります。
ADVISE_MQT	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 の列統計情報 (COLSTATS (XML 形式)、NUMROWS など) に加え、各 MQT の詳細な統計を取得するためのサンプリング照会が入ります。
ADVISE_PARTITION	db2advis 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) を保管します。
ADVISE_WORKLOAD	表の各行はワークロードにおける 1 つの SQL または XQuery ステートメントです。db2advis 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	個々の演算子に固有の特性に関する情報がある場合、それを示します。

表 54. Explain 表のサマリー (続き)

表名	説明
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 表で記録された特定の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。
EXPLAIN_INSTANCE	すべての Explain 情報用の主コントロール表。Explain 表の各行は、この表内のある固有の 1 行に明示的にリンクされます。 Explain 対象の SQL または XQuery ステートメントのソースに関する基本情報、および環境情報は、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 情報に関する SQL または XQuery ステートメントのテキストが含まれます。この表には、ユーザーが入力した元の SQL または XQuery ステートメントと、アクセス・プランを選択するのにオプティマイザーで使用されるバージョンとが保管されます。 Explain スナップショットが要求されると、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、Visual Explain が求める書式で EXPLAIN_STATEMENT 表の SNAPSHOT 列に保管されます。この書式は他のアプリケーションでは使用できません。
EXPLAIN_STREAM	個々の演算子とデータ・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。データ・オブジェクト自体は、EXPLAIN_OBJECT 表に示されています。データ・ストリームに関連する演算子は、EXPLAIN_OPERATOR 表にあります。

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ー・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、オプ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とき、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。

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用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ー (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。

$C*n* 派生列の名前。*n* は整数値を表します。

$CONSTRAINT$
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。$WITH_CONTEXT$ 接頭部と組み合わせて表示されます。

$DERIVED.T*n*
派生表の名前。 *n* は整数値を示します。

$INTERNAL_FUNC$
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。

$TRIGGER$
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。$WITH_CONTEXT$ 接頭部と組み合わせて表示されます。

$WITH_CONTEXT$(...)
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。

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・コンセントレ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は実質的に異なる場合があります。Explain 表、SET CURRENT EXPLAIN MODE ステートメント、および Visual Explain 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) は非常に便利です。 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 (*tn*) 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いことを示しています。

    ```
    Relation Scan
    |  Prefetch: None
    ```
  - 次のステートメントは、プリフェッチされるページ数をオプティマイザーが事前に判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、索引ページをプリフェッチすることを示します。

```
Index Prefetch: Eligible
```

– 次のステートメントは、データ・ページのプリフェッチが行われないことを示しています。

```
Data Prefetch: None
```

– 次のステートメントは、データ・ページをプリフェッチすることを示します。

```
Data Prefetch: Eligible
```

– 索引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・インテント・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 Predicates(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ー・プール以外にキャッシュされることを示します。

```
Keep Rows In Private Memory
```

- 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す。

  ```
  Sort Limited To Estimated Row Count
  ```

- 対称マルチプロセッサー (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ー定義関数 (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。特定の名前は、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                       Not Deterministic
|  Called on NULL Input         Disallow Parallel
|  Not Federated                Not Threadsafe
```

結合の情報:

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。

結合が実行されると、以下のステートメントのいずれかが必ず表示されます。

```
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もの特性があり、最初のデータ・ストリーム・ステートメントの後には 1 つまたは複数のステートメントに続けて、それらの特性を記述することができます。

- データ・ストリームの操作がアクセス・プランで以前に生成された値によって決まる場合、そのデータ・ストリームには、以下のマークが付けられます。

  ```
  Correlated
  ```

- ソート済みの一時表と同様、以下のステートメントは、データ・ストリームの結果をメモリーに保持するかどうかを示します。

  ```
  Piped
  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 db2expln 出力に表示されます。

```
Aggregation
Predicate Aggregation
Partial Aggregation
Partial Predicate Aggregation
Intermediate Aggregation
Intermediate Predicate Aggregation
Final Aggregation
Final Predicate Aggregation
```

述部集約とは、データにアクセスした際に、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
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、db2expln 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ときに判別された、ある 1 つのデータベース・パーティションだけがサブセクションを受け取ることを示します。

  ```
  Directed to Any Node
  ```

• パーティション・データベース環境内のサブセクション同士の間、または対称マルチプロセッサー (SMP) 環境にあるサブエージェント同士の間でデータを移動するために、表キューが使用されます。

- 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す。

```
#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。

db2expln 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:

db2expln 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。
- 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使用します。処置に何も示されていない場合は、条件が真であるときにのみデータ操作命令が処理されます。

- ALL、ANY、または EXISTS 副照会がアクセス・プラン内で処理中である場合には、以下のステートメントのうちのいずれかが表示されます。

```
ANY/ALL Subquery
EXISTS Subquery
EXISTS SINGLE Subquery
```

- 特定の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す。

```
Return Data to Application
|  #Columns = n
```

操作が表アクセスにプッシュダウンされた場合、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
```

- ストアード・プロシージャー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。最初の 100 000 リテラルのみが置き換えられ、残りのリテラルはリテラルのままになります。

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 firstnme, lastname from employee where empno='000020'
select firstnme, 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 firstnme, lastname from employee where empno=:L0000000000
```

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、`:L0000000000` 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。

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、準備ができたら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 プロシージャーは、SQL プロシージャー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。このクラスの特性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、次の特性があります。

- 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、次の特性があります。

- 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貪欲型結合列挙を使用する点を除けば、クラス 5 と同様な働きをします。このクラスは、貪欲型結合列挙アルゴリズムを使用するクラスの中では最も高度な最適化であり、複雑な照会の場合にあまり代替プランを考慮しないので、クラス 3 以上と比べてコンパイル時間は少なくて済みます。意思決定支援またはオンライン分析処理 (OLAP) 環境において非常に複雑な照会を行う場合には、クラス 2 をお勧めします。このような環境では、特定の照会が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。このクラスの特性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の経路指定を含めた広範囲のアクセス方式が考慮されます。

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。縮小のエクステントは、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貪欲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、貪欲型結合列挙アルゴリズムを使用します。
- 同じ列上にたくさんの結合述部を持つ表が多数ある場合で、コンパイル時間に制約があるという場合には、最適化クラス 1 または 2 を使用します。
- 実行時間が 1 秒未満の非常に短い照会には、低い最適化クラス (0 または 1) を使用します。この種の照会には、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。

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になります。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。

照会最適化クラスを指定するには、以下のステップに従ってください。

1. 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す。

- 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、db2advis、runstats、db2expln、および最適化クラス設定などのユーティリティーを使用して、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(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))
```

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(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 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、通常の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・ガイドラインを含めることができます。これは、プロファイルが有効であ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るものです。また、最適化プロファイルには、パッケージ内の単一 DML ステートメントに適用される特定のガイドラインを含めることもできます。

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指定のステートメントを見つけるときに、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には、2 つのタイプのガイドラインを指定できる 2 つの主なセクションがあります。つまり、グローバル最適化ガイドライン・セクションには 1 つの OPTGUIDELINES エレメントを含められ、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可能です。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、DB2 バージョン 9.1 に有効な最適化プロファイルの例を示しています。これには、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 VERSION="9.1.0.0">

  <!--
       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 エレメントによって表されます。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すべてのステートメントを処理する際には MQT Test.AvgSales および Test.SumSales を考慮すべきことが指定されています。

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 エレメントによって表されます。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。

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収めた XML 文書です。

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。

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新規の 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記述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 VERSION="9.1.0.0">
   </OPTPROFILE>
   ```

5. OPTPROFILE エレメント内に、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。

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。

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、以下のようにします。

1. 最適化プロファイル表を作成します。 表の各行には最適化プロファイルを 1 つ含めることができます。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、オプ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 フィールドで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ているステートメントと正確に一致しなければなりません。それにより、DB2 はターゲット・ステートメントを明確に識別できます。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>
```

オプ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 特殊レジスターの値は、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。

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 特殊レジスター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。

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、以下のようにします。

- 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する場合、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する場合、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、構成アシスタントまたは 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 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 オプションを使用して、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。

API (例えば sqlaprep) またはコマンド行プロセッサー (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、QUALIFIER オプションが暗黙の修飾子として使用されます。

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には、その文書を編集し、それを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して、SYSTOOLS.OPT_PROFILE 表内のオリジナル文書を新しいバージョンに置き換えます。

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。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 ステートメントを使用することにより、古いプロファイルを最適化プロファイル・キャッシュから除去し、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*)。最適化プロファイルを変更するには、以下のようにします。

1. 最適化プロファイルを編集し、必要な変更を加え、XML を妥当性検査します。
2. SYSTOOLS.OPT_PROFILE 表を新規プロファイルで更新します。
3. 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除去します。

   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ます。

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。また、古い最適化プロファイルで準備されたステートメントは論理的に無効化されるため、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。

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除去します。

   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ます。

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ので、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ので、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 つのカテゴリーに分けることができます。ステートメント・レベルのカテゴリーは、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ーは、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、特定の述部にのみ適用されます。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_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.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、オプティマイザーは 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ー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、オプ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、オプ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、オプティマイザーが 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、オプ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に指定します。INDEX 属性がないため、オプ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、オプティマイザーが 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、オプティマイザーが 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、オプティマイザーが、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は、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。

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。

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。

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、以下のようにします。

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の作成を実行します。
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
```

4. 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。例えば、

```
<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>
```

5. 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>
```

6. ファイルを妥当性検査して保管します。

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になります。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と、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ず、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す。

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、オプティマイザーは、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せん。

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。

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。

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>
      <!-- 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: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 エレメント用の XML スキーマ:

OPTPROFILE エレメントは、最適化プロファイルのルートです。

エレメントは、次のように定義します。

XML Schema

```
   <xs:element name="OPTPROFILE">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="OPTGUIDELINES" type="globalOptimizationGuidelinesType"
              minOccurs="0"/>
            <xs:element name="STMTPROFILE" type="statementProfileType" minOccurs="0"
              maxOccurs="unbounded"/>
         </xs:sequence>
```

```
        <xs:attribute name="VERSION" use="optional">
           <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                 <xs:enumeration value="9.7.0.0"/>
              </xs:restriction>
           </xs:simpleType>
        </xs:attribute>
     </xs:complexType>
  </xs:element>
```

説明

オプションの OPTGUIDELINES サブエレメントは、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。各 STMTPROFILE サブエレメントは、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検証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 エレメントの 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、computationalPartitionGroupChoices、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、オプティマイザー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="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・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。オプティマイザーはステートメント・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。これにより、アプリケーションを編集しなくても、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・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。

STMTKEY エレメントの XML スキーマ:

STMTKEY エレメントを使用すると、オプティマイザーは、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、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。

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・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するには、ステートメント・キーを使用します。

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ー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 コンパイラー・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。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は、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。

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。

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、オプ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、オプティマイザーの検索スペースを変更するために使用できます。

```
<!--****************************************************************************** --> \
<!-- 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:sequence>
</xs:group>
```

注: 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を使用して、一般最適化ガイドラインを定義することができます。このガイドラインは、最適化検索スペースに作用するため、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作用します。

DEGREE 要求:

DEGREE 一般要求エレメントを使用して、DEGREE バインド・オプションの設定、**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、オプ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、オプティマイザーは、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、オプ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・オプションの設定、**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・オプション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・マーカー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・マーカー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・マーカー値ごとに、ステートメントを最適化することを指定します。

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: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

  上記のエレメントは、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。これらのエレメントは、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エレメントは、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 エレメントを使用する必要があります。XML 照会の場合、このエレメントは、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 エレメントに共通する局面は、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。コンパイラーが共用スキャン、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、共用スキャン、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。共用スキャンは、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、その共用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、既にバッファー・プール内にあるページを活用できます。共用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。オプティマイザーは、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。オプティマイザーは、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、オプティマイザーがこの索引を有用と見なすなら、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、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 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、オプティマイザーはコスト・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、個々の索引スキャ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。オプティマイザーは、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、オプ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、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、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。新規のエレメント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
    XMLEXISTS('$SDOC/Security/SecurityInformation/
      StockInformation[Industry = "Software"]') AND
    XMLEXISTS('$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: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

  オプティマイザー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.ps_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 結合要求エレメントを使用して、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 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 プロシージャーを呼び出します。

```
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 プロシージャー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に、結合される両方の表にあるデータのコロケーションによって、データをあるデータベース・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 操作を定期的に実行してください。

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ー (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して、DB2 データ・サーバーによる runstats 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、オプ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消費されます。 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。また、表のために統計プロファイルが提供されていない場合に、自動統計収集によってこれらのデフォルト設定が使用されます。

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。

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))
```

オプティマイザーは、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、オプ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、オプティマイザーは、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、オプティマイザーは、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、オプティマイザーは、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、確かに独立していません。なぜなら、各部門は一度に 1 人の管理者によって普通管理されるからです。オプティマイザーは、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。

単純な等価述部の相関

結合述部相関に加えて、オプ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、すなわちカーディナリティー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 ユーティリティー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。

より洗練された統計は、より複雑な関係を表す際に必要となります。複雑な関係には、式を含む比較 (例: price > MSRP + Dealer_markup)、複数の表にまたがる関係 (例: product.name = 'Alloy wheels' and product.key = sales.product_key)、または独立した属性および単純な比較操作を行う述部以外のものなどがあります。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 によく似ていますが、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ずっと多かった可能性もあります。

現時点では、自動統計収集を統計ビューで使用できません。こうしたビューに関する統計を、統計ビューによって参照される基本表が大幅に更新されたときはいつでも収集してください。

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。

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* といいます。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。

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、オプ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

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に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 55. STORE (63 行)

列	storekey	store_number	city	state	district	...
属性	integer 非 NULL 主キー	char(2)	char(20)	char(5)	char(14)	...

表 56. 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 57. 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 58. PROMOTION (35 行)

列	promokey	promotype	promodesc	promovalue	...
属性	integer 非 NULL 主キー	integer	char(30)	decimal(5)	...

表 59. PERIOD (2922 行)

列	perkey	calendar_date	month	period	year	...
属性	integer 非 NULL 主キー	date	char(3)	smallint	smallint	...

表 60. 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するとします。 表 61 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 61. 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%

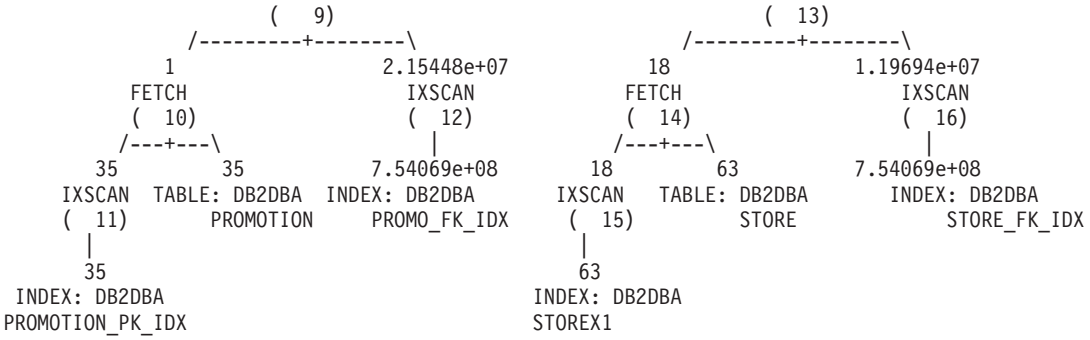
この表は、登録済みのお客様の割引が、提供された 35 種類のプロモーションのわずか 2.86% に相当することを示しています。

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、オプ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。

```
                6.15567e+06
                   IXAND
                   (   8)
     /-----------------+-----------------\
2.15448e+07                          2.15448e+08
  NLJOIN                               NLJOIN
```



ネストされたループ結合 (番号 9) において、オプティマイザーは、販売商品の約 2.86% が割引価格で同じ商品を買いにお客様が戻ってきた結果であると推定します (2.15448e+07 ÷7.54069e+08 ≈ 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 62 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 62. 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。

	結合前		結合後	
述部	**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、オプティマイザーは 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 ÷ 6 155 670 ≈ 2.09)。

オプティマイザーが、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 62 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。

オプ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、オプティマイザーは、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。

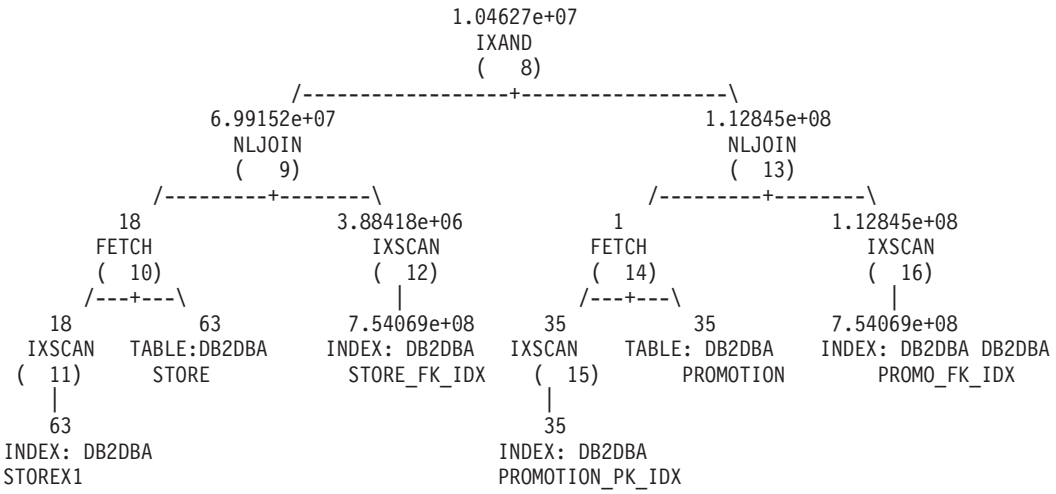


表 63 は、半結合ごとの結合の前後のカーディナリティー推定値とその割合 (フィルタリング効果) を要約します。

表 63. 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。

	結合前		**結合後** (*統計ビューなし*)		**結合後** (*統計ビューあり*)	
述部	**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 400 ページの表 62の統計ビューなしの推定に比べて大きな改善です。

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更新されます。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。

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
- マルチディメンション・クラスタリング (MDC) 表の場合、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・カウントの合計

- 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・ページの数の比率
- 索引の最初の列の固有値の数
- 索引の最初、2 番目、3 番目、4 番目の列の固有値の数
- 索引のすべての列の固有値の数
- 索引キーの順序で、間をあまり空けずにディスクに位置づけられたリーフ・ページの数
- 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ー・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。

パーティション化索引の場合、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。

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 オプション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 オプションが指定された場合に、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 64. 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 65. 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)
SUB_COUNT	サブエレメントの平均数	はい	いいえ (注 2)
SUB_DELIM_LENGTH	サブエレメントを分ける各区切り文字の平均長	はい	いいえ (注 2)

表 65. 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
注: 1. 列統計は、索引キーの中の最初の列に関して収集されます。 2. これらの統計では、ブランクで区切られている一連のサブフィールドまたはサブエレメントを含む列の、データに関する情報が提供されています。 SUB_COUNT および SUB_DELIM_LENGTH 統計が収集されるのは、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 66. 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	列グループのカーディナリティー	はい	いいえ

表 67. 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 68. 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 68. 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 69. 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 69. 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	異なるバッファー・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ー・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 69. 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 69. 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: 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。 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。 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。 DENSITY は、索引ページがプリフェッチされたときに、他の索引から不適切なページが平均してどのくらい読み取られたのかについて、オプティマイザーが見積もるのに使用されます。 4. この統計は、表が DMS 表スペースにある場合は計算できません。 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。 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。			

表 70. 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 70. 列分散統計 (SYSCAT.COLDIST および SYSSTAT.COLDIST) (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 ユーティリティー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。新しいデータベースの作成時に、自動バックグラウンド統計収集はデフォルトで有効になります。自動リアルタイム統計収集は、**auto_stmt_stats** データベース構成パラメーターの値を ON に設定すると有効になります。このパラメーターは、**auto_runstats** 構成パラメーターの子です。

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 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、統計は実際には派生させられず、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。リアルタイム統計収集が使用可能でないと、2 時間間隔で非同期統計収集が行われます。一部のアプリケーションでは、正確な統計を出すためには、この間隔では不十分な場合があります。

リアルタイム統計収集が使用可能な場合にも、非同期統計収集の検査は引き続き 2 時間間隔で行われます。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ます。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。これにより、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ますし、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最適な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最適な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。(非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自動的に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。

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。

- 統計ビュー
- 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)
- 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、統計ファブリケーションは行われません。

- 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いくらかの統計ファブリケーションが行われます。

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ーは行われません。

リアルタイム統計が有効になったなら、定義済みの保守時間枠をスケジュールしなければなりません。デフォルトでは、保守時間枠は定義されていません。保守時間枠が定義されていないと、同期統計収集だけが行われます。その場合、収集される統計はメモリー内に限定され、通常はサンプリング (小さな表の場合を除く) を使用して収集されます。

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。

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。

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 71. 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
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、この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。カタログ・キャッシュは同じ SYSTABLES オブジェクトの複数の項目を格納できます。これにより、すべてのデータベース・パーティションのカタログ・キャッシュが増加します。リアルタイム統計収集が有効な場合は、**catalogcache_sz** データベース構成パラメーター値を大きくすることを考慮してください。

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

構成パラメーターを 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もの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。

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。このことは、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、当てはまることがあります。

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。

単一データベース・パーティションがシングル・プロセッサー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。

自動統計収集を有効にするには、以下のようにします。

1. データベース・インスタンスを構成するために、**auto_maint**、**auto_tbl_maint**、および **auto_runstats** データベース構成パラメーターを ON に設定します。**auto_maint** パラメーターは、**auto_tbl_maint** および **auto_runstats** の親です。
2. オプション: 自動統計プロファイルの生成を有効にするには、**auto_stats_prof** および **auto_prof_upd** データベース構成パラメーターを ON に設定します。**auto_maint** パラメーターは、**auto_stats_prof** および **auto_prof_upd** の親です。
3. オプション: リアルタイム統計収集を有効にするには、**auto_stmt_stats** 構成パラメーターを ON に設定します。 照会の最適化のために必要とされるたびに、表統計が、ステートメントのコンパイル時に自動的に収集されます。 **auto_maint** パラメーターは、**auto_stmt_stats** の親です。

統計プロファイルを使用した統計の収集:

runstats ユーティリティーには、統計プロファイルを登録して使用するオプションがあります。このプロファイルは、特定の表にどの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 オプション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 オプション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 オプション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、データベース活動に関する情報が収集され、照会フィードバック・ウェアハウスに保管されます。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が不確かさが軽減されます。

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・データの収集を活動化し、**auto_prof_upd** 構成パラメーターは統計プロファイルの生成を活動化し、自動統計収集で使用できるようにします。

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、および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ていません。

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 プロシージャーを使用して、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ーについて詳しくは、『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 1 KB 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ーのロギング:

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ー (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 72. 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)	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ー (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。 統計収集アクティビティーに可能な値は、以下のとおりです。 **TABLE STATS** 表統計が収集されます。 **INDEX STATS** 索引統計が収集されます。 **TABLE AND INDEX STATS** 表統計と索引統計の両方が収集されます。 自動統計収集に可能な値は、以下のとおりです。 **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。 **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。 **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。 **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。
OBJNAME	VARCHAR(255)	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。
OBJNAME_QUALIFIER	VARCHAR(255)	統計ロギングの場合、これは表または索引のスキーマです。

表 72. 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
EVENTTYPE	VARCHAR(24)	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。統計ロギングに可能な値は、以下のとおりです。 **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。 **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。 **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。 **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。 **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。
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。統計ロギングの場合、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。統計ロギングの場合、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。統計ロギングの場合、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 72. 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)	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。 **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。 **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。 **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。 **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。 **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。 **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。 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。
THIRD_EVENTQUALIFIERTYPE	VARCHAR(64)	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。

表 72. 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)	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ー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。 **Timeout** 同期統計収集の予定時間を超過しました。 **Error** エラーが原因で統計アクティビティーが失敗しました。 **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。 **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。 • 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 **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。 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。
EVENTSTATE	VARCHAR(255)	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。 • 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              OBJSCHEMA 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 sl
```

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 sl_ix1 on db2user.stats_log(eventtype, event1);
create index sl_ix2 on db2user.stats_log(objtype, event1);
create index sl_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 ユーティリティーは、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 ユーティリティーを使用して統計を収集します。

- 表にデータがロードされ、適切な索引が作成された後
- 表に新規の索引を作成した後
- 表が REORG ユーティリティー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とき、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 を実行した後には、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 ユーティリティーを使用します。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ー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。
- 詳細な索引統計のために実行されるバックグラウンド計算量を減らすために、SAMPLE DETAILED 節を指定して詳細な索引統計を収集します。 SAMPLE DETAILED 節を指定すると、統計を収集するのに必要な時間が短縮され、ほとんどの場合に十分な精度が得られます。
- データが入っている表の索引を作成する場合は、索引の作成時に統計が作成されるように、COLLECT STATISTICS 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 を再度使用して統計を更新します。
- 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。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. 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。
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。

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。

表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。

表データの無作為標本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ー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 ユーティリティー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、各表ページを完全表スキャンに取り込むため、I/O コストは減りません。しかしながら、I/O の量が減ることはなくても、統計収集はプロセッサーでの処理を主としたものなので、行レベルのサンプリングでもパフォーマンスはかなり向上します。

データ値が高クラスター化されている状況では、ページ・レベル・サンプリングよりも行レベル・サンプリングのほうが良いサンプルを提供できます。ページ・レベル・サンプリングと比べると、行レベルのサンプルでは各データ・ページから *P* パーセントの行をサンプリングするので、全データの統計的傾向をより良く反映することができます。ページ・レベルのサンプリングでは、*P* パーセントのページに含まれる行すべてがサンプリング対象になります。行が表の中でランダムに分散している場合は、行サンプル統計とページ・サンプル統計の正確さはほぼ同じです。

RUNSTATS コマンドに (前のサンプルが再生成される) REPEATABLE オプションを指定せずに何回も呼び出すと、各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。

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 オプションを使用して、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 ユーティリティーで 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。

DB2_LIKE_VARCHAR レジストリー変数は、次の形式の述部をオプティマイザーが処理する方法に影響します。*<column* > like '%*<character-string*>%'。このレジストリー変数についての詳細は、『照会コンパイラーの変数』を参照してください。

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 ユーティリティー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うため)。

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。

表が作成されるとき、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 ユーティリティー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際には、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 73 に示します。

表 73. 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際には、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、OVERFLOW だけです。マルチディメンション・クラスタリング (MDC) 表の場合は ACTIVE_BLOCKS も含まれます。
- CARD は、それに対応する SYSSTAT.COLUMNS 内の表のすべての COLCARD 値以上でなければなりません。
- CARD は、NPAGES より大きくなければなりません。
- FPAGES は、NPAGES より大きくなければなりません。
- NPAGES は、(この統計が索引に関連している場合) どの索引の PAGE_FETCH_PAIRS 列内のどの「フェッチ」値よりも、小さいか等しくなければなりません。
- CARD は、(この統計が索引に関連している場合) どの索引の PAGE_FETCH_PAIRS 列内のどの「フェッチ」値よりも、大きくなければなりません。

フェデレーテッド・データベース・システムでは、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。ニックネームが戻す行数などの統計情報は、このリモート・ビューの評価にかかる実際のコストを反映していないことがあるので、DB2 オプティマイザーが正しく動作しないことがあります。ただし、統計の更新がリモート・ビューで役立つ場合もあります。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上にリモート・ビューが定義されている場合などです。複合ビューでは、照会ごとに調整が必要な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 オプションを指定して索引に 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、SAMPLED DETAILED オプションは、統計ヒープとして 2 MB 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。

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。

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 オプションでは、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'
  ```

  説明

  ```
  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し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値というように、**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 74. 列におけるデータ値の頻度

データ値	頻度
20	5
30	10
40	10

表 74. 列におけるデータ値の頻度 (続き)

データ値	頻度
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。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のアクセス・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。具体的には、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ように見積もられます。

```
   KEY2 - KEY1
------------------  x CARD
HIGH2KEY - LOW2KEY
```

以上の見積もりが正確な見積もりとなるのは、列内のデータ値の真の分散が、十分に均一である場合だけです。使用できる分散統計がなく、データ値の頻度が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、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。

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。

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 オプションを指定した 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す。

```
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。

```
CARD - NUM_FREQ_ROWS
--------------------
    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。

```
見積もられた行数 - 実際の行数
----------------------------  X 100
         実際の行数

10 - 2
------  X 100 = 400%
   2
```

頻出値統計 (*N* = 1) を使用すると、オプティマイザーは、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。

```
(50 - 40)
--------- = 3
 (5 - 1)
```

また、エラー率は、次のように 1 桁減ります。

```
 3 - 2
 ----- = 50%
   2
```

変位値統計の例

以下の変位値統計の説明では、「K 変位値」という用語が使用されます。列の *K* 変位値 とは、さまざまなデータ値のうち、データ値が *V* 以下である行が *K* 個以上あるような最小のデータ値 *V* のことです。K 変位値を計算するには、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、オプティマイザーが範囲述部を満たす行の数をより正確に見積もることができます。以下の値を含む列 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 <= 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。

  ```
     KEY2 - KEY1
  ------------------  X CARD
  HIGH2KEY - LOW2KEY
  ```

  KEY1 を LOW2KEY に置き換えると、述部を満たす行の数は、次のように見積もられます。

  ```
   8.5 - 5.1
  ----------  X 10 ≈ 0
  93.6 - 5.1
  ```

  ≈ は「ほぼ等しい」という意味です。この見積もりのエラーは、約 -100% です。

  変位値統計を使用できる場合、オプティマイザーは、 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 <= 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 <= 8.5` を満たす行の数、`r_2` は、述部 `C > 8.5 かつ C <= 10` を満たす行の数です。前述の例のとおり、`r_1 = 7` です。 `r_2` を見積もるため、オプティマイザーは線形補間を使用します。

  ```
         10 - 8.5
  r_2 ≈ ----------  X (8.5 より大きく 100.0 以下である値を持つ行の数)
         100 - 8.5

         10 - 8.5
  r_2 ≈ ----------  X (10 - 7)
         100 - 8.5

         1.5
  ```

```
r_2 ≈ -----   X (3)
      91.5

r_2 ≈ 0
```

最終的な見積もりは、r_1 + r_2 ≈ 7 であり、エラーは -12.5% のみです。

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。

```
30 - 20
-------  X 100 = 25
70 - 30
```

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。

```
100 - 80
-------- = 5
  7 - 3
```

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。

```
  30 - 20
  -------  X (20 より大きく 40 以下である値を持つ行の数)
  40 - 20

  30 - 20
= -------  X (25 - 5)
  40 - 20

= 10
```

最終見積もりは 15 になり、エラーは 3 分の 1 になりました。

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。 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。

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* x *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 ユーティリティーでは、 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 75 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 75. 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
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ーによって、またはオペレーティング・システムで使用可能なモニター・ツールによって提供される出力を使用します。

モデル化および what-if の計画のカタログ統計

計画に役立てるため、システム・カタログ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・カタログ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'
```

カタログ統計を手動で更新するときは慎重に行ってください。軽率な変更により、以降の照会のパフォーマンスに対して重大な影響が及ぶ可能性があります。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 ユーティリティーを使用してカタログ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ーの数も実動システムと同じでなければなりません。

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が、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サンプリングを使用して (TABLESAMPLE SYSTEM または 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)` 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。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る、またはディスクに書き込む必要のあるデータ量が減少し、結果として入出力コストを削減できます。

現在使用できるデータ圧縮の形式には、以下の 2 通りあります。

- *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* では、1 つの行の複数の列値に渡る反復パターンをより短いシンボル・ストリングで置き換えます。行圧縮論理は、反復データと重複データを圧縮する表をスキャンします。コンプレッション・ディクショナリーにはそのデータへの短い数字キーが含まれ、圧縮行ではこうしたキーが実際のデータと置き換えられます。

DB2 バージョン 9.1 より前は、オフライン表再編成を実行して、コンプレッション・ディクショナリーを手動で作成する必要がありました。バージョン 9.5 以降では、データ圧縮が可能な表に関してはデータ・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ます。

今回のリリースでは、バージョン 9.5 で永続表に導入されたオートノミック行圧縮が拡張され、すべての一時表が組み込まれるようになりました。一時表のデータ圧縮では、以下のようになります。

- 大規模で複雑な照会で必要な一時ディスク・スペース量が削減されます。
- 照会パフォーマンスが向上します。

DB2 Storage Optimization Feature では、一時表のデータ圧縮が自動的に使用可能になります。

行圧縮に適格なそれぞれの一時表では、コンプレッション・ディクショナリーを作成するために 2 から 3 MB のメモリーがさらに必要となります。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。

また圧縮された一時表上にある索引オブジェクトと索引も圧縮が可能で、ストレージ・コストを削減できます。これは、通常はとても大きな索引を多数使用する、大規模なオンライン・トランザクション処理 (OLTP) およびデータウェアハウス環境

で特に役立ちます。どちらの場合も、索引圧縮を行うと入出力制約環境ではかなりのパフォーマンスの改善になり、CPU 制約環境ではパフォーマンスが減少することはほとんどまたは全くありません。

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。これは、XML 列が DB2 製品の現行バージョンに追加された表にも適用されます。XDA 圧縮は、XML 列がこのバージョンより前に作成された表ではサポートされていません。そのような表の場合、データ・オブジェクトのみが圧縮されます。

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・バッファーが満杯になるとき

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、非常に困難なことがあります。設計アドバイザーは、ワークロードのパフォーマンスを改善するのに必要なすべてのオブジェクトを識別します。

設計アドバイザーは、ワークロードの SQL ステートメントのセットを考慮に入れ、以下の推奨を生成します。

- 新規索引
- 新規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・ユーティリティーを起動するには、db2advis 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

- アクティビティー・モニター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
- 共用メモリー・ユーティリティーに関するヘルス・センターからの推奨、またはソートを集中的に行うワークロードのソート・ヒープの問題に関するヘルス・センターからの推奨に応じた処置をする
- 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

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、db2advis コマンド上で -m オプションが使用されたかどうかを示します。
  - COMPRESSION フィールドは、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・プロシージャー呼び出しが ADVISE_TABLE 表の ALTER_COMMAND 列に置かれます。 ALTER TABLE ストアード・プロシージャー呼び出しは、ALTOBJ ストアード・プロシージャー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 索引を非クラスター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>
```

db2advis コマンドで -o オプションを使用すると、設計アドバイザーの推奨をファイルに保存できます。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)、ならびに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、それらは出力に表示されます。

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

db2advis 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。

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 db2advis 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. db2advis 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。

ワークロード とは、データベース・マネージャーが所定の期間に処理しなければならない一連の SQL ステートメントのことです。設計アドバイザーの実行対象は以下のとおりです。

- db2advis 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

新規のワークロード・ファイルを作成するか、または既存のワークロード・ファイルを変更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
- Query Patroller 履歴データ表 (コマンド行から -qp オプションを使用する)
- 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。

- 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、以下のようにします。
  1. 次のコマンドを実行して、データベース・モニターをリセットします。

```
db2 reset monitor for database database-name
```

2. データベースに対して動的 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 オプションを使用して、db2advis 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 オプションも使用します。

- 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、以下のようにします。
  1. 手動でワークロード・ファイルを作成するか (各 SQL ステートメントをセミコロンで区切る)、または前述のソース (複数可) から SQL ステートメントをインポートします。
  2. ワークロード内のステートメントの頻度を設定します。ワークロード・ファイル内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。ワークロード内にある特定のステートメントの頻度または重みを手動で変更することも可能で、そのステートメントの次の行に `--#SET FREQUENCY n` を挿入します。ここで、*n* はステートメントに割り当てる頻度値です。
  3. -i オプションとワークロード・ファイルの名前を使用して、db2advis 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 オプションとワークロード名を指定して db2advis 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、データベース・パーティション・フィーチャー (DPF) 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す。

3. db2advis 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. db2advis 出力ファイルを若干変更してから、設計アドバイザーによって生成された DDL ステートメントを実行します。 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化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 db2advis 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。
- db2advis 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 db2advis 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

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よいかが分かりません。

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との間には、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ジ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、または分析のために以下のツールを使用できます。

- db2dart

  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。

- db2diag

  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で必要なレコードを見つけるための時間を短縮できます。

- db2greg

  グローバル・レジストリーは、db2greg ツールを使用して表示および編集できます。

- db2level

  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。

- db2look

  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、実動システムに悪影響を与えないようにそのテスト・システムでテストする方が理にかなっています。 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。

- db2ls

  システムに DB2 製品の複数のコピーをインストールできる能力と、選択したパスに DB2 製品およびフィーチャーをインストールできる柔軟性があるため、インストールされている内容とその場所を把握しておくのに役立つツールが必要です。サポートされている Linux および UNIX オペレーティング・システムにおいて、 db2ls コマンドは、システムにインストールされている DB2 製品およびフィーチャー (DB2 バージョン 9 HTML 資料が含まれます) をリストします。

- db2pd

db2pd ツールは、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。

- db2support

  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 ユーティリティーは、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。

- db2val

  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。

- トレース

  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。通常の状況では、IBM ソフトウェア・サポートに依頼された場合にのみ、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。

- プラットフォーム固有のツール (Windows) (Linux および UNIX)

  Windows、Linux、および UNIX オペレーティング・システムで提供されている有用な診断ツールを使用して、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ことができます。

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。

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活動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。

データベースを非活動化しなかった場合、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 76. 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	あり	なし
内部ページ・ヘッダー・フィールドの内容の検証	あり	あり
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ト・マップの検査	あり	なし
すべてのエクステント・マップ・ページでの整合性ビット・エラーの検査	なし	あり
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	なし	あり
内部ページ・ヘッダー・フィールドの内容の検証	あり	あり
エクステント・マップが表スペース・マップと一致することの検査	あり	なし

表 77. 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	あり	あり

表 77. 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
特殊制御行の内容の検査	あり	なし
可変長列の長さと位置の検査	あり	なし
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	あり	なし
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	なし	あり
内部ページ・ヘッダー・フィールドの内容の検証	あり	あり
各行レコード・タイプとその長さの検査	あり	あり
行がオーバーラップしていないことの検査	あり	あり

表 78. 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	あり	あり
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	あり	あり
索引におけるキーの順序付けの検査	あり	なし
サマリー合計ページと使用済みページの判別	なし	あり
内部ページ・ヘッダー・フィールドの内容の検証	あり	あり
ユニーク・キーの一意性の検査	あり	なし
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	なし	あり
データ値に照らした各キーの検査	なし	あり

表 79. 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	あり	あり
サマリー合計ページと使用済みページの判別	なし	あり
内部ページ・ヘッダー・フィールドの内容の検証	あり	あり

表 80. 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	あり	あり
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)	なし	あり

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で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, sqleDatabaseQuiesce, 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, sqldPoolCreate, 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、その後にエラー・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)
- OPSTATS (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、以下のようにします。

```
db2diag -facility MAIN
```

OPSTATS 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、以下のようにします。

```
db2diag -fac OPSTATS -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、以下のようにします。

```
db2diag -fac all -g instance=harmistr,level=Error
```

OPSTATS 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、以下のようにします。

```
db2diag -fac opstats -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: 分割された診断ディレクトリー・パス・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

この例では、現行ホスト上の 3 つのデータベース・パーティションからファイルをマージする方法を示します。分割された診断ディレクトリー・パスを取得するために、**diagpath** データベース・マネージャー構成パラメーターが次のように設定されました。

```
db2 update dbm cfg using diagpath '"$n"'
```

以下は、マージする 3 つの db2diag ログ・ファイルのリストです。

- ~/sqllib/db2dump/NODE0000/db2diag.log
- ~/sqllib/db2dump/NODE0001/db2diag.log
- ~/sqllib/db2dump/NODE0002/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
```

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、$HOME/sqllib/global.reg にあります ($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 sqllib/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。

db2greg ツールは、IBM ソフトウェア・サポートからの要請があった場合にのみ使用するべきです。

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 ユーティリティー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、およびバッファー・プールの DDL も作成する場合には、上記のコマンドの -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、バッファー・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。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 データベース製品のリスト表示 (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、db2ls 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 9.7 HTML 文書を含む) がリスト表示されます。

始める前に

db2ls 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 以降の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できるためには、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、db2ls 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。

db2ls 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。 db2ls コマンドはこのどちらかの場所から実行できます。db2ls 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。

db2ls 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

db2ls 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。

- db2ls コマンドを root 権限で実行する場合、ルート DB2 インストールだけが照会されます。
- db2ls コマンドを非ルート (non-root) ID で実行する場合、ルート DB2 インストール、および一致する非ルート ID が所有する非ルート・インストールが照会されます。その他の非ルート (non-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、db2ls 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、変更しなければなりません。

Windowsオペレーティング・システムで db2ls 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。

  ```
  db2ls
  ```

  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。

```
db2ls -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。DB2 バージョン 8 製品を照会すると、空白値が戻されます。
– **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。

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、9.5.0.0。
  – 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、取得対象の情報の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プ
- 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 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     Firstlsn       Lastlsn        LogSpace SpaceReserved TID           AxRegCnt GXID
0x0780000020251B80 11       [000-00011] 2       4     READ  0x00000000 0x00000000 0x000000000000 0x000000000000 0        0             0x0000000000B7 1       0
0x0780000020252900 12       [000-00012] 3       4     WRITE 0x00000000 0x00000000 0x000000FA000C 0x000000FA000C 113      154           0x0000000000B8 1       0
```

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

0x07800000006879E0 12       [000-00012] 1         1073336 UOW-Waiting   0       0        17      1        *LOCAL.burford.060303225602
0x0780000000685E80 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。ApplHandl 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 CoorEDU    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
  ClientAccntng :          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
  ClientAccntng :          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・マネージャー構成パラメーター **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
Datbase: 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            AppHandl [nod-index] NumAgents  CoorPid  Status
0x00000002006D2120 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
0x0780000000740068 62       resynch    2          112         1 1746  1583816485
0x0780000000725688 62       resynch    1          108864      1 127   1599127346
0x07800000001F4348 57       ostrack    6          5160048     1 3047  698130716
0x07800000001B5608 57       ostrack    5          240048      1 3034  698130716
0x07800000001A0068 57       ostrack    1          80          1 2970  698130716
0x07800000001A00E8 57       ostrack    2          240         1 2983  698130716
0x07800000001A0208 57       ostrack    3          80          1 2999  698130716
0x07800000001A0288 57       ostrack    4          80          1 3009  698130716
0x0780000000700068 70       apmh       1          360         1 1024  3878879032
0x07800000007001E8 70       apmh       2          48          1 914   1937674139
0x0780000000700248 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 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です。

```
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   FscrUpdates 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
0x0000000220A08C40 78     1        2       2       3       2       declare global temporary table temp1 (c1 char(6)) not logged
0x0000000220A8D960 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ことにより、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
0x0000000200661840 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回の 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
0x0000000200698080 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 CoorPiid 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   NoChgUpdts Reads   FscrUpdates Inserts ...
0x0780000020CC0D30 TEMP (00001,00002) 0       0     0          0          0       0           43219   ...
0x0780000020CC14B0 TEMP (00001,00003) 0       0     0          0          0       0           42485   ...
0x0780000020CC21B0 TEMP (00001,00004) 0       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 の値の間の関係に注目してください。

```
Tablespace Configuration:
Address            Id    Type Content PageSz ExtentSz Auto Prefetch BufID BufIDDisk FSC NumCntrs MaxStripe LastConsecPg 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  CoorPid    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 LSN (現在の LSN)」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         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、出力は、ロックの数、最初のログ・シーケンス番号 (LSN)、最後の LSN、使用ログ・スペース、および予約済みスペースに関する情報を提供します。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
0x0000000022026D980 797      [000-00797] 2        108    WRITE  0x00000000
0x0000000022026E600 806      [000-00806] 3        157    WRITE  0x00000000
0x0000000022026F280 807      [000-00807] 4        90     WRITE  0x00000000

Tflag2     Firstlsn       Lastlsn        LogSpace   SpaceReserved
0x00000000 0x000001072262 0x0000010B2C8C 4518       95450
0x00000000 0x000001057574 0x0000010B3340 6576       139670
0x00000000 0x00000107CF0C 0x0000010B2FDE 3762       79266

TID            AxRegCnt   GXID
0x000000000451 1          0
0x0000000003E0 1          0
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       State      Size  Pages  Filename
0x0000000023001BF58 0x000001B58000 0x00000000 1000  1000   S0000002.LOG
0x0000000023001BE98 0x000001F40000 0x00000000 1000  1000   S0000003.LOG
0x0000000230008F58 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。

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。

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。

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 ユーティリティーは、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。

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、どのファイルを収集す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。

- 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。

  db2support ユーティリティーが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 (例えばインスタンス所有者) によって実行されなければなりません。SYSADM 権限を持たないユーザーが db2support ユーティリティーを実行した場合、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・サポートに情報を送るために db2support ユーティリティーを使用する場合には、問題がシステムで発生している最中に db2support コマンドを実行してください。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。

  ほとんどの場合、問題をデバッグするためには、以下のような基本的オプションを指定して起動すれば、必要な情報が収集されます (なお、-c オプションを使用すると、ユーティリティーはデータベースへの接続を確立します)。

  ```
  db2support <output path> -d <database name> -c
  ```

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。

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 ユーティリティー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 ユーティリティー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
- バッファー・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
- 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)

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
- CPU 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空きスペース、および UNIX の i ノード
- Java™ SDK のレベル
- データベース・マネージャー構成
- データベース・リカバリー履歴ファイルのリスト
- 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
- すべてのインストール済みソフトウェアのリスト

-s オプションを指定した場合には、以下の情報が db2support.html ファイルに表示されます。

- 詳細なディスク情報 (パーティションのレイアウト、種類、LVM 情報など)
- 詳細なネットワーク情報
- カーネル統計
- ファームウェアのバージョン
- その他のオペレーティング・システム固有のコマンド

DB2 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)
- CLI 構成
- 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

-c オプションを指定し、データベース接続が正常に確立された場合には、以下の情報が db2support.html ファイルに含まれます。

- 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
- バッファー・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

db2support.zip ファイルの内容の例

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・マネージャー、BP、CLI、および Java Developer Kit)
- DB2DUMP/ - db2diag.log ファイルのここ 3 日間の内容
- DB2MISC/ - sqllib 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、および LIST APPLICATIONS) の出力
- db2supp_opt.zip - オプティマイザー問題の診断情報
- db2supp_system.zip - オペレーティング・システム情報
- db2support.html - HTML セクションにフォーマットされた診断情報
- db2support_options.in - db2support 収集を開始するのに使用されるコマンド行のオプション

オプティマイザーに関する情報は、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/ - オプティマイザー問題の診断情報
- OPTIMIZER/optimizer.log - ファイルにすべてのアクティビティー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
  - FUNCTIONS
  - 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 (~/sqllib/cfg のファイル)
- DB2MISC/ - DB2SYSTM 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/ - 重要なバッファー・プール・メタファイル (~/sqllib/sqldbdir)
- 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ーのログ

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。

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、そのデータベースへの接続、および DPF 環境の状態を検証することにより、 DB2 コピーの中核となる機能を検査します。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。通常の状況では、IBM ソフトウェア・サポートに依頼された場合にのみ、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ーを避けてください。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。
- GUI (グラフィカル・ユーザー・インターフェース) ツールのいずれかで問題が発生し、DB2 コマンド・ウィンドウにある明示コマンドを介して実行した際には、その同じアクションが成功した場合、コントロール・センターのトレースが適切です。これによって収集されるのは、コントロール・センターから起動可能なツールに関連した問題のみです。
- 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・バッファー・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オーバーヘッド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 SYSMAINT 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・バッファーのサイズを指定します。バッファー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・バッファー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、バッファー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されます。

バッファーが小さすぎると、情報が失われる可能性があります。デフォルトでは、バッファーがいっぱいになると最も新しいトレース情報のみが保持されます。バッファーが大きすぎると、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・バッファーが必要になります。

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、各プラットフォームでの動作についても、上で説明したとおりです。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ーが使用されます。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ないで、メモリー・バッファーを 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ーを使用します。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ー・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ーを「クリア」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・バッファーをクリアできます。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ます。

```
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・バッファーが小さすぎて、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になります。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・バッファーのサイズを大きくして操作をやり直すことになります。

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・バッファー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、診断ディレクトリーに `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。

受信バッファーが SQLCA 情報を含んでいる場合、その後に、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。送信バッファーと受信バッファーは、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ーには、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。バッファー内の 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。

送信バッファーと受信バッファー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
  - 受信バッファー
- -s
  - DRDA 要求のタイプ
  - 送信バッファー
- -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
  - 機能
  - 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、ファイル上の許可により消去を禁止しているのでない限り、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 番目のバッファーには、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 番目のバッファーには、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。このバッファーには、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 番目のバッファーにあります。 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相関させることができ、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 番目のバッファーの中で 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。

アカウンティング・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。

コード点がそれぞれ X'119D' と X'119E' になっている追加のオブジェクト CCSIDDBC (2 バイト文字の CCSID) と CCSIDMBC (混合バイト文字の CCSID) も、この 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 番目のバッファー内にあるアクセス RDB 応答メッセージ (ACCRDBRM) にも戻されます。このバッファーには、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 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ー (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。

504 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  F2829740404040 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 F261C1C9E7F6F400   ....G....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  404040404040                        @@@@@@

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 F20014116DE2E3D3   ....G.......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......  ..}......>...s..
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  4040404040040000C 11A1D9858799F485   @@@@@@..........        ....Regr4e
  0030  A599000A11A09585 A6A39695           ............      vr....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   ........ ...!.$.  ..}....x........
  0010  00172135C7F9F1C1 F0C4F3C14BD7C1F8   ..!5........K...  ....G91A0D3A.PA8
  0020  F806030221064600 162110E2E3D3C5C3   ....!.F..!......  8..........STLEC
  0030  F140404040404040 4040404040000C11   .@@@@@@@@@@@@...  1            ...
  0040  2EE2D8D3F0F9F0F0 F0000D002FD8E3C4   ............/...  .SQL09000....QTD
  0050  E2D8D3C1E2C30016 00350006119C0333   .........5.....3  SQLASC..........
  0060  0006119D04B00006 119E0333003C2104   ...........3.

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..  ..}....n........
  0010  000D002FD8E3C4E2 D8D3F3F7F0000C11   .../............  ....QTDSQL370...
  0020  2EC4E2D5F0F8F0F1 F500160035000611   ............5...  .DSN08015.......
  0030  9C04B80006119E04 B80006119D04B000   ................  ................
  0040  0C11A0D5C5E6E3D6 D540400006212524   .........@@..!%$  ...NEWTON  .....
  0050  34001E244E000624 4C00010014244D00   4..$N..$L....$M.  ....+...<.....(.
  0060  06244FFFFF000A11 E8091E768301BE00   .$O........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  0053D00510001004D 200A00442113E2E3   .S.Q...M ..D!...  ..}....(......ST
  0010  D3C5C3F140404040 4040404040404040   ....@@@@@@@@@@@@  LEC1
  0020  D5E4D3D3C9C44040 4040404040404040   ......@@@@@@@@@@  NULLID
  0030  4040E2D8D3C3F2C6 F0C1404040404040   @@........@@@@@@    SQLC2F0A
  0040  4040404041414141 41484C5600CB0005   @@@@AAAAAHLV....      ......<.....
  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             s1.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 4F544C2000FFFFFE   2704DSNXOTL ....  ......+.!.<.....
  0020  0C00000000000000 00FFFFFFFF000000   ................  ................
  0030  0000000000572020 2057202020202020   .....W   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ー情報

それ以降の送信バッファーと受信バッファーを分析して、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 番目のバッファーは、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。

504 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ーの例を示しています。

コントロール・センターのトレース

コントロール・センターで問題のトレースを試みる前に、DB2 コマンドから明示的なコマンドを介して同等のアクションを実行したときに同じ問題が生じないか、まず確認しておくようお勧めします。

コントロール・センター内でタスク (またはコントロール・センターから開始できる、他の GUI ツールの 1 つ) を実行すると、しばしば「コマンドの表示」ボタンが表示されます。これを使用すると、ツールが使用するコマンドの構文がそのとおりに示されます。このコマンドを DB2 コマンド・プロンプトからそのまま実行すると成功するのに、GUI ツールから実行すると失敗する場合は、コントロール・センターのトレースを取得するのが適当です。

コントロール・センター内でのみ再現可能な問題のトレースを取得するには、以下のようにしてコントロール・センターを開始します。

```
db2cc -tf filename
```

これによってコントロール・センターのトレース機能がオンになり、指定したファイルにトレースの出力が保管されます。出力ファイルは、Windows では <DB2 インストール・パス>¥sqllib¥tools に、UNIX および Linux では /home/<userid>/sqllib/tools に保管されます。

注: トレース機能を使用可能に設定してコントロール・センターを開始した場合は、できるだけ少ないステップで問題を再現してください。不要または無関係なツールの項目をクリックしないようにしてください。問題を再現したら、コントロール・センターを閉じます (問題を再現するために他の GUI ツールを開いた場合はそれらも閉じます)。

生成されるトレース・ファイルは、分析のために IBM ソフトウェア・サポートに送信する必要があります。

JDBC トレース・ファイル

Linux、UNIX、および Windows 用の DB2 JDBC Type 2 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、Linux、UNIX、および Windows システム用の DB2 JDBC Type 2 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。

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。

- Linux、UNIX、および Windows 用の DB2 JDBC Type 2 ドライバー (DB2 JDBC Type 2 ドライバー) を使用する JDBC アプリケーション
- DB2 JDBC ストアード・プロシージャーの中

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 DB2 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 DB2 サポートの推奨によるものです。 DB2 JDBC Type 2 ドライバーは、内部で DB2 CLI ドライバーを使用してデータベースにアクセスします。例えば、Java の getConnection() メソッドは、内部で DB2 JDBC Type 2 ドライバーによって DB2 CLI SQLConnect() 関数にマップされます。その結果、Java 開発者は DB2 CLI トレースを、DB2 JDBC トレースを補完するものとして活用できます。

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 では次のようにします。

   ```
   mkdir c:¥temp¥trace
   ```

   Linux および UNIX 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。これを行うには、以下の 2 つの方法があります。
   - db2cli.ini ファイルを手動で編集する。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ーのタイプ、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。
     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。
     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]
        JDBCTrace=1
        JDBCTracePathName=<path>
        JDBCTraceFlush=1
        ```

        ここで、<path> は、例えば Windows では C:¥temp¥trace、Linux または UNIX オペレーティング・システムでは /tmp/trace などとなります。
     c. ファイルの最後にブランク行を少なくとも 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。)

- UPDATE CLI CFG コマンドを使用して db2cli.ini ファイルを更新する。次のコマンドを実行します。

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTrace 1
db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTracePathName <path>
```

ここで、<path> は、例えば Windows では C:¥temp¥trace、Linux または UNIX オペレーティング・システムでは /tmp/trace などとなります。

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTraceFlush 1
```

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を及ぼし、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。

3. 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 ファイルはアプリケーション開始時のみに読み取られるため、変更を有効にするには、アプリケーションを再開する必要があります。

JDBC ストアード・プロシージャーをトレースする場合には、DB2 インスタンスの再開が必要です。

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。トレース時に実行している JDBC アプリケーションだけを用いて問題を再現するというような状況は、できるだけ避けてください。そうするなら、トレース・ファイルをより明快なものにできます。
6. JDBC トレースを使用不可にします。

手動で db2cli.ini の [COMMON] セクションに JDBCTrace=0 キーワードを設定するか、次のコマンドを実行します。

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 0
db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTrace 0
```

7. 実行中およびトレース中のアプリケーションを再開します。
8. トレース・ファイルを収集します。

JDBC トレース・ファイルは、JDBCTracePathName キーワードで指定されたパスに書き込まれます。生成されるファイル名すべてには、.trc という拡張子が最後に付きます。問題再現の際にトレース・パスに生成された、すべてのファイルが必要となります。

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。

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 プロパティーを 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/foobar.txt;" ;
  ```

CLI トレース・ファイル

CLI トレースは、DB2 CLI ドライバーにアクセスするアプリケーションに関する情報を取り込みます。CLI トレースは、DB2 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 DB2 CLI インターフェースを使用して DB2 にアクセスした後)
- DB2 CLI ストアード・プロシージャー
- JDBC アプリケーションおよびストアード・プロシージャー

ODBC アプリケーションを診断する際、ODBC トレースまたは DB2 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。DB2 CLI トレースは DB2 に固有のものです。一般的な ODBC トレースより多くの情報が含まれて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。

Linux、UNIX、および Windows 用の DB2 JDBC Type 2 ドライバー (DB2 JDBC Type 2 ドライバー) は、データベースにアクセスするために DB2 CLI ドライバーに依存しています。そのため、Java 開発者は DB2 CLI トレースを使用可能にすることにより、自分のアプリケーションがさまざまなソフトウェア層を介してデータベースと対話する方法に関する追加情報を得ることができます。DB2 JDBC および DB2 CLI トレース・オプションは (どちらも db2cli.ini ファイルで設定されますが)、相互に依存していません。

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ーのタイプ、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. ファイルの最後にブランク行を少なくとも 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・プロシージャーをトレースすると、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。

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、p*pid**tid*.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

通常の関数の場合と同様、DB2 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 DB2 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・マーカーを宣言および使用することにより、DB2 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・マーカーを示します)。

```
 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・マーカーを最大長 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。 rbgValue=″000010″ は、実行時にアプリケーションによってパラメーター・マーカー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"303030303130",
 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

DB2 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、DB2 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、プロセッサー・ティックおよび ISO タイム・スタンプ、プロセッサー・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

DB2 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、DB2 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
```

上記の 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・カーソルが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)。

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。

ワトソン博士
「ワトソン博士」ユーティリティーは、一般保護障害 (GPF) の発生時に呼び出されます。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 ディ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・システムがフル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 アクティビティー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ーの場合)

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係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? まだしていないなら、「*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』を検討してください。
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。この場合は、本書の別の場所にある『インストール問題についてのデータの収集』のトピックを参照してください。
   - インスタンスまたは DB2 管理サーバー (DAS) の作成、ドロップ、更新、またはアップグレード中。この場合は、本書の別の場所にある「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」のトピック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。この場合は、本書の別の場所にある『データ移動問題についてのデータの収集』のトピック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。。

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。

完全な出力を得るには、インスタンス所有者が db2support ユーティリティーを呼び出す必要があります。

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。

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、「*databasename.database_partition_groupname. timestamp*」(Linux および UNIX の場合) および「*databasename.database_partition_groupname.date.time*」(Windows の場合) というファイルを探してください。それは、*$HOME*/sqllib/db2dump ディレクトリーまたは $DB2PATH¥sqllib¥redist にそれぞれ置かれています (*$HOME* はインスタンス所有者のホーム・ディレクトリー)。

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 管理サーバー (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。

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、以下のようにします。

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ので、該当するすべてのファイルを確実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。

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。

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。

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。 .

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。

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、 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、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。確認するには (またはロック待機問題について考慮するには)、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、ロック・エスカレーションの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ときは、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か、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。

手順

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。以下は、ロック待機の 3 つの異なる品質と品質を診断する最適な方法のリストです。

- 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_V97 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。

次の作業

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『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

524 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 プロシージャーを実行して、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あります。
- 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。

次の作業

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。

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、ビクティム・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。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。 .

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。

手順

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。ビクティム・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様ではなく、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・マネージャー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。

次の作業

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

524 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には、DB2

ロック待機およびセマフォーなどの DB2 以外のリソースの待機を伴うデッドロックがある場合があります。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ーを待っている場合があります。

次の作業

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・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。

手順

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にまれなことですが、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、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・マネージャー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。

次の作業

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

524 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あります。

次の作業

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、ロック待機の数が標準より多く、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。 .

手順

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になります。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。

次の作業

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に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

524 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。

手順

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_escals** 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・マネージャー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。

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るからです。
   - 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が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際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。

次の作業

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。

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も判別しています。「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。

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。

手順

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予想されていますが、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. 次のコマンドを実行して、インスタンスおよび中断された EDU を強制的にシャットダウンします。

   ```
   db2_kill
   ```

   注: db2stop コマンドの発行では、インスタンスがトラップを受けた場合は完了しません。
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。

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あります。
     - 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。

次の作業

上記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・サポートを使用して Problem Management Record (PMR) を開くことを考慮してください。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。

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・プロシージャー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下回る一時表に対して行圧縮が使用されないことを除き、システムに対するリスクは存在しません。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ー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 (...)
```

db2ReadLog API を使用するクライアント・ログ・リーダーが、以下に示す最初の INSERT ステートメントに対応する次のログを読み取ります。

```
-> 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

ユーザー応答:

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ないことがあります。

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 許可を持っている限り、問題にはなりません。READ 許可さえあれば、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に起こり得る問題と、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 許可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、いくつかの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、自分が参照できるグローバル変数値と同じグローバル変数を 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 を使用する許可をセキュリティー管理者 (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、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"
```

高可用性のトラブルシューティング

AIX 6.1 では Tivoli System Automation for Multiplatforms (SA MP) Base Component は DB2 バージョン 9.5 GA によってインストールされない

DB2 バージョン 9.5 GA 高可用性機能に含まれている IBM Tivoli® SA MP Base Component は、AIX 6.1 オペレーティング・システムをサポートしていません。AIX 6.1 用の適切なバージョンの SA MP Base Component を取得するには、DB2 バージョン 9.5 フィックスパック 1 以降をインストールしてください。

症状

DB2 バージョン 9.5 GA データベース製品を AIX 6.1 にインストールすると、インストーラーは AIX 6.1 が使用されていることを検出し、SA MP Base Component はインストールされません。

原因

DB2 バージョン 9.5 GA にバンドルされている SA MP Base Component は AIX 6.1 をサポートしていません。

問題の解決

DB2 バージョン 9.5 フィックスパック 1 以降を AIX 6.1 にインストールすると、SA MP Base Component は正常にインストールされます。

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。不整合が見つかれば、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ので、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。

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。

次の作業

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。例えば、以下のようにします。

   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。
   - 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。

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。

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。例えば、以下のようになります。

  ```
  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。

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。

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。

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用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になります。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 に対する書き込み許可がないためです。

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と、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えます。

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、指定する値が既に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。ライセンス交付違反が存在する場合は、適切なライセンス・キーを取得するか、問題のある DB2 データベース製品またはフィーチャーを除去することによって、それらに対処することができます。

始める前に

以下の各ステップでは、ライセンス・センターまたは 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ー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、ライセンス・センターまたは 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。

   一部の DB2 フィーチャーには、ソフト・ストップ・ポリシーがあります。つまり、フィーチャーが違反の場合でも、ライセンス・キーを取得して適用する時間を与えて動作し続けます。他のフィーチャーには、違反の場合フィーチャーが機能しなくなるハード・ストップ・ポリシーがあります。

   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. ライセンスを購入するのではなく、問題のあるオブジェクトのドロップまたは削除を選択する場合は、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')
```

- 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')
```

次の作業

違反に対する対処を完了 (フィーチャーのライセンスを取得するか、または違反の原因を除去) したら、ライセンス・センターから、または次のコマンドを発行して、ライセンス準拠レポートをリセットすることができます。

```
db2licm -x
```

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。

オプ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。オプ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した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 を以下の 559 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_REQUESTOR、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す。

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

問題は、DB2 Enterprise Server Edition と DB2 Database Partitioning Feature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 6.1 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と、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と、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 を使用するので、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。

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 コマンドを使っ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。

レジストリー変数エラーの診断

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。ストレージ保護キーは POWER6™ プロセッサーで使用でき、AIX 5L™ バージョン 5.3 (5300-06 Technology Level を適用) オペレーティング・システムの時点でサポートされています。

第 7 章 DB2 Connectのトラブルシューティング

DB2 Connect 環境には、複数のソフトウェア、ハードウェア、および通信製品が含まれます。トラブルシューティングの最も良い方法は、利用できるデータを排除および限定していき、結論 (エラーの発生箇所) に到達することです。

関係のある情報を収集し、あてはまるトピックを選択していって、参照されたセクションへ進んでください。

診断ツール

問題が発生したときは、以下のツールを使用することができます。

- ダンプ・ファイル、トラップ・ファイル、エラー・ログ、通知ファイル、およびアラート・ログを含むすべての診断データは、診断データ・ディレクトリー・パス (**diagpath**) データベース・マネージャー構成パラメーターによって指定されたパスにあります。

  この構成パラメーターの値が NULL の場合、診断データは以下のいずれかのディレクトリーまたはフォルダーに書き込まれます。

  – Linux およびUNIX 環境の場合は、`INSTHOME/sqllib/db2dump` です。*INSTHOME* はインスタンスのホーム・ディレクトリーです。
  – サポートされている Windows 環境の場合
    - **DB2INSTPROF** 環境変数が設定されていない場合、`x:¥SQLLIB¥DB2INSTANCE` が使用されます。`x:¥SQLLIB` は **DB2PATH** レジストリー変数で指定されたドライブ参照およびディレクトリー、**DB2INSTANCE** の値はインスタンスの名前です。

      注: ディレクトリーに SQLLIB という名前を付ける必要はありません。
    - **DB2INSTPROF** 環境変数が指定されている場合は、`x:¥DB2INSTPROF¥DB2INSTANCE` が使用されます。 **DB2INSTPROF** はインスタンス・プロファイル・ディレクトリーの名前、**DB2INSTANCE** はインスタンスの名前 (デフォルトでは、 Windows 32 ビット・オペレーティング・システムの **DB2INSTDEF** の値) です。
- Windows オペレーティング・システムについては、イベント・ビューアーを使用して、管理通知ログを表示します。
- 使用できる診断ツールには、**db2trc**、**db2pd**、**db2support**、および **db2diag** が含まれます。
- Linux および UNIX オペレーティング・システムについては、**ps** コマンド。これ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すものです。
- UNIX オペレーティング・システムについては、コア・ファイル。これは重大エラーが起きたとき、現行ディレクトリー内で作成されます。コア・ファイルには、終了したプロセスのメモリー・イメージが入っていて、どの機能がエラーの原因となっているかを判別するのに使用できます。

関係のある情報の収集

トラブルシューティングには、問題の範囲を絞り込み、考えられる原因を調査することが含まれます。はじめに、関係する情報を収集し、分っている事柄、まだ収集していないデータや、省略できるパスなどの判別を行ってください。少なくとも以下の質問に答えるようにしてください。

- 初期の接続は成功しましたか。
- ハードウェアは正常に機能していますか。
- 通信パスは機能していますか。
- 以前のディレクトリー項目を無効にするような通信ネットワークの変更が行われましたか。
- データベースは始動していますか。
- 通信切断は 1 つ以上のクライアントと DB2 Connect サーバー (ゲートウェイ) の間ですか。DB2 Connect ゲートウェイと IBM メインフレーム・データベース・サーバーの間ですか。それとも DB2 Connect Personal Edition と IBM メインフレーム・データベース・サーバーの間ですか。
- メッセージ内容およびメッセージに戻されているトークンから、何を判別することができますか。
- 現時点での db2trc、db2pd、または db2supportなどの診断ツールの使用は効果がありそうですか。
- 他のマシンで同じような作業を行っている場合、正しく作動していますか。
- これがリモート・タスクの場合、ローカルに行っても正常に作動しますか。

初期接続が正常に行われなかった場合

以下の質問を検討して、インストール・ステップが正しく行われたか確認してください。

1. *インストール・プロセスは正常に完了しましたか。*
   - 前提条件のソフトウェア製品はすべて使用可能でしたか。
   - メモリーおよびディスク・スペースは十分ありましたか。
   - リモート・クライアント・サポートはインストールされましたか。
   - 通信ソフトウェアのインストールは、何のエラー状態もなく完了しましたか。
2. *UNIX オペレーティング・システムの場合、製品のインスタンスは作成されましたか。*
   - root として、インスタンス所有者になるユーザーおよび sysadm グループになるグループを作成しましたか。
3. *ライセンス情報は正常に処理されましたか (該当する場合)。*
   - UNIX オペレーティング・システムの場合、ノード・ロック・ファイルを編集し、 IBM が指定したパスワードを入力しましたか。
4. *IBM メインフレーム・データベース・サーバーとワークステーションとの通信の構成は正常に行われましたか。*
   - 考慮すべき 3 つの構成があります。
     a. IBM メインフレーム・データベース・サーバーの構成は、サーバーに対してアプリケーション・リクエスターを識別します。 IBM メインフレー

ム・サーバー・データベース管理システムは、そのリクエスターをロケーション、ネットワーク・プロトコル、およびセキュリティーの点で定義するシステム・カタログ項目を持つことになります。

b. DB2 Connect ワークステーション構成は、サーバーに対してクライアント数を定義し、クライアントに対して IBM メインフレーム・サーバーを定義します。

c. クライアント・ワークステーション構成では、ワークステーションの名前と通信プロトコルが定義されている必要があります。

- 初期接続が実行されなかった場合の問題分析には、 PU (物理装置) 名が完全であって正しいかどうか確認することが必要です。また、TCP/IP 接続を検査し、正しいポート番号とホスト名を指定しているかどうか確認することが必要です。
- IBM メインフレーム・サーバーのデータベース管理者およびネットワーク管理者の両方は、問題の診断に利用可能なユーティリティーを持っています。

5. *IBM メインフレーム・サーバーのデータベースを使用するための、IBM メインフレーム・サーバーのデータベース管理システムが必要とするレベルの権限を持っていますか。*
   - ユーザーのアクセス権限、表修飾子の規則、および予測される結果を考慮してください。
6. *IBM メインフレーム・データベース・サーバーに対してコマンド行プロセッサー (CLP) を使用して SQL ステートメントを発行した場合、正常に実行できませんか。*
   - コマンド行プロセッサー (CLP) を IBM メインフレーム・データベース・サーバーへバインドする手順を守り行いましたか。

初期接続後に発生する問題

問題の範囲を絞り込むのに役立つ始点として、次の質問が挙げられます。

1. *何か特別なもしくは異常な操作状況が見られますか。*
   - それは新しいアプリケーションですか。
   - 新しいプロシージャーが使用されていますか。
   - システムに影響を与える可能性のある変更が最近行われましたか。例えば、アプリケーションまたはシナリオが正常に実行された後、ソフトウェア製品またはアプリケーションのどれかが変更されましたか。
   - アプリケーション・プログラムについて、そのプログラムを作成するためにどのようなアプリケーション・プログラミング・インターフェース (API) が使用されましたか。
   - そのソフトウェアまたは通信 API を使用した他のアプリケーションが、ユーザーのシステムで実行されたことがありますか。
   - フィックスパックを最近インストールしましたか。オペレーティング・システムにおいて、インストールされて以来使用されていない (もしくはロードされていない) フィーチャーを使用しようとして問題が発生した場合は、IBM の最新のフィックスパックを判別し、そのフィーチャーをインストールした後、それをロードしてください。
2. *このエラーは以前にも起こりましたか。*

- 以前のエラー状態には文書化された解決法がありましたか。
- システムへの参加者はだれでしたか。その人たちが、考えられる処置の方向性にヒントを与えることができませんか。

3. *ネットワークに関する情報を戻す通信ソフトウェア・コマンドを利用して、よく調べてみましたか。*
   - TCP/IP コマンド、およびデーモンを使用して取り出した情報には、重要なものが含まれる可能性があります。
4. *SQLCA (SQL 連絡域) に戻される有用な情報がありますか。*
   - 問題処理手順は、 SQLCODE および SQLSTATE の各フィールドの内容を調べるステップを含んでいる必要があります。
   - SQLSTATE によって、アプリケーション・プログラマーはデータベース製品の DB2 ファミリーに共通するエラーのクラスについてテストすることができます。分散リレーショナル・データベース・ネットワーク内でこのフィールドは、共通の基礎を提供する場合があります。
5. *START DBM をサーバーで実行しましたか。* 加えて、DB2COMM 環境変数が、サーバーにリモート・アクセスするクライアント用に正しく設定されているかを確認してください。
6. *同じ作業を行っている他のマシンは、サーバーに正常に接続することができましたか。* サーバーに接続しようとしているクライアントが、最大数に達している場合があります。別のクライアントがサーバーから切断された場合、以前は接続できなかったクライアントは、今は接続することができますか。
7. *マシンは適正なアドレッシングを行っていますか。* そのマシンがネットワーク上で固有のものかどうか検査してください。
8. *リモート接続をしている場合、適正な権限がクライアントに認可されていますか。* インスタンスへの接続が正常に行われても、データベースや表レベルでは権限が認可されていない場合があります。
9. *これはリモート・データベースに接続する最初のマシンですか。* 分散環境内では、ネットワーク間のルーターやブリッジが、クライアントとサーバーとの通信をブロックする場合があります。例えば、TCP/IP を使用する場合は、リモート・ホストを PING できるかどうか確認してください。

サポートされない DDM コマンド

DDM コマンドの BNDCPY、BNDDPLY、DRPPKG、および DSCRDBTBL は、DB2 Version 9.5 for Linux, UNIX, and Windows が DRDA アプリケーション・サーバー (DRDA AS) として機能している場合はサポートされません。

症状

DRDA アプリケーション・リクエスター (DRDA AR) が DB2 Version 9.5 for Linux, UNIX, and Windows に接続されている場合に、以下のいずれかのコマンドを発行すると、そのコマンドは失敗します。

表 81. サポートされない DDM コマンド

DDM コマンド	DDM コード・ポイント	説明
BNDCPY	X'2011'	既存のリレーショナル・データベース (RDB) パッケージをコピーします
BNDDPLY	X'2016'	既存の RDB パッケージをデプロイします
DRPPKG	X'2007'	パッケージをドロップします
DSCRDBTBL	X'2012'	RDB 表を記述します

また、SQLDTA 記述子内でパラメーターによる (または列方向の) 配列入力に使用される以下のコード・ポイントもサポートされません。

表 82. サポートされない FD:OCA データ・オブジェクト

FD:OCA データ・オブジェクト	DDM コード・ポイント	説明
FDOEXT	X'147B'	定様式データ・オブジェクト・コンテンツ体系 (FD:OCA) のデータ・エクステント
FDOOFF	X'147D'	FD:OCA のデータ・オフセット

この状態での最も一般的なエラー・メッセージは、SQL30020N (「実行が、後続のコマンドおよび SQL ステートメントの正常な実行に影響を与える分散プロトコル・エラーのために失敗しました。」) です。

原因

分散データ管理体系 (DDM) は、DRDA プロトコルの一部です。DDM コマンドの BNDCPY、BNDDPLY、DRPPKG、および DSCRDBTBL は、DB2 Version 9.5 for Linux, UNIX, and Windows によってサポートされるすべての DRDA レベルに存在しますが、DRDA アプリケーション・サーバーでは、これらの DDM コマンドをサポートしていません。

同様に、DB2 Version 9.5 for Linux, UNIX, and Windows の DRDA アプリケーション・サーバーでは、 **FDOEXT** と **FDOOFF** のコード・ポイントがサポートされません。これらのコード・ポイントは、列方向配列の入力要求を実行依頼する場合にサーバーに送信される SQLDTA 記述子内で使用されます。

問題の診断

DRDA アプリケーション・サーバー上で DB2 トレースを取得すると、これらのコマンドに応答して `ERROR MSG = Parser: Command Not Supported` に類似したメッセージが示されます。

問題の解決

DDM コマンドの BNDCPY と BNDDPLY 用にサポートされる代替手段は現在存在しません。

パッケージをドロップするには、SQL ステートメントの DROP PACKAGE を使用します。例えば、DB2 Version 9.5 for Linux, UNIX, and Windows の DRDA アプリケーション・サーバーに接続して、EXECUTE IMMEDIATE 要求内で DROP PACKAGE ステートメントを送信します。 DB2 Version 9.5 for Linux, UNIX, and Windows では、この要求を正常に処理します。

RDB 表を記述するには、DDM コマンドの DSCSQLSTT (SQL ステートメントの記述) または PRPSQLSTT (SQL ステートメントの準備) のいずれかを使用します。例えば、表 TAB1 の記述が必要な場合は、`SELECT * FROM TAB1` というステートメントを記述または準備します。

注: DRDA AR で、PRPSQLSTT コマンドを発行する場合は、インスタンス変数 **RTNSQLDA** を TRUE の値で指定する必要もあります。そうしないと、SQLDA Reply Data (SQLDARD) 記述子がサーバーによって返されません。

FDOEXT と **FDOOFF** のコード・ポイントに関する問題を回避するには、パラメーターによる (または列方向の) 配列入力要求ではなく行方向の配列入力要求を使用します。

一般的な DB2 Connect の問題

このトピックでは、DB2 Connect の使用時に接続問題が生じたときの最も一般的な症状をリストします。 どの場合でも、以下の形式で示されます。

- 表示されたメッセージに関連した、メッセージ番号と戻りコード (またはプロトコル固有の戻りコード) の組み合わせ。各メッセージと戻りコードの組み合わせには、個別の見出しがあり、この見出しはメッセージ番号順、その後に戻りコードの順で並べられます。
- 症状。通常は、サンプル・メッセージのリスト形式で示されます。
- 解決方法。エラーの推定原因が示されます。場合によっては、複数の解決方法が提示されることがあります。

SQL0965 または SQL0969

症状 メッセージ SQL0965 および SQL0969 は、DB2 for IBM i、DB2 for z/OS、および DB2 Server for VM and VSE からさまざまな異なる戻りコードとともに発行される場合があります。

いずれかのメッセージが出された場合、そのメッセージを発行したデータベース・サーバー製品の資料で、元の SQL コードを調べる必要があります。

解決方法

IBM メインフレーム・データベースから受信された SQL コードを変換できません。 そのエラー・コードに基づいて問題を訂正してから、失敗したコマンドを再発信してください。

SQL5043N

症状 1 つまたは複数の通信プロトコルに対するサポートが正常に開始できませんでした。ただし、コアとなるデータベース・マネージャーの機能は正常に開始されました。

おそらく、TCP/IP プロトコルが DB2 Connect サーバーで開始されていません。以前に成功したクライアント接続がまだ残っている可能性があります。

diaglevel = 4 の場合、db2diag ログ・ファイルに同様の項目が含まれている可能性があります。例えば次のようになります。

```
2001-05-30-14.09.55.321092   Instance:svtdbm5   Node:000
PID:10296(db2tcpcm)   Appid:none
common_communication  sqlcctcpconnmgr_child   Probe:46
DIA3205E Socket address "30090" configured in the TCP/IP
services file and
required by the TCP/IP server support is being used by another
process.
```

解決方法

この警告は、DB2 Connect (リモート・クライアントのサーバーとして動作している) が 1 つまたは複数のクライアント通信プロトコルを処理する際に問題が発生していることを示しています。これらのプロトコルは TCP/IP およびその他のものであり、このメッセージは通常、DB2 Connect に定義されているこれらの通信プロトコルのいずれかが正しく構成されていないことを示しています。

DB2COMM プロファイル変数が定義されていないか、不正に定義されていることが原因かもしれません。一般に、問題は DB2COMM 変数とデータベース・マネージャー構成で定義した名前 (例えば、svcename または nname) との間のミスマッチの結果です。

可能性のあるシナリオとして 1 つあげられるのは、以前に成功した接続がそのままになっており、構成が変更されていないのに、SQL5043 エラー・メッセージを受け取ってしまったというものです。 これは、TCP/IP プロトコルの使用時にリモート・システムが何らかの理由で接続を異常終了したときに発生する可能性があります。これが発生した場合、接続はクライアント上にまだ存続しているように見えることがあり、下記のコマンドを実行することによってさらなる介入なしで、接続をリストアすることができます。

一番多いのは、 DB2 Connect サーバーに接続しているクライアントの 1 つが TCP/IP ポート上でハンドルを持ったままになるというケースです。DB2 Connect サーバーに接続している各クライアント・マシン上で、以下のコマンドを入力します。

```
db2 terminate
db2stop
```

SQL30020

症状 SQL30020N 実行が、後続のコマンドおよび SQL ステートメントの正常な実行に影響を与える分散プロトコル・エラーのために失敗しました。

解決方法

このエラーが発生したら、サービス担当者に連絡してください。サービス担当者に連絡する前に db2support コマンドを実行します。

SQL30060

症状 SQL30060N *"<authorization-ID>"* が、処理 *"<operation>"* を実行する権限を持っていません。

解決方法

DB2 for z/OSへの接続時に、コミュニケーション・データベース (CDB) 表が正しく更新されていません。

SQL30061

症状 間違った IBM メインフレーム・データベース・サーバー・ロケーションに接続しています。ターゲット・データベースが見つかりません。

解決方法

DCS ディレクトリー項目に誤ったサーバー・データベース名を指定した可能性があります。これが生じた場合、SQLCODE -30061 がアプリケーションに戻されます。

DB2 ノード、データベース、および DCS ディレクトリー項目を調べてください。 DCS ディレクトリー項目のターゲット・データベース名のフィールドは、プラットフォームに基づいたデータベースの名前に対応していなければなりません。例えば、DB2 for z/OS データベースの場合、使用する名前はブートストラップ・データ・セット (BSDS) の『LOCATION=*locname*』フィールドで使用した名前と同じでなければなりません。これは、分散データ機能 (DDF) を開始するときの DSNL004I メッセージにも示されています (LOCATION=*location*)。

TCP/IP ノードへの正しいコマンドは次のとおりです。

```
db2 catalog tcpip node <node_name> remote <host_name_or_address>
            server <port_no_or_service_name>
db2 catalog dcs database <local_name> as <real_db_name>
db2 catalog database <local_name> as <alias> at <node node_name>
            authentication server
```

その後にデータベースへ接続するには、次のコマンドを実行します。

```
db2 connect to <alias> user <user_name> using <password>
```

SQL30081N (戻りコード 79)

症状

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".
Location
where the error was detected: "".  Communication function
detecting the error:
"connect".  Protocol specific error code(s): "79", "*", "*".
SQLSTATE=08001
```

解決方法

このエラーは、リモート・クライアントが DB2 Connect サーバーへの接続に失敗した場合に発生する可能性があります。 さらに、DB2 Connect サーバーから IBM メインフレーム・データベース・サーバーへの接続時にも発生することがあります。

1. DB2COMM プロファイル変数が、DB2 Connect サーバーで正しく設定されていない可能性があります。このことを確認してください。例えば、AIX で DB2 Enterprise Server Edition を実行している場合は、コマンド db2set db2comm=tcpip は sqllib/db2profile に存在していなければなりません。

2. TCP/IP のサービス名およびポート番号の仕様が、IBM Data Server Client と DB2 Connect サーバーで一致しない場合があります。 両方のマシンで、TCP/IP services ファイル内の項目を確認してください。
3. DB2 Connect サーバーで DB2 が開始していることをチェックします。次のコマンドを使用して、データベース・マネージャー構成の diaglevel を 4 に設定してください。

   ```
   db2 update dbm cfg using diaglevel 4
   ```

   DB2 を停止して再始動したら、db2diag ログ・ファイルを参照して、DB2 TCP/IP 通信が開始していることをチェックします。次のような出力が含まれているはずです。

   ```
   2001-02-03-12.41.04.861119   Instance:svtdbm2   Node:00
   PID:86496(db2sysc)   Appid:none
   common_communication  sqlcctcp_start_listen   Probe:80
   DIA3000I "TCPIP" protocol support was successfully started.
   ```

SQL30081N (プロトコル固有のエラー・コード 10032)

症状

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".
Location
where the error was detected: "9.21.85.159".  Communication
function detecting
the error: "send".  Protocol specific error code(s): "10032",
"*", "*".
SQLSTATE=08001
```

解決方法

このエラー・メッセージは、TCP/IP 通信に失敗したマシンから切断しようとするときに受け取ることがあります。 TCP/IP サブシステムの問題を修正してください。

問題を修正する方法は、ほとんどのマシンでは、単にそのマシンの TCP/IP プロトコルを再始動することです。マシン全体を再生しなければならないこともあります。

CONNECT 時の SQL30082 RC=24

症状 SQLCODE -30082 指定されたユーザー名またはパスワードが正しくありません。

解決方法

必要であれば CONNECT ステートメントに正しいパスワードを指定してあるか確認してください。 ターゲット・サーバー・データベースへ送信するときに使用できないパスワードです。パスワードを IBM Data Server Client からターゲット・サーバー・データベースに送信する必要があります。 特定のプラットフォーム、例えば AIX などでは、パスワードは CONNECT ステートメントに指定してある場合に限り、入手することができます。

第 8 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 9 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。

1. 以下の Web ページからそれぞれフィックス・リストを表示し、フィックスパックを取得できます。
   - Linux、 UNIX、および Windows サポート用の DB2 9
   - Linux、 UNIX、および Windows 用の DB2 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したお客様が一般にこれを入手できます。
- 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したお客様が一般にこれを入手できます。
- 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 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/support/db2_9/) で「My notifications」の E メール更新に登録してください。

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。

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。

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。

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。

第 10 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: 最初のオカレンスのデータ収集) 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こっ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。

- 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、適切なリカバリー・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す。

- 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

  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

ご使用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ている DB2 診断情報は、**diagpath** データベース・マネージャー構成パラメーターによって指定される診断データ・ディレクトリーに保管されます。

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するかは、**diagpath** データベース・マネージャー構成パラメーターを使った、診断データ・ディレクトリー・パスの指定によって決めることができます。

単一の診断データ・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。

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。

diagpath データベース・マネージャー構成パラメーターの設定について詳しくは、「*データベース: 管理の概念および構成リファレンス*」の『diagpath - 診断データ・ディレクトリー・パスの構成パラメーター』を参照してください。

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。

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。

ホストごと、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、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別個のディレクトリーを作成し、名前を付けることができます。こうして、今まで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・ダンプの発生元のホストまたはデータベース・パーティションに従って別個のディレクトリーに保管することができます。

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・ダンプの発生元のホスト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個別に保管するために、診断データ・ディレクトリー・パスを分割することができます。

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

- **物理ホストごとの診断データ・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・パスを物理ホストごとに分割します。

      ```
      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ホスト名で作成されます。

      Default_diagpath/HOST_*hostname*

– 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、診断データ・ディレクトリー・パス /home/usr1/db2dump/ を物理ホストごとに分割します。

```
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ホスト名で作成されます。

/home/usr1/db2dump/HOST_*hostname*

- **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/NODE*number*

– 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/NODE*number*

- **物理ホストごとおよび物理ホストのデータベース・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・パスを物理ホストごとおよび物理ホストのデータベース・パーティションごとに分割します。

```
db2 update dbm cfg using diagpath '"$h$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ホスト上の各論理パーティションのサブディレクトリーがホスト名およびパーティション番号で作成されます。

Default_diagpath/HOST_*hostname*/NODE*number*

– 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・パスを物理ホストごとおよび物理ホストのデータベース・パーティションごとに分割します。

    ```
    db2 update dbm cfg using diagpath '"/home/usr1/db2dump/ $h$n"'
    ```

    注: ブランク・スペースによって /home/usr1/db2dump/ と hn 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ホスト上の各論理パーティションのサブディレクトリーがホスト名およびパーティション番号で作成されます。

  /home/usr1/db2dump/HOST_*hostname*/NODE*number*

  例えば、boson という名前の AIX ホストに、ノード番号としてそれぞれ 0、1、2 を持つ 3 つのデータベース・パーティションがあるとします。ディレクトリーのリスト出力の例は、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 db2sampl_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
```

次の作業

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・パスが指定される場合 ($n または hn) には常に、ホストごとに 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ホストにデータベース・パーティション 0 が存在しない場合には、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 hn 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 469 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ー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ー
- ファイル・パス
- ストレージの問題
- モニター・アクティビティー
- 索引付けアクティビティー
- 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』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』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』トピックを参照してください。

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。

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。

- 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、以下のようにします。

  ```
  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に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。

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。

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.log) 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, sqldInitDBCB, 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″ にというように、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 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。

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。

- 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、以下のようにします。

  ```
  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、共用メモリー・セグメント、セマフォー)
- 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 (DB2 がリソースを求めて競合するかどうか)

- 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。次の演習では、トラップ・ファイルを取り上げます。トラップ・ファイルには、DB2 によって生成されるさらに多くのデータが含まれています。これらのデータは、ユーザー環境やリソースの制限を把握するのに役立ちます。

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"
        "db2systm"
        "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。

- システムのリブ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で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_all 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、ユーティリティー、トランザクション、バッファー・プール、ロック、トランザクション・ログ、表スペース、およびコンテナー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/sqllib/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.
          Diagnostic information has been recorded.  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
0x0FFFFFFFFFFFD5C0 : 0000 000B 0000 0000 0000 0009 0000 0000    ................
0x0FFFFFFFFFFFD5D0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0FFFFFFFFFFFD5E0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0FFFFFFFFFFFD5F0 : 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、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。

FODC 情報は、以下のファイルで確認できます。

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.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: **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、単一の診断ログ・ファイルの動作 (db2diag.log) は、循環ログの動作 (db2diag.*N*.log) に変更されます。

DB2 管理サーバー (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ーの働きをします。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

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ー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になる場合があります。

一般的な障害問題に基づく診断情報の収集

インスタンスに影響を与える障害が発生した時点で、診断情報がパッケージに自動的に収集されるように設定できます。パッケージ内の情報を手動で作成することも可能です。

DB2 のインスタンスやデータベースで作業しているときにトラブルが発生した場合は、その問題が起きた時点でデータを収集してください。First Occurrence Data Collection (FODC) は、DB2 環境でトラブルが発生したときに実行されていたアクションを表すために使用する用語です。db2pdcfg ツールを使用して DB2FODC レジストリー変数にオプションを設定することによって、障害の発生時にどのデータを収集するのかを制御できます。DB2FODC レジストリー変数のオプションを変更するには、db2pdcfg -fodc コマンドを使用します。これらのオプションは、FODC 状態のデータ・キャプチャーに関するデータベース・システムの動作に影響を与えます。

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。診断メッセージには、FODC ディレクトリー名と、FODC ディレクトリーが作成された時のタイム・スタンプが組み込まれます。 FODC パッケージの記述ファイルは、新しい FODC ディレクトリーに配置されます。

表 83. 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*	インスタンス全体でのトラップが発生した	自動	db2cos_trap(.bat)

表 83. 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_Panic_*timestamp*	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)
FODC_BadPage_*timestamp*	不正なページが検出された	自動	db2cos_datacorruption(.bat)
FODC_DBMarkedBad_*timestamp*	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_Node#	EDU 全体の索引エラーが発生した。(db2cos_indexerror_short(.bat) および/または db2cos_indexerror_long(.bat) がディレクトリーにダンプされます。)	自動	N/A

診断情報の手動収集

ハングする可能性がある状況や、重大なパフォーマンス上の問題がある状況に関する情報は、First Occurrence Data Collection コマンド (db2fodc) を使用して収集できます。db2fodc コマンドを実行すると、新しいディレクトリー FODC_hang_*timestamp* が、現在の診断パスに自動的に作成されます。その後、db2cos_hang スクリプトが実行されます。このスクリプトは、データを収集して FODC サブディレクトリーに配置するデータ収集操作を制御します。この FODC サブディレクトリーが存在するかどうかは、db2fodc コマンドの実行方法や DB2 レジストリー変数の構成に依存しています。

表 84. 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_Hang_*timestamp*	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_Perf_*timestamp*	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)

表 84. 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_IndexError_ *timestamp*_PID_EDUID_Node# 内にあるスクリプト	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。 DPF では、db2_all ″<<+node#< db2fodc -indexerror *FODC_IndexError_directory* [basic | full]″ を使用します。node# は、*FODC_IndexError_directory* ディレクトリー名の最後の番号です。 db2fodc -indexerror を db2_all コマンドと共に使用する場合は、絶対パスが必要です。	手動	db2cos_indexerror_long(.bat)、db2cos_indexerror_short (.bat)

診断情報の自動収集のための構成

データベース・マネージャーが自動的にアクションを実行するには、どのアクションを実行するのかをデータベース・マネージャーに対して指定する必要があります。

データベース操作中にエラーや警告が検出されたときに、データベース・マネージャーが実行するアクションを示すフラグを設定します。実行するアクションとしては、次のようなアクションがあります。

- db2diag ログ・ファイルのスタック・トレースの生成。(デフォルト)
- コールアウト・スクリプト db2cos の実行。(デフォルト)
- トレース (db2trc) コマンドの停止。

First Occurrence Data Capture (FODC) オプションの変更

問題判別動作用の DB2 データベースの構成 (db2pdcfg) コマンドを使用して、First Occurrence Data Capture (FODC) オプションを変更します。DB2FODC レジストリー変数の FODC オプションを設定するには、db2pdcfg ツールを使用します。これらのオプションは、FODC 状態のデータ・キャプチャーに関するデータベース・システムの動作に影響を与えます。

FODC の一部として収集されたデータとその配置

インスタンス内で発生した障害のタイプによっては、First Occurrence Data Capture (FODC) の結果としてサブディレクトリーが作成され、特定の内容が収集されます。一連のサブディレクトリーが作成され、それぞれのファイルやログが格納されます。

FODC 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 の構成や db2fodc の実行モードによっては、これらのディレクトリーが常に存在するとは限りません。

障害のタイプに応じて、以下の内容が FODC ディレクトリーとサブディレクトリーにあります。

- トラップ・ファイル
- 障害発生時のデータ・キャプチャーで生成され、各種のコンポーネントによって完成された、あらゆる種類のバイナリー・ファイルとプレーン・テキスト・ダンプ・ファイル
- db2evlog のイベント・ログ・ファイル
- DB2 トレース・ダンプ (障害発生時にトレースが実行されていた場合)
- コア・ファイルを含むディレクトリー
- DB2FODC ログ・ファイル
  - 手動 FODC では、「ログ」ファイルを 1 つだけ使用します。ハングの場合は db2fodc_hang.log、不正ページの場合は db2fodc_badpage.log です。
- データ破損に関連した情報
  - プロセス情報: ps (UNIX の場合) と db2pd -edus 出力
  - db2support によって現在収集されている追加情報 (オプション)
    - errpt -a 出力 (AIX)
    - UNIX プラットフォームのシステム・ログ。例えば、SunOS では /var/adm/messages、HP/UX では /var/adm/syslog.log になります。これが行われるのは、これらのファイルが収集されている可能性がある場合です (Linux の場合は、syslog ファイルをコピーするためのルート・アクセスが必要です)。

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。

DB2 Query Patroller と First Occurrence Data Capture (FODC)

DB2 Query Patroller 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。

qpdiag.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: **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。
- Query Patroller システムがアクティブになると自動的に作成されます。
- Query Patroller についての情報および診断レコードが含まれます。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・サポートが使用するためのものです。

qpmigrate.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: **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。
- qpmigrate ユーティリティーによって自動的に作成されます。qpmigrate コマンドは、Query Patroller のインストール時に暗黙的に実行することも (Query Patroller を実行する対象の既存のデータベースを指定した場合)、インストール後に明示的に実行することもできます。
- Query Patroller があるバージョンから別のバージョンへマイグレーションされる際の情報およびエラー・メッセージをキャプチャーします。これは、Query Patroller 管理者が使用するためのものです。

qpsetup.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: **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。
- qpsetup ユーティリティーによって自動的に作成されます。qpsetup コマンドは、Query Patroller のインストール時に暗黙的に実行することも (Query Patroller を実行する対象の既存のデータベースを指定した場合)、インストール後に明示的に実行することもできます。
- qpsetup ユーティリティーの実行中に発生する情報およびエラー・メッセージをキャプチャーします。これは、Query Patroller 管理者が使用するためのものです。

qpuser.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: **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。
- Query Patroller システムがアクティブになると自動的に作成されます。
- Query Patroller に関する通知メッセージが含まれています。例えば、Query Patroller が停止した時や開始した時を示します。これは、Query Patroller 管理者が使用するためのものです。

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

コマンド・エディター、データウェアハウス・センター、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。

コマンド・エディターのログ

- 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: このログ・ファイルの名前およびロケーションは、DB2 ツールバーの「コマンド・エディター」ページを使用して指定されます。パスが指定されていない場合、ログは Windows では `$DB2PATH¥sqllib¥tools` ディレクトリーに、Linux および UNIX では `$HOME/sqllib/tools` 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) に保管されます。
- コマンド・エディターで「**コマンド履歴をファイルに記録する**」を選択し、ファイルとロケーションを指定した場合に作成されます。
- コマンド・エディターからのコマンドおよびステートメントの実行履歴が含まれます。

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。
- 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。
- 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:
        SQLO ; oper system services (Component Index: 15)
Reason Code:
        10 (0x000A)

Identifer:
        SQLO_FNEX
        SQLO_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 SQLO_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 値または ECF 値の完全なリストを参照するには、それぞれコマンド db2diag -rc zrc および db2diag -rc ecf を使用します。

メッセージの概要

ここでは、DB2 がインストールされているオペレーティング・システムの機能に通じていることが前提となります。以下の章に含まれる情報を使用して、エラーまたは問題を識別し、適切なリカバリー・アクションを使用して問題を解決することができます。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。

メッセージ構造

メッセージ・ヘルプはメッセージの原因を説明し、メッセージに応答して取るべきアクションを示します。

メッセージ ID は、3 文字のメッセージ接頭語、4 または 5 桁のメッセージ番号、1 文字の接尾部からなります。例えば、*SQL1042C* です。メッセージ接頭語のリストは、613 ページの『メッセージ・ヘルプの呼び出し』および 615 ページの『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、*C* で終わるメッセージ ID は重大メッセージ、*E* で終わるものは緊急メッセージ、*N* で終わるものはエラー・メッセージ、*W* で終わるものは警告メッセージ、*I* で終わるものは通知メッセージを表します。

ADM メッセージの場合、*C* で終わるメッセージ ID は重大メッセージ、*E* で終わるものは緊急メッセージ、*W* で終わるものは重要メッセージ、*I* で終わるものは通知メッセージを表します。

SQL メッセージの場合、*C* で終わるメッセージ ID は重大システム・エラー、*N* で終わるものはエラー・メッセージ、*W* で終わるものは警告または通知メッセージを表します。

一部のメッセージには、トークン (メッセージ変数と呼ぶ場合もあります) が含まれます。 DB2 がトークンを含むメッセージを生成する際に、各トークンは発生したエラー条件に固有の値に置き換えられ、ユーザーがそのエラー・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 は以下のとおりです。

- コマンド行プロセッサーからの場合

  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・センターからの場合

  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>″ と ″<length>″ があります。実行時にこのメッセージが生成される時に、これらのメッセージ・トークンはそれぞれ、エラーを引き起こしたオブジェクトの実際の名前と、そのタイプのオブジェクトに許される最大長で置き換えられます。

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。例えば以下のようになります。

```
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

以下の DB2 メッセージはコマンド行プロセッサーからアクセス可能です。

接頭部 説明

ADM 多数の DB2 コンポーネントが生成するメッセージ。これらのメッセージは管理通知ログ・ファイルに書き込まれ、システム管理者に追加情報を提供するためのものです。

AMI MQ Application Messaging Interface が生成するメッセージ

ASN DB2 レプリケーションが生成するメッセージ

CCA 構成アシスタントが生成するメッセージ

CLI コール・レベル・インターフェースが生成するメッセージ

DBA データベース管理ツールが生成するメッセージ

DBI インストールおよび構成で生成されるメッセージ

DBT データベース・ツールが生成するメッセージ

DB2 コマンド行プロセッサーが生成するメッセージ

DQP Query Patroller が生成するメッセージ

EAS 組み込みアプリケーション・サーバーが生成するメッセージ

EXP EXPLAIN ユーティリティーが生成するメッセージ

GSE DB2 Spatial Extender が生成するメッセージ

LIC DB2 License Manager が生成するメッセージ

MQL MQ Listener が生成するメッセージ

SAT サテライト環境で生成されるメッセージ

SPM 同期点マネージャーが生成するメッセージ

SQL 警告やエラー状態が検出されたときにデータベース・マネージャーが生成するメッセージ

XMR XML Metadata Repository が生成するメッセージ。

メッセージ・ヘルプを呼び出すには、コマンド行プロセッサーを開いて、次のように入力します。

```
? 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。

表示された 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取得できます。

```
? nnnnn
```

または

```
? nn
```

ここで、*nnnnn* は 5 桁の SQLSTATE (英数字) 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 は、大文字と小文字の区別をしません。さらに、1 文字の接尾部はオプションで、無視されます。

このため、以下のコマンドは同じ結果になります。

- ? SQL0000N
- ? sql0000
- ? SQL0000w

UNIX ベースのシステムのコマンド行でメッセージ・ヘルプを実行するには、次のように入力します。

```
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
```

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 監査機能が生成するメッセージ。

DIA 多数の DB2 コンポーネントが生成する診断メッセージ。これらのメッセージは 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と DB2 サービス担当員に追加情報を提供するためのものです。

GOV DB2 管理プログラム・ユーティリティーが生成するメッセージ。

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が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する場合、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、セキュリティー、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ー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たくさん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。

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。

- 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。

以下のステップを使用して、コア・ファイル・ダンプが発生する原因になった関数を判別できます。

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。

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 *) pcAdress[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。

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、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

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。 ビューアーを開く方法は、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。

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

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・ビューアーに再ロードできます。この形式を使用した場合、ビューアー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て便利です。
- 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、ビューアー内でそのワー

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 システム上で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。

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を履歴として記録します。ワトソン博士のログよりも DB2 トラップ・ファイルの方が役立ちますが、ワトソン博士のログはシステム全体の安定性を評価したり、DB2 トラップの履歴を保存するために使用できます。

ワトソン博士のログ・ファイルは次のロケーションで見つけることができます。 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ーにあります。

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。

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。

「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 11 章 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。

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。

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。

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 ソフトウェア・サポートへのデータの送信

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。

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。

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
- 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。

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。

第 3 部 付録

付録 A. DB2 技術情報の概説

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)
  - DB2 ツールのヘルプ
  - 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、DB2 ライブラリーについて説明しています。DB2 ライブラリーに関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。英語および翻訳された DB2 バージョン 9.7 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が示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: *DB2 インフォメーション・センター*は、PDF やハードコピー資料よりも頻繁に更新されます。

表 85. 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	SC88-5883-01	入手可能	2009 年 11 月
管理ルーチンおよびビュー	SC88-5880-01	入手不可	2009 年 11 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	SC88-5885-01	入手可能	2009 年 11 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	SC88-5886-01	入手可能	2009 年 11 月
コマンド・リファレンス	SC88-5884-01	入手可能	2009 年 11 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	SC88-5903-00	入手可能	2009 年 8 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	SC88-5904-01	入手可能	2009 年 11 月
データベース: 管理の概念および構成リファレンス	SC88-5870-01	入手可能	2009 年 11 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	SC88-5872-01	入手可能	2009 年 11 月
データベース・セキュリティー・ガイド	SC88-5905-01	入手可能	2009 年 11 月

表 85. 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 Text Search ガイド	SC88-5902-01	入手可能	2009 年 11 月
ADO.NET および OLE DB アプリケーションの開発	SC88-5874-01	入手可能	2009 年 11 月
組み込み SQL アプリケーションの開発	SC88-5875-01	入手可能	2009 年 11 月
Java アプリケーションの開発	SC88-5878-01	入手可能	2009 年 11 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SC88-5879-00	入手不可	2009 年 8 月
SQL および外部ルーチンの開発	SC88-5876-01	入手可能	2009 年 11 月
データベース・アプリケーション開発の基礎	GI88-4201-01	入手可能	2009 年 11 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4202-00	入手可能	2009 年 8 月
グローバリゼーション・ガイド	SC88-5906-00	入手可能	2009 年 8 月
DB2 サーバー機能 インストール	GC88-5888-01	入手可能	2009 年 11 月
IBM データ・サーバー・クライアント機能 インストール	GC88-5889-00	入手不可	2009 年 8 月
メッセージ・リファレンス 第 1 巻	SC88-5897-00	入手不可	2009 年 8 月
メッセージ・リファレンス 第 2 巻	SC88-5898-00	入手不可	2009 年 8 月
Net Search Extender 管理およびユーザーズ・ガイド	SC88-5901-01	入手不可	2009 年 11 月
パーティションおよびクラスタリングのガイド	SC88-5907-01	入手可能	2009 年 11 月
pureXML ガイド	SC88-5895-01	入手可能	2009 年 11 月
Query Patroller 管理およびユーザーズ・ガイド	SC88-5908-00	入手不可	2009 年 8 月

表 85. 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
Spatial Extender および Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス	SC88-5900-00	入手不可	2009 年 8 月
SQL プロシージャー言語: アプリケーションのイネーブルメントおよびサポート	SC88-5877-01	入手可能	2009 年 11 月
SQL リファレンス 第 1 巻	SC88-5881-01	入手可能	2009 年 11 月
SQL リファレンス 第 2 巻	SC88-5882-01	入手可能	2009 年 11 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	SC88-5871-01	入手可能	2009 年 11 月
DB2 バージョン 9.7 へのアップグレード	SC88-5887-01	入手可能	2009 年 11 月
Visual Explain チュートリアル	SC88-5899-00	入手不可	2009 年 8 月
DB2 バージョン 9.7 の新機能	SC88-5893-01	入手可能	2009 年 11 月
ワークロード・マネージャー ガイドおよびリファレンス	SC88-5894-01	入手可能	2009 年 11 月
XQuery リファレンス	SC88-5896-01	入手不可	2009 年 11 月

表 86. 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	SC88-5891-01	入手可能	2009 年 11 月
DB2 Connect サーバー機能 インストールおよび構成	SC88-5892-01	入手可能	2009 年 11 月
DB2 Connect ユーザーズ・ガイド	SC88-5890-01	入手可能	2009 年 11 月

表 87. *Information Integration* 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
Information Integration: フェデレーテッド・システム管理ガイド	SC88-4166-02	入手可能	2009 年 8 月
Information Integration: レプリケーションおよびイベント・パブリッシングのための ASNCLP プログラム・リファレンス	SC88-4167-04	入手可能	2009 年 8 月
Information Integration: フェデレーテッド・データ・ソース構成ガイド	SC88-4185-02	入手不可	2009 年 8 月
Information Integration: SQL レプリケーション ガイドとリファレンス	SC88-4168-02	入手可能	2009 年 8 月
Information Integration: レプリケーションとイベント・パブリッシング 概説	GC88-4187-02	入手可能	2009 年 8 月

DB2 の印刷資料の注文方法

DB2 の印刷資料が必要な場合、オンラインで購入することができますが、すべての国および地域で購入できるわけではありません。 DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。 *DB2 PDF* ドキュメンテーション DVD の一部のソフトコピー・ブックは、印刷資料では入手できないことに留意してください。例えば、「*DB2 メッセージ・リファレンス*」はどちらの巻も印刷資料としては入手できません。

DB2 PDF ドキュメンテーション DVD で利用できる DB2 の印刷資料の大半は、IBM に有償で注文することができます。国または地域によっては、資料を IBM Publications Center からオンラインで注文することもできます。お客様の国または地域でオンライン注文が利用できない場合、DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。 DB2 PDF ドキュメンテーション DVD に収録されている資料の中には、印刷資料として提供されていないものもあります。

注: 最新で完全な DB2 資料は、DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7) で参照することができます。

DB2 の印刷資料は以下の方法で注文することができます。

- 日本 IBM 発行のマニュアルはインターネット経由でご購入いただけます。詳しくは http://www.ibm.com/shop/publications/order をご覧ください。資料の注文情報にアクセスするには、お客様の国、地域、または言語を選択してください。その後、各ロケーションにおける注文についての指示に従ってください。

- DB2 の印刷資料を IBM 営業担当員に注文するには、以下のようにします。
  1. 以下の Web サイトのいずれかから、営業担当員の連絡先情報を見つけてください。
     - IBM Directory of world wide contacts (www.ibm.com/planetwide)
     - IBM Publications Web サイト (http://www.ibm.com/shop/publications/order)。国、地域、または言語を選択し、お客様の所在地に該当する Publications ホーム・ページにアクセスしてください。このページから、「このサイトについて」のリンクにアクセスしてください。
  2. 電話をご利用の場合は、DB2 資料の注文であることをご指定ください。
  3. 担当者に、注文する資料のタイトルと資料番号をお伝えください。タイトルと資料番号は、628 ページの『DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)』でご確認いただけます。

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果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。

SQL 状態ヘルプを開始するには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate or ? 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

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/ にアクセスしてください。

DB2 インフォメーション・センターでの希望する言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、 DB2 インフォメーション・センターでは英語でトピックが表示されます。

- Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。
  1. Internet Explorer の「**ツール**」 -> 「**インターネット オプション**」 -> 「**言語...**」ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     - リストに新しい言語を追加するには、「**追加...**」ボタンをクリックします。

       注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。
     - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。
- Firefox または Mozilla Web ブラウザーの場合に、希望する言語でトピックを表示するには、以下のようにします。
  1. 「**ツール**」 -> 「**オプション**」 -> 「**詳細**」 ダイアログの「**言語**」セクションにあるボタンを選択します。「設定」ウィンドウに「言語」パネルが表示されます。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     - リストに新しい言語を追加するには、「**追加...**」ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
     - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。

ブラウザーとオペレーティング・システムの組み合わせによっては、オペレーティング・システムの地域の設定も希望のロケールと言語に変更しなければなりません。

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された 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 9.7 インフォメーション・センターが既にインストールされている必要があります。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 - 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、更新中にインフォメーション・センターが使用できなくなる時間が最小限で済むというメリットもあります。さらに、自動更新は、定期的に実行する他のバッチ・ジョブの一部として実行されるように設定することができます。
- 手動更新 - 更新処理中にフィーチャーまたは言語を追加する場合に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。

手順

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。

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・センターを自動で更新するには、次のようにします。

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/V9.7` 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 `ic-update` スクリプトを実行します。

      ```
      ic-update
      ```
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¥Version 9.7` 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 ic-update.bat ファイルを実行します。

```
ic-update.bat
```

結果

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、doc¥eclipse¥configuration ディレクトリーにあります。ログ・ファイル名はランダムに生成された名前です。例えば、1239053440785.log のようになります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

このタスクについて

ローカルにインストールされた *DB2 インフォメーション・センター* を手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の *DB2 インフォメーション・センター* を停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。DB2 インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

   注: ご使用の環境において、インターネットに接続されていないマシンに *DB2 インフォメーション・センター* の更新をインストールする必要がある場合、インターネットに接続されていて *DB2 インフォメーション・センター* がインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシーを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。
   更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。
3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の *DB2 インフォメーション・センター* を再開します。

注: Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、 「**管理者として実行**」を選択します。

手順

コンピューターまたはイントラネット・サーバーにインストール済みの *DB2* インフォメーション・センター を更新するには、以下のようにします。

1. *DB2* インフォメーション・センター を停止します。
   - Windows では、「**スタート**」 → 「**コントロール パネル**」 → 「**管理ツール**」 → 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**停止**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv97 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¥Version 9.7 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/V9.7 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 (Find Updates)**」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/db2icdv97 start
     ```

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 製品のさまざまな機能について学習するのを支援します。この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/db2help/) 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。

「*Visual Explain チュートリアル*」の『**Visual Explain**』
Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 トラブルシューティング情報

DB2 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 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 問題判別ガイド*」、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、DB2 診断ツールおよびユーティリティーを使用して、問題を切り分けて識別する方法、最も頻繁に起こる幾つかの問題に対するソリューションについての情報、および DB2 データベース製品を使用する際に発生する可能性のある問題の解決方法についての他のアドバイスがあります。

DB2 Technical Support の Web サイト
現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、DB2 Technical Support の Web サイトを参照してください。Technical Support サイトには、最新の DB2 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。

DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/support/db2_9/) 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。

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。

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で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

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。

〒242-8502
神奈川県大和市下鶴間1623番14号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

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

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを

経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは、現存するままの状態で提供されるものであり、いかなる種類の保証も提供されません。 IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*お客様の会社名*) (*西暦年*). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _*年を入れる*_. All rights reserved.

商標

IBM、IBM ロゴおよび ibm.com® は、世界の多くの国で登録された International Business Machines Corp. 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、Sun Microsystems, Inc. の米国およびその他の国における商標です。
- 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。
- Intel®、Intel ロゴ、Intel Inside®、 Intel Inside ロゴ、Intel® Centrino®、Intel Centrino ロゴ、Celeron®、Intel® Xeon®、Intel SpeedStep®、Itanium®、Pentium® は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、 Windows、 Windows NT®、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

［サ行］

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

A

C

D

E

F

H

I

J

L

M

N

X

Z

[特殊文字]

Printed in Japan

SC88-5871-01

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

IBM DB2 9.7 for Linux, UNIX, and Windows　バージョン 9 リリース 7　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