IBM® DB2 Universal Database™

管理ガイド: プランニング

バージョン 8.2

SC88-9135-01
(英文原典：SC09-4822-01)

IBM® DB2 Universal Database™

管理ガイド: プランニング

バージョン 8.2

SC88-9135-01
(英文原典：SC09-4822-01)

ご注意!

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

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

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

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

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

(URL は、変更になる場合があります)

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

原　典： SC09-4822-01
IBM® DB2 Universal Database™
Administration Guide: Planning
Version 8.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.8

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

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

目次

第 6 章 分散データベースの設計 . . . 197

第 7 章 XA 準拠のトランザクション・マネージャー用の設計 211

第 3 部 付録 241

付録 A. リリース間の非互換性 243

付録 B. 各国語サポート (NLS) 273

付録 C. 64 ビット環境におけるラージ・ページのサポートの使用可能化 (AIX) 331

付録 D. DB2 Universal Database の技術情報の概要 333

付録 E. 特記事項 365

索引 369

IBM と連絡をとる. 377

本書について

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

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

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

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

  コントロール・センターは、3 種類のビューで利用できます。

  - 基本。このビューには、データベース、表、ストアード・プロシージャーなどの重要なオブジェクトに対する DB2 UDB のコア関数が表示されます。
  - 拡張。このビューには、使用可能なすべてのオブジェクトおよびアクションが表示されます。エンタープライズ環境で作業していて、 DB2 for z/OS または IMS に接続する場合には、このビューを使用します。

– カスタム。このビューでは、オブジェクト・ツリーとオブジェクト・アクションを調整する機能が備わっています。

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

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

本書の対象読者

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

本書の構成

本書には、以下の主な項目に関する情報が記載されています。

データベースの概念

- 『第 1 章 基本的なリレーショナル・データベースの概念』では、データベース・オブジェクトおよびデータベース概念の概要を説明します。
- 『第 2 章 並列データベース・システム』では、 DB2 で実現される並列処理のタイプを紹介します。
- 『第 3 章 データウェアハウジングについて』では、データウェアハウジングおよびウェアハウジング・タスクについて概説します。

データベースの設計

- 『第 4 章 論理データベースの設計』では、論理データベースの設計に関する概念と指針について説明します。
- 『第 5 章 物理データベースの設計』では、物理データベースの設計 (スペース所要量および表スペース設計を含む) の指針について説明します。

- 『第 6 章 分散データベースの設計』では、単一トランザクションで複数のデータベースをアクセスする方法を説明します。
- 『第 7 章 XA 準拠のトランザクション・マネージャー用の設計』では、分散トランザクション処理環境でデータベースを使用する方法について説明します。

付録

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

他の管理ガイドの巻の概要

管理ガイド: インプリメンテーション

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

設計の実現

- 『データベースを作成する前に』では、データベースを作成する際の前提条件について説明します。
- 『DB2 Administration Server (DAS) の作成と使用』では、 DAS について、またその作成と使用の方法について説明します。
- 『データベースの作成』では、データベースおよび関係するデータベース・オブジェクトの作成に関連したタスクを説明します。
- 『表および関連するその他の表オブジェクトの作成』では、データベース設計のインプリメントにおいて、特定の特性を備えたデータベースを作成する方法について説明します。
- 『データベースの変更』では、データベースを変更する前に実行しなければならないタスクや、データベースまたは関係するデータベース・オブジェクトの変更またはドロップに関するタスクについて説明します。
- 『表および関連するその他の表オブジェクトの変更』では、表をドロップする方法や、それらの表に関連する特定の特性を変更する方法について説明します。関連する表オブジェクトのドロップや変更についても、ここで説明されています。

データベースのセキュリティー

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

付録

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

注: この資料からは 2 つの章が除去されています。

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

「*データ移動ユーティリティー ガイドおよびリファレンス*」が、これらのトピックに関する主要かつ唯一の情報源です。

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

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

「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」が、これらのトピックに関する主要かつ唯一の情報源です。

管理ガイド : パフォーマンス

「*管理ガイド: パフォーマンス*」では、パフォーマンスに関する問題、つまり、アプリケーションや DB2 Universal Database 製品のパフォーマンスの設定、テスト、および改善に関連したトピックや問題を扱います。この巻のそれぞれの章と付録は、以下のように構成されています。

パフォーマンスの紹介

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

アプリケーション・パフォーマンスのチューニング

- 『アプリケーションについての考慮事項』では、アプリケーションの設計時点で、データベースのパフォーマンスを向上させる手法をいくつか説明します。
- 『環境についての考慮事項』では、データベース環境の設定時点で、データベースのパフォーマンスを向上させる手法をいくつか説明します。
- 『システム・カタログ統計』では、最適なパフォーマンスを達成するために、データについての統計を収集し使用する方法について説明します。
- 『SQL コンパイラーに関する解説』では、 SQL コンパイラーを使用してコンパイルしたときに、 SQL ステートメントがどうなるかについて説明します。
- 『SQL Explain 機能』では、Explain 機能について説明します。この機能により、データにアクセスするために SQL コンパイラーが行った選択を調べることができます。

システムのチューニングと構成

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

付録

- 『DB2 登録変数と環境変数』では、プロファイル・レジストリーの値と環境変数を示します。

- 『Explain 表と定義』では、 DB2 Explain 機能が使用する表と、それらの表の作成方法について説明します。
- 『SQL Explain ツール』では、 DB2 Explain ツールである db2expln および dynexpln の使用法について説明します。
- 『db2exfmt — Explain 表フォーマット・ツール』では、 DB2 Explain ツールを使用して Explain 表データをフォーマットする方法について説明します。

第 1 部 データベースの概念

第 1 章 基本的なリレーショナル・データベースの概念

データベース・オブジェクト

インスタンス:

インスタンス (データベース・マネージャー と呼ばれることがある) は、データを管理する DB2® コードです。データベース・マネージャーは、データに対して何ができるかを制御し、割り当てられたシステム・リソースを管理します。各インスタンスは、1 つの完全な環境です。そこには、特定の並列データベース・システムに定義された、すべてのデータベース・パーティションが含まれます。インスタンスは、(他のインスタンスがアクセスできない) 独自のデータベースを持っており、そのすべてのデータベース・パーティションは、同じシステム・ディレクトリーを共用します。またインスタンスは、同じマシン (システム) 上の他のインスタンスとは別個のセキュリティー規則を設定できます。

データベース:

リレーショナル・データベース は、データを表の集合として表します。表は、定義された数の列と任意の数の行によって構成されています。各データベースには、データの物理構造と論理構造を説明したシステム・カタログ表と、データベースに割り振られたパラメーター値が含まれる構成ファイル、進行中のトランザクションとアーカイブするトランザクションによるリカバリー・ログの集合が含まれる。

データベース・パーティション・グループ:

データベース・パーティション・グループ は、 1 つ以上のデータベース・パーティションのセットです。データベースに対して表を作成したい場合、まず、表スペースが保管される予定のデータベース・パーティション・グループを作成した後、表が保管される予定の表スペースを作成します。

DB2 UDB の以前のバージョンでは、データベース・パーティション・グループは、ノード・グループ として知られていました。

表スペース:

データベースは、表スペース と呼ばれるオブジェクトで構成されています。表スペースは、表を保管するスペースです。表を作成するときに、索引やラージ・オブジェクト (LOB) データなどの特定のオブジェクトを他の表データとは別にして保管することができます。表スペースは、1 つまたは複数の物理ストレージ・デバイスに分散させることもできます。次の図では、複数の表スペースに表のデータを分散させた状態を示しています。

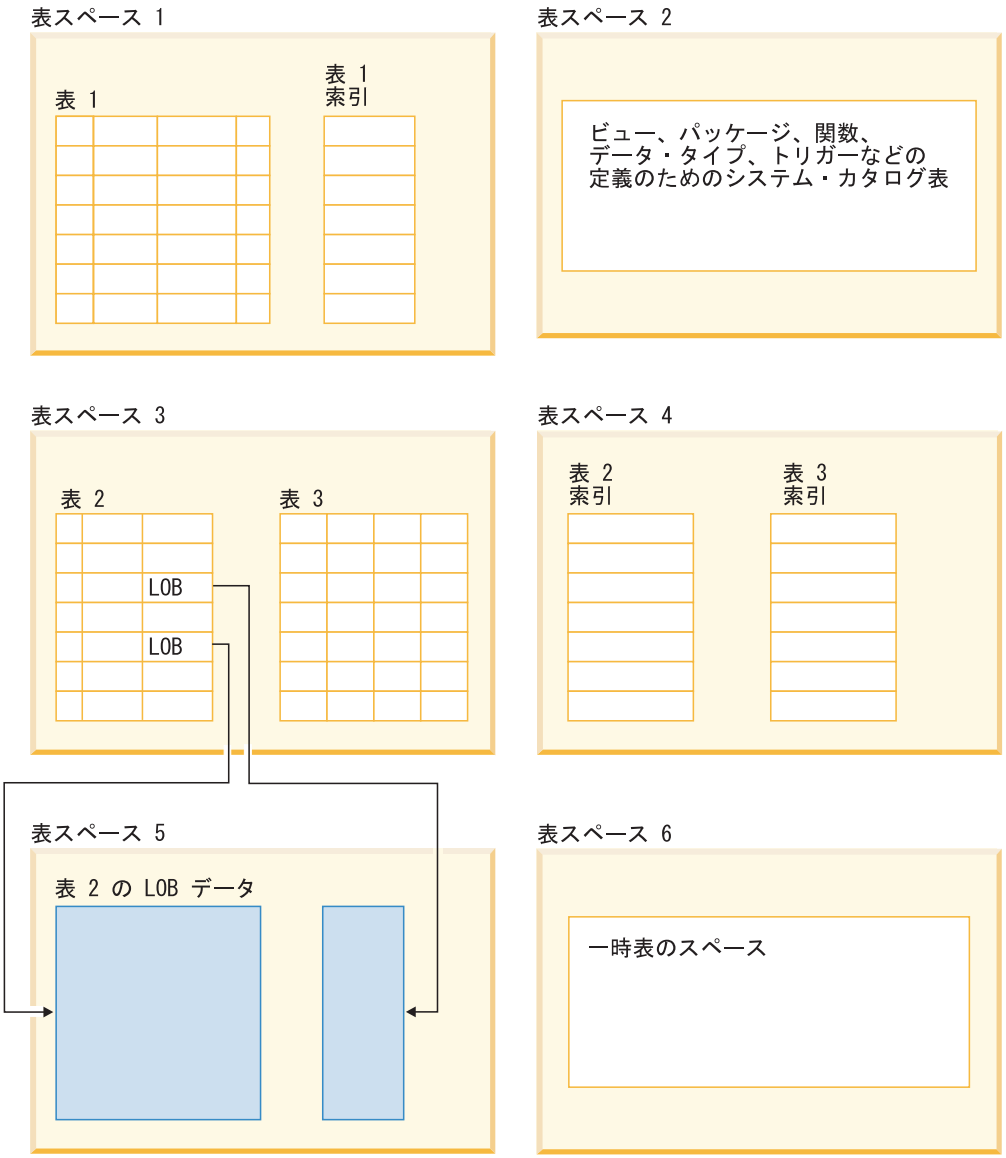

図 1. 複数の表スペースに表のデータを分散

表スペースは、データベース・パーティション・グループの中に配置します。表スペースの定義と属性は、データベース・システム・カタログに記録されています。

表スペースにはコンテナーが割り当てられています。コンテナー は、割り振られた物理ストレージ (ファイルまたはロー・デバイスなど) です。

表スペースとして、システム管理スペース (SMS) またはデータベース管理スペース (DMS) のどちらかを使用できます。 SMS 表スペースの場合、それぞれのコンテナーはオペレーティング・システムのファイル・スペースにあるディレクトリーであり、オペレーティング・システムのファイル・マネージャーがストレージを制御します。 DMS 表スペースの場合、それぞれのコンテナーはサイズが固定されている事前割り振りのファイルであるか、またはディスクなどの物理デバイスであり、データベース・マネージャーがストレージを制御します。

図 2 は、表、表スペース、および 2 つのタイプのスペース間のリレーションシップを示しています。この図はまた、表、索引、ロング・データが表スペースに保管されている様子も示しています。

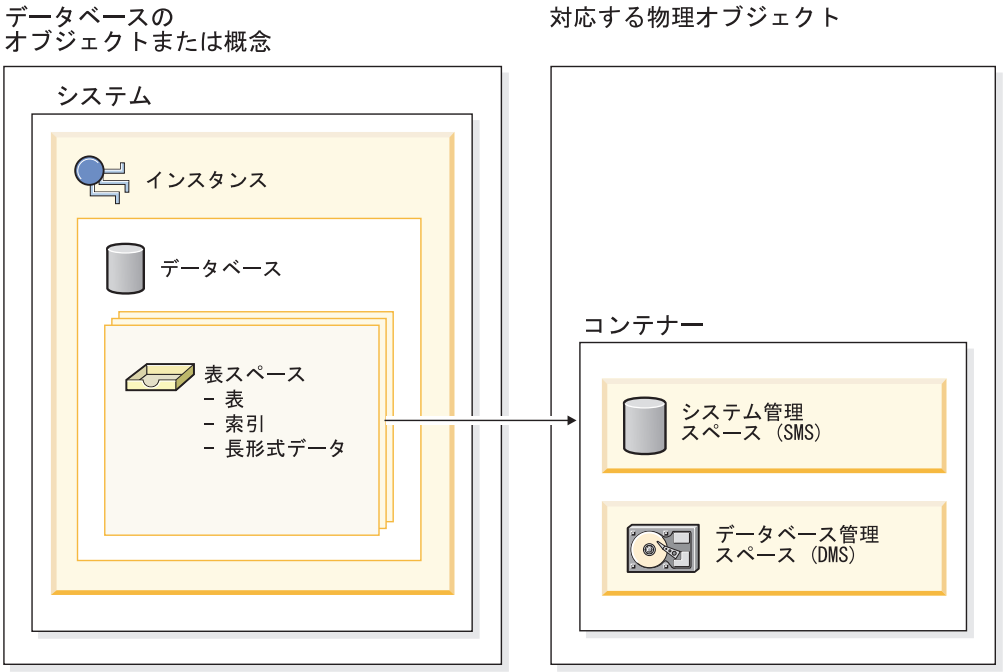

図 2. データを入れる表スペースおよびコンテナー・タイプ

6 ページの図 3 は、3 つの表スペース・タイプを示しています。これらは *REGULAR*、*TEMPORARY*、および *LARGE* です。

ユーザー・データを含む表は、REGULAR 表スペースにあります。デフォルトのユーザー表スペースは USERSPACE1 と呼ばれます。システム・カタログ表は、REGULAR 表スペースに存在します。デフォルトのシステム・カタログ表スペースは SYSCATSPACE と呼ばれています。

ロング・フィールド・データまたはラージ・オブジェクト・データを含む表、たとえばマルチメディア・オブジェクトなどは、LARGE 表スペースまたは REGULAR 表スペースに存在します。これらの列の基本列データは、REGULAR 表スペース内に保管されますが、ロング・フィールドまたはラージ・オブジェクト・データは、同じ REGULAR 表スペースまたは指定された LARGE 表スペース内に保管できます。

索引は、REGULAR 表スペースまたは LARGE 表スペース内に保管できます。

一時表スペース は、システム表またはユーザー表として分類されます。システム*一時表スペース* は、ソート、表の再編成、索引の作成、および表の結合などの SQL 操作中に必要となる内部一時データを保管するために使用します。システム一時表スペースはいくつでも作成できますが、大多数の表が使用するページ・サイズを使用して 1 つだけ作成することをお勧めします。デフォルトのシステム一時表スペースは TEMPSPACE1 と呼ばれています。ユーザー*一時表スペース* は、アプリケーシ

ョン一時データを保管する宣言済み一時表を保管するために使用されます。ユーザー一時表スペースは、デフォルトではデータベース作成の時点で作成されません。

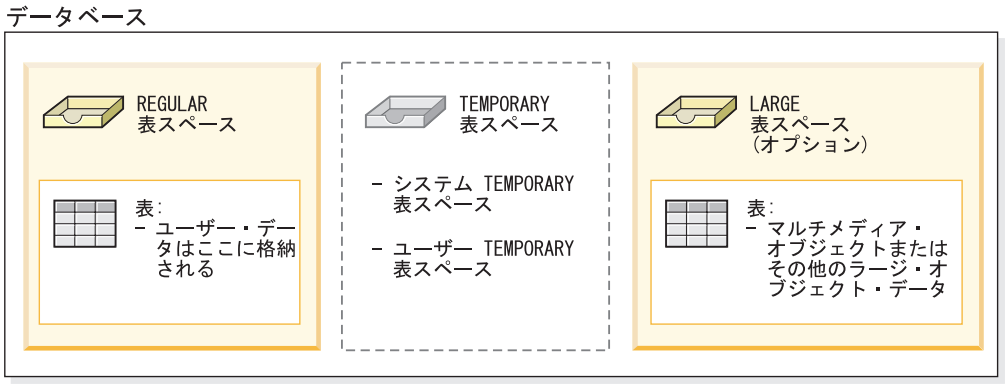

図 3. さまざまなタイプの表スペース

表:

リレーショナル・データベースは、データを表の集合として表します。*表* は、論理的に列と行に配列されたデータからなります。すべてのデータベースおよび表データは、表スペースに割り当てられます。表の中のデータは論理的に関連付けられ、そのリレーションシップは、複数の表の間で定義することができます。データは、*リレーション* (関係) と呼ばれる数学的な法則および操作に基づいて、表示または処理されます。

表データは、リレーショナル・データベース内のデータの定義および処理のための標準化された言語である構造化照会言語 (SQL) を使用してアクセスされます。*照会*は、データベースからデータを検索するために、複数のアプリケーション内または複数のユーザーによって使用されます。照会は、次のような SQL ステートメントを使用します。

```
SELECT <data_name> FROM <table_name>
```

ビュー:

ビュー は、データを保守せずに表すための効率的な方法です。ビューは実際の表ではなく、また永続ストレージを必要とすることもありません。「仮想表」が作成され、使用されます。

ビューには、ベースとなっている表の列または行のすべてまたは一部を含めることができます。たとえば、ビューの中で部署表と従業員表を結合して、特定の部署の従業員をすべてリストすることができます。

7 ページの図 4 は、表とビューの関連を示しています。

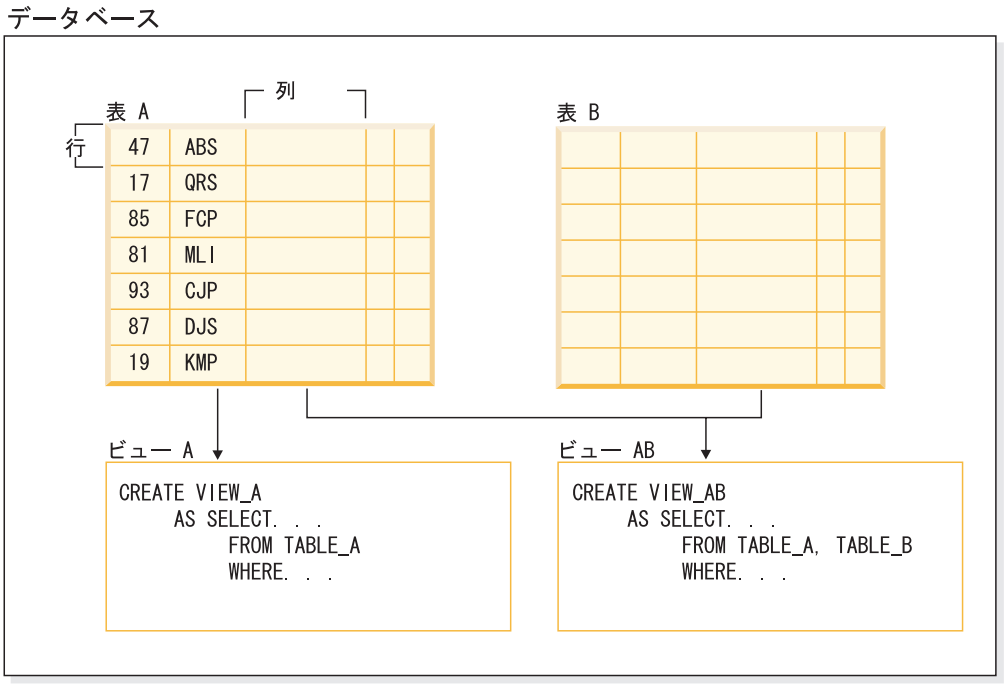

図 4. 表とビューの関係

索引:

索引 はキーの組み合わせであり、それぞれのキーは表の列に該当します。たとえば、 8 ページの図 5 の表 A には、表の従業員番号に基づいた索引があります。このキー値は、表の行を指すポインターの機能を提供します。従業員番号 19 は従業員 KMP を指します。索引では、ポインターを介して直接データへのパスを作成できるので、さらに効率よく表の行にアクセスできます。

SQL *オプティマイザー* は表のデータにアクセスする効率的な方法を自動的に選択します。オプティマイザーはデータへの一番速いアクセス・パスを判別するときに、索引を考慮に入れます。

ユニーク索引は、索引キーが必ずユニークになるようにするために作成できます。*索引キー* は、索引が定義されている列または列の順序付き集合です。ユニーク索引を使用すると、索引になっている列にある索引キーごとの値または列の値が必ずユニークなものとなります。

8 ページの図 5 は、索引と表のリレーションシップを示しています。

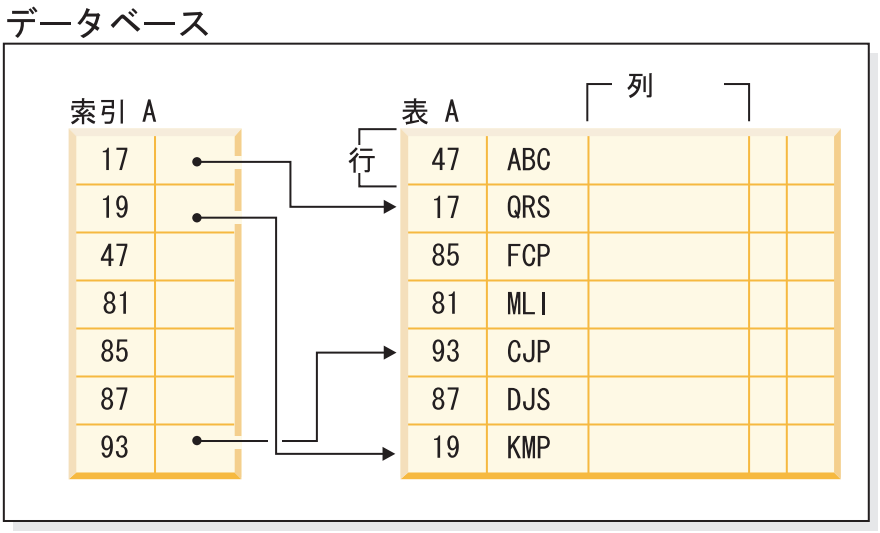

図 5. 索引と表の関係

図 6 では、データベース・オブジェクトのいくつかの関連を図示しています。この図はまた、表、索引、ロング・データが表スペースに保管されている様子も示しています。

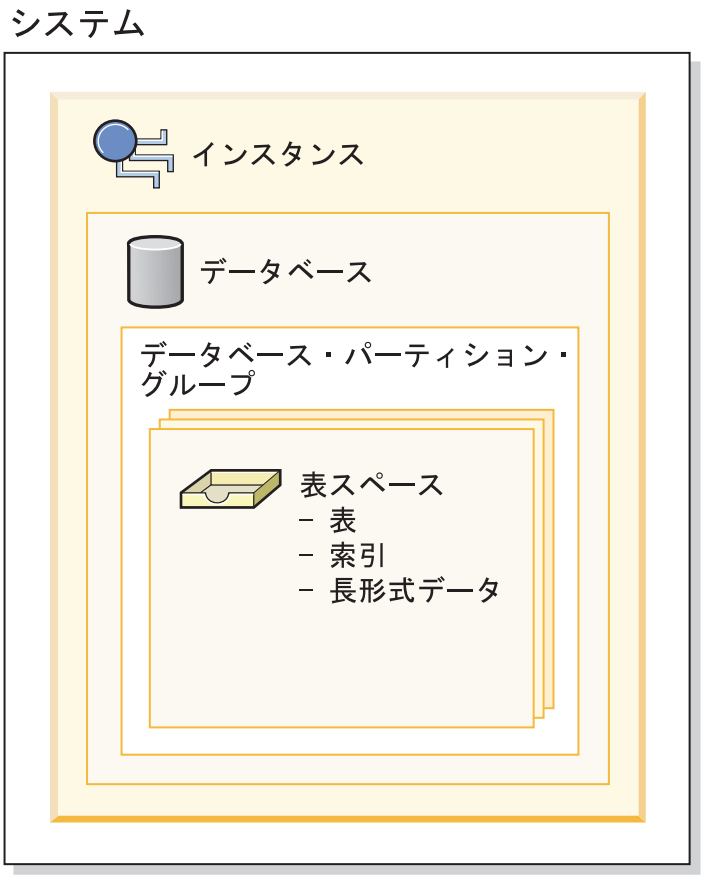

図 6. 一部のデータベース・オブジェクトの相互関係

スキーマ:

スキーマ はグループ表および他のデータベース・オブジェクトをグルーピングするユーザー ID などの ID です。スキーマは個人で所有することができ、所有者はデータおよびそのデータの中のオブジェクトへのアクセスを制御できます。

スキーマは、データベースのオブジェクトでもあります。スキーマは、スキーマ内の最初のオブジェクトが作成されるときに自動的に作成されます。そのようなオブジェクトとしては、スキーマ名によって修飾できるものであれば、どんなものでも使用できます。たとえば、表、索引、ビュー、パッケージ、特殊タイプ、関数、またはトリガーなどがあります。スキーマが自動的に作成されるには、IMPLICIT_SCHEMA 権限がなければなりません。そうでない場合は明示的にスキーマを作成します。

スキーマ名は 2 つの部分からなるオブジェクト名の最初の部分として使用されます。オブジェクトの作成時には、それを特定のスキーマに割り当てることができます。スキーマを指定しない場合、オブジェクトはデフォルトのスキーマに割り当てられます。これは通常はオブジェクトを作成した人のユーザー ID になります。名前の 2 番目の部分は、オブジェクトの名前になります。たとえば、Smith という名前のユーザーの表は SMITH.PAYROLL などとなります。

システム・カタログ表:

各データベースには、データの論理および物理構造の情報を格納する一連のシステム・カタログ表 が含まれます。 DB2 UDB は各データベースごとに一連のシステム・カタログ表を追加作成し、保守します。これらの表には、データベース・オブジェクト (たとえば、表、ビュー、および索引) の定義についての情報と、これらのオブジェクトに対してユーザーが持っている権限についてのセキュリティーの情報が含まれています。これらはデータベースの作成時に作成され、通常の操作中に更新されます。これらを明示的に作成したりドロップしたりすることはできませんが、カタログ・ビューを使用して内容の照会や表示を行うことは可能です。

コンテナー:

コンテナー は、物理ストレージ・デバイスです。これは、ディレクトリー名、デバイス名、またはファイル名によって識別できます。

1 つのコンテナーが 1 つの表スペースに対して割り当てられます。単一の表スペースはいくつものコンテナーにまたがることができますが、それぞれのコンテナーが属することができる表スペースは 1 つだけです。

10 ページの図 7 は、データベース内の表および表スペースと、それに関連しているコンテナーおよびディスクとのリレーションシップを示しています。

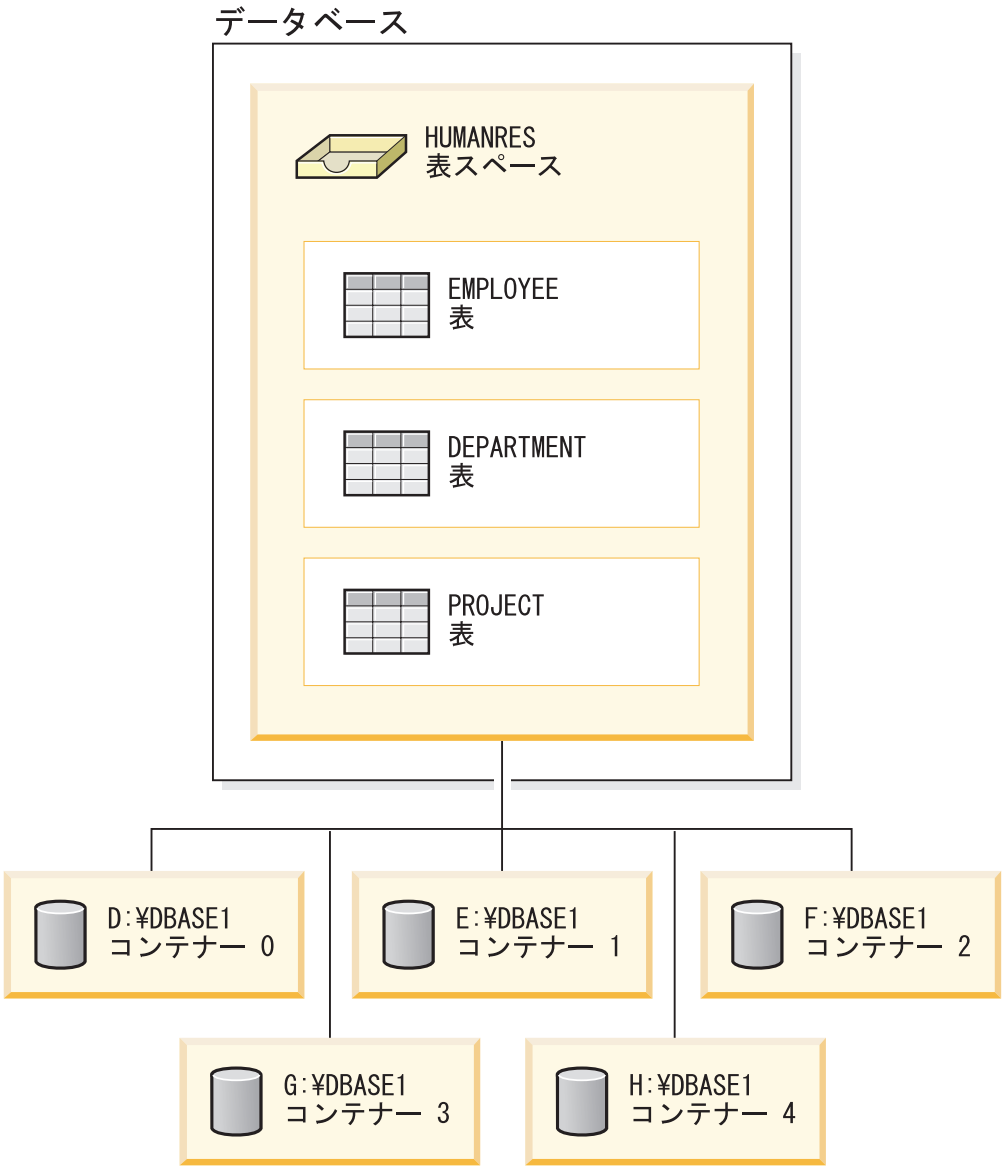

図 7. 表スペースとそのコンテナーの関係

EMPLOYEE、DEPARTMENT、および PROJECT 表は HUMANRES 表スペースにあり、これらはコンテナー 0、1、2、3、および 4 にまたがっています。この例では、それぞれのコンテナーは別々のディスクの中に存在しています。

表のデータは、表スペースにあるすべてのコンテナーにラウンドロビン方式で保管されます。このため、特定の表スペースに属するコンテナー間でデータが均等になります。別のコンテナーを使用する前に、データベース・マネージャーがコンテナーに書き込むページ数は、エクステント・サイズ と呼ばれます。

バッファー・プール:

バッファー・プール はメイン・メモリーの一部であり、ディスクからキャッシュ表または索引データ・ページが読み取られるか、または変更されるときにそれらに割り当てられます。バッファー・プールの目的は、システム・パフォーマンスを向上させることです。データへのアクセスは、ディスクよりもメモリーからの方がずっ

と速く行うことができます。したがって、データベース・マネージャーがディスクに対して読み書き (入出力) を行う回数が少ないほど、パフォーマンスは高くなります。 (複数のバッファー・プールを作成することもできますが、ほとんどの状況で必要になるバッファー・プールは 1 つだけです。)

バッファー・プールの構成によって入出力が遅いために生じる遅れを削減することができるため、これは、最も重要なチューニング対象の一つになります。

図 8 は、バッファー・プールとコンテナーのリレーションシップを示しています。

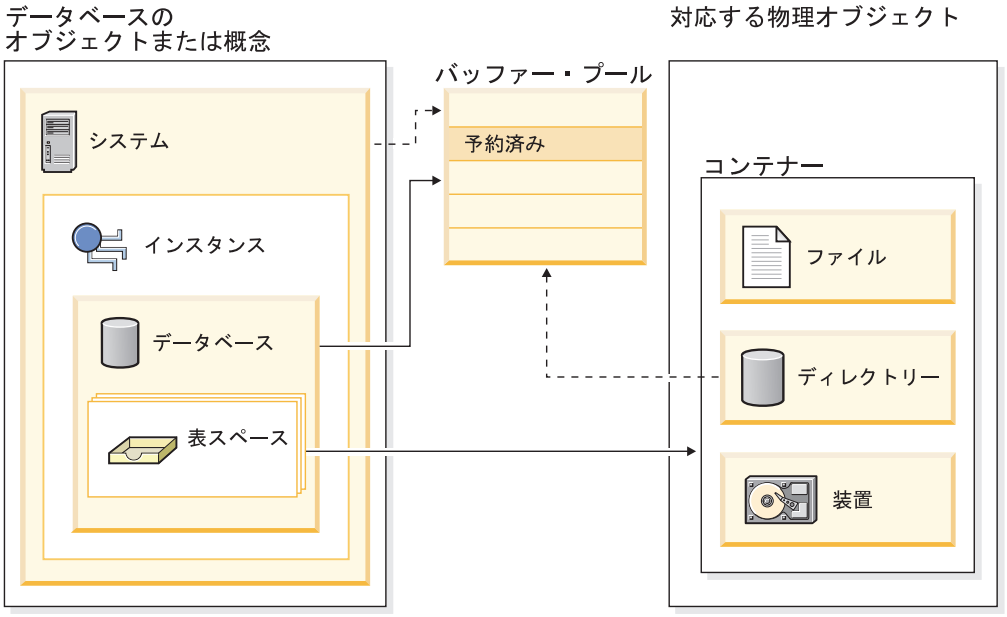

図 8. バッファー・プールとコンテナーの関係

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『索引』
- 「*SQL リファレンス 第 1 巻*」の『表』
- 「*SQL リファレンス 第 1 巻*」の『リレーショナル・データベース』
- 「*SQL リファレンス 第 1 巻*」の『スキーマ』
- 「*SQL リファレンス 第 1 巻*」の『ビュー』
- 「*SQL リファレンス 第 1 巻*」の『表スペースおよびその他のストレージ構造』

構成パラメーター

DB2 Universal Database™ インスタンスまたはデータベースが作成されると、対応する構成ファイルが作成され、そのパラメーター値はデフォルトになります。これらのパラメーター値を変更して、インスタンスまたはデータベースのパフォーマンスおよび他の特性を向上させることができます。

構成ファイル には、 DB2 UDB 製品および個々のデータベースに割り当てられているリソースや、診断レベルなどの値を定義するパラメーターが含まれています。構成ファイルには、以下の 2 つのタイプがあります。

- 各 DB2 UDB インスタンスのデータベース・マネージャー構成ファイル
- 個々のデータベースのデータベース構成ファイル

データベース・マネージャー構成ファイル は、 DB2 UDB インスタンスを作成するときに作成されます。これに含まれるパラメーターは、インスタンスの一部であるデータベースに関係なく、インスタンス・レベルでシステム・リソースに影響します。システムの構成によっては、これらのパラメーターの多くの値をシステム・デフォルト値から変更して、パフォーマンスを向上させたり容量を増やしたりすることができます。

また、それぞれのクライアント・インストールごとに、 1 つのデータベース・マネージャー構成ファイルがあります。このファイルには、特定のワークステーションのクライアント・イネーブラーに関する情報があります。サーバーに使用可能なパラメーターのサブセットは、クライアントにも適用できます。

データベース・マネージャー構成パラメーターは `db2systm` というファイルに保管されます。このファイルは、データベース・マネージャーのインスタンスの作成時に作成されます。 UNIX ベース環境では、このファイルはデータベース・マネージャーのインスタンスの `sqllib` サブディレクトリーにあります。 Windows では、このファイルのデフォルトのロケーションは、 `sqllib` ディレクトリーのインスタンス・サブディレクトリーです。 DB2INSTPROF 変数が設定されている場合、このファイルは、 DB2INSTPROF 変数によって指定された `instance` サブディレクトリー内にあります。

パーティション・データベース環境では、このファイルはファイル共有システムに置かれているので、すべてのデータベース・パーティション・サーバーが同じファイルへアクセスします。データベース・マネージャーの構成は、すべてのデータベース・パーティション・サーバーで同じです。

パラメーターのほとんどは、データベース・マネージャーの単一インスタンスへ割り振られるシステム・リソースの量に影響するか、または環境の考慮事項に基づいて、データベース・マネージャーおよびさまざまな通信サブシステムのセットアップを構成します。さらに、情報提供のみを目的とし、変更は行えないほかのパラメーターもあります。これらのパラメーターはすべて、データベース・マネージャーのインスタンスの下に保管されているどのような単一データベースとも関係なく、グローバルに適用可能です。

データベース構成ファイル は、データベースを作成するときに作成され、データベースと同じ場所に置かれます。データベースごとに 1 つの構成ファイルがあります。このパラメーターはとりわけ、データベースに割り当てられるリソースの量を指定します。パラメーターの多くの値を変更して、パフォーマンスを向上させたり容量を増やしたりすることができます。特定のデータベースでの活動のタイプによっては、別の変更が必要になることがあります。

個別のデータベースのパラメーターが、 `SQLDBCON` という構成ファイルに保管されます。このファイルは、データベースのほかの制御ファイルとともに `SQLnnnnn` ディレクトリーに保管されます。ここで、 `nnnnn` はデータベースの作成時に割り当て

られた番号です。データベースごとに独自の構成ファイルがあり、ファイル内のパラメーターのほとんどは、そのデータベースに割り当てられるリソースの量を指定します。このファイルには、記述情報のほかに、データベースの状況を示すフラグも含まれています。

パーティション・データベース環境では、データベース・パーティションごとに別個の SQLDBCON ファイルがあります。 SQLDBCON ファイルの値は、データベース・パーティションごとに同じでも、あるいは異なっていても構いませんが、データベース構成パラメーターの値はすべてのパーティションで同じにするようにお勧めします。

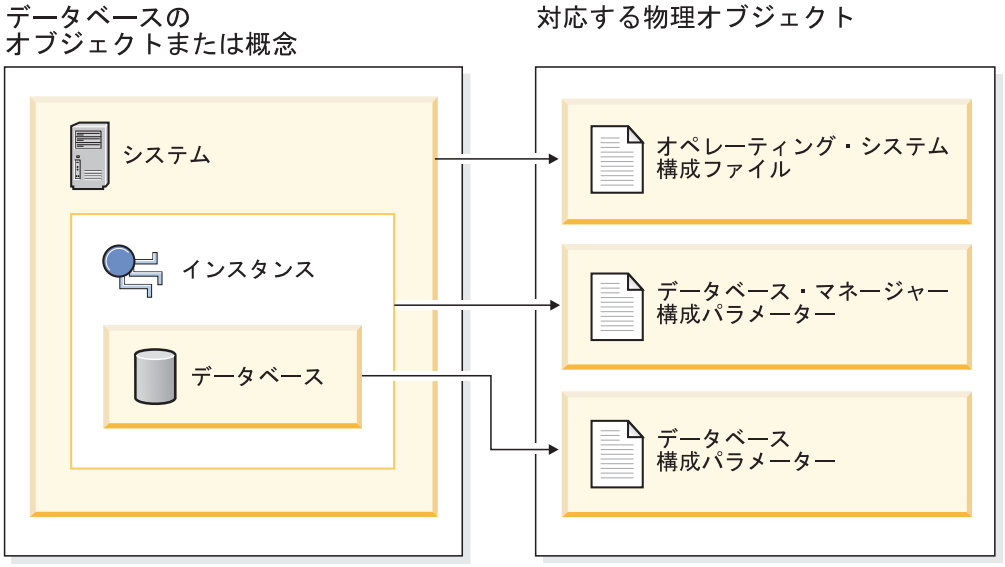

図 9. データベース・オブジェクトと構成ファイルの関係

関連概念:

- 「*管理ガイド: パフォーマンス*」の『構成パラメーターの調整』

関連タスク:

- 「*管理ガイド: パフォーマンス*」の『構成パラメーターによる DB2 の構成』

データに関するビジネス・ルール

どの業務でも、データが特定の制限または規則に従っていなければならない場合があります。たとえば、従業員番号がユニークでなければならない、などです。DB2® Universal Database (DB2 UDB) は、このような規則を構成する手段として*制約* を提供します。

DB2 UDB は、以下のタイプの制約を提供します。

- NOT NULL 制約
- ユニーク制約
- 主キー制約
- 外部キー制約
- チェック制約

- 情報制約

NOT NULL 制約

NOT NULL 制約は、NULL 値が列に入力されないようにします。

ユニーク制約

ユニーク制約は、一連の列にある値がユニークとなり、表のすべての行が非 NULL であるようにします。たとえば、部署表での典型的なユニーク制約では、部署番号がユニークかつ非 NULL でなければなりません。

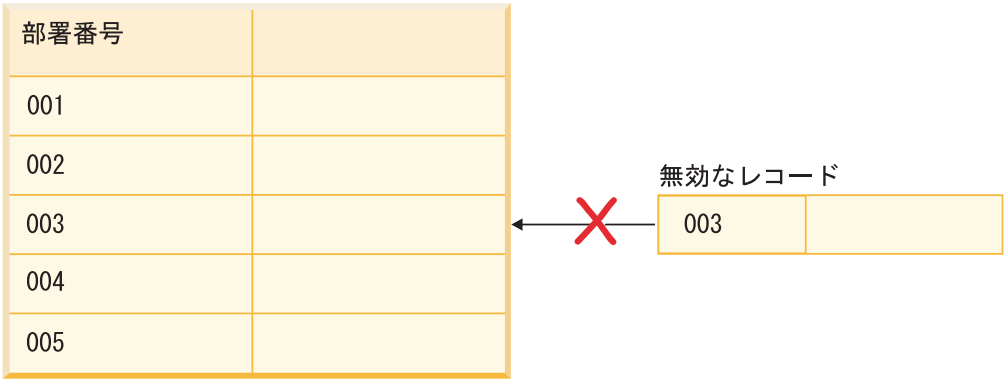

図 10. ユニーク制約は、データが重複しないようにするためのものです。

データベース・マネージャーは、挿入および更新操作中に制約を強制し、データ保全性を保証します。

主キー制約

それぞれの表は、1 つの主キーを持つことができます。主キーとは、ユニーク制約と同じプロパティーを持つ、列または列の組み合わせです。主キー制約と外部キー制約を使用して、表間のリレーションシップを定義できます。

主キーは表の行を識別するために使用するため、ユニークでなければならず、追加または削除は少なくなければなりません。 1 つの表は複数の主キーを持つことはできませんが、複数のユニーク・キーを持つことはできます。主キーはオプションであり、表の作成時または変更時に定義できます。これらには、データのエクスポート時または再編成時にデータを配列するという利点もあります。

次の表で、DEPTNO と EMPNO は、それぞれ DEPARTMENT 表と EMPLOYEE 表の主キーです。

表 1. DEPARTMENT 表

DEPTNO (主キー)	DEPTNAME	MGRNO
A00	Spiffy Computer Service Division	000010
B01	Planning	000020
C01	Information Center	000030
D11	Manufacturing Systems	000060

表 2. EMPLOYEE 表

EMPNO (主キー)	FIRSTNAME	LASTNAME	WORKDEPT (外部キー)	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O'Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

外部キー制約

外部キー制約 (参照保全制約ともいう) により、表間および表内での必須リレーションシップを定義することができます。

たとえば、典型的な外部キー制約は、従業員表の全従業員が、部署表に定義されているとおりに既存の部署のメンバーでなければならない、というものです。

このリレーションシップを確立するには、従業員表の部署番号を外部キーとして定義し、部署表の部署番号を主キーとして定義できます。

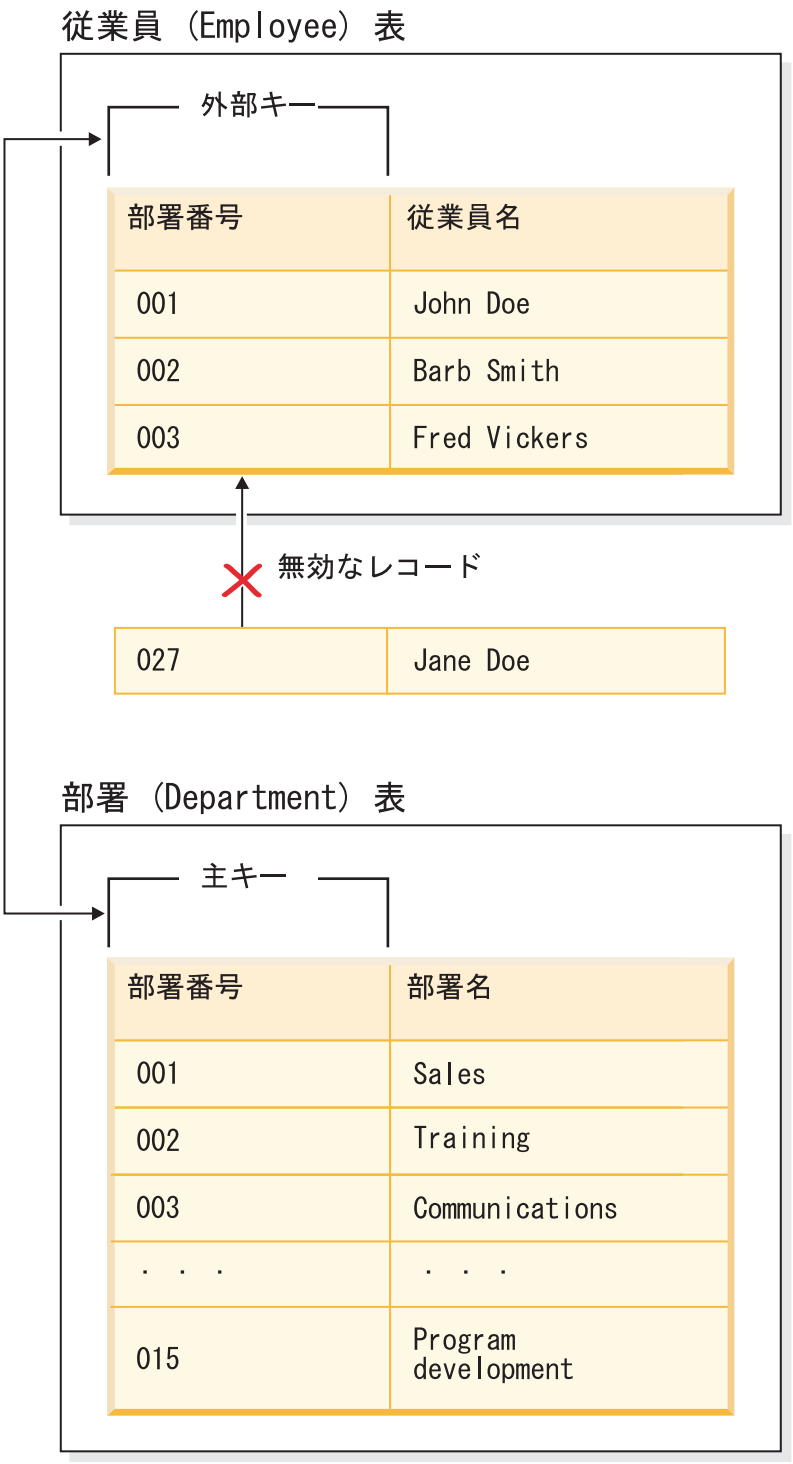

図 11. 外部キー制約と主キー制約

チェック制約

チェック制約は、表の各行にある 1 つまたは複数の列で許可される値を指定するためのデータベース規則です。

たとえば従業員表では、「ジョブのタイプ (Type of Job)」列を、「販売 (Sales)」、「管理 (Manager)」、または「事務 (Clerk)」となるように定義できます。この制約を使用すると、「ジョブのタイプ (Type of Job)」列にこ

れ以外の値のあるレコードはすべて無効になり、リジェクトされることにより、表で許可されているデータのタイプに関する規則が強制されます。

情報制約

情報制約は、SQL コンパイラーでは使用できるが、データベース・マネージャーでは適用されない規則です。この制約の目的は、データベース・マネージャーによるデータの付加的な検査を実施することではなく、照会のパフォーマンスを向上させることにあります。

情報制約は、CREATE TABLE または ALTER TABLE ステートメントを使用して定義されます。参照保全またはチェック制約を追加しますが、その後、それらに制約属性を関連付けて、データベース・マネージャーがその制約を適用するかどうか、またその制約が照会の最適化のために使用されるかどうかを指定します。

トリガー をデータベースで使用することもできます。トリガーは制約より複雑かつ強力な定義を行うことが可能です。トリガーは、指定した基本表に対する INSERT、 UPDATE、DELETE 文節と一緒に実行される一連のアクション、またはそれらの文節によってトリガー起動される一連のアクションを定義するものです。トリガーを使えば、一般的な保全規則や業務規則 (ビジネス・ルール) をサポートできます。たとえば、トリガーによって、注文に応じる前に顧客のクレジット限度を調べたり、銀行業務アプリケーションで使用して、アカウントからの引き出しが顧客の標準的な引き出しパターンに合わなかった場合にアラートを立ち上げたりできます。

関連概念:

- 70 ページの『制約』
- 75 ページの『トリガー』

バックアップおよびリカバリー・ストラテジーの開発

データベースはハードウェア障害またはソフトウェア障害 (あるいはその両方) が原因で使用不能になることがあります。ストレージ問題、電源遮断、およびアプリケーション障害が生じることがありますが、障害が異なれば異なったリカバリー処置が必要になります。十分にリハーサルされたリカバリー・ストラテジーを適所に持つことによって、障害の可能性からご使用のデータを保護してください。リカバリー・ストラテジーを開発するときに応答が必要な質問のいくつかは、以下のとおりです。

- データベースはリカバリー可能ですか?
- データベースをリカバリーするためにどのくらいの時間を費やすことができますか?
- バックアップ操作間にどのくらいの時間を費やすことができますか?
- バックアップ・コピーおよびアーカイブ・ログのためにどのくらいのストレージ・スペースを割り振ることができますか?
- 表スペース・レベルのバックアップで十分ですか? それともデータベース全体のバックアップが必要ですか?
- スタンバイ・システムを構成する必要がありますか (手動リカバリーまたは高可用性災害時リカバリー (HADR) のどちらにするか)?

データベース・リカバリー・ストラテジーは、データベース・リカバリーが必要なときにすべての情報が使用可能であることを確認する必要があります。データベース・リカバリー・ストラテジーには、データベースのバックアップをとる日常のスケジュール、およびパーティション・データベース・システムの場合、システムが拡張されるとき (データベース・パーティション・サーバーまたはノードが追加またはドロップされるとき) のバックアップが含まれている必要があります。また、全体的なストラテジーには、コマンド・スクリプト、アプリケーション、ユーザー定義関数 (UDF)、オペレーティング・システム・ライブラリー内のストアード・プロシージャー・コード、およびロード時に作成されるコピー・ファイルも含まれている必要があります。

続くセクションでは、異なるリカバリー・メソッドについて説明されており、ご使用の業務環境にどのリカバリー・メソッドが最善であるかが分かるでしょう。

データベース・バックアップ の概念は、他のデータ・バックアップの概念と同じです。つまり、オリジナルで障害または損傷が起こる場合のために、データのコピーを取り、異なるメディアに保管します。一番単純なバックアップでは、データベースをシャットダウンして、トランザクションがこれ以上生じないようにしてから、単純にそのバックアップを取ります。そのようにすると、いくつかの方法でデータベースが損傷を受けたりまたは破壊される場合に、データベースを再作成することができます。

このデータベースの再作成のことを*リカバリー* といいます。*バージョン・リカバリー* は、以前のバージョンのデータベースのリカバリーであり、バックアップ操作で作成されたイメージを使用して行われます。*ロールフォワード・リカバリー* は、データベースまたは表スペースのバックアップ・イメージがリストアされた後にデータベース・ログ・ファイル内に記録されるトランザクションの再適用を行います。

クラッシュ・リカバリー は、 1 つ以上の単位の作業 (トランザクション) の一部であるすべての変更が完了およびコミットする前に障害が起こる場合のデータベースの自動リカバリーです。これは、未完了のトランザクションをロールバックし、故障発生時にメモリーに残っていたコミット済みトランザクションを完了することによって行われます。

リカバリー・ログ・ファイルおよびリカバリー履歴ファイルは、データベースの作成時に自動的に作成されます (19 ページの図 12)。これらのログ・ファイルは、脱落または損傷したデータのリカバリーが必要な場合に重要です。

それぞれのデータベースには、*リカバリー・ログ* があります。このログは、アプリケーション・エラーやシステム・エラーからのリカバリーに使用されます。データベースのバックアップと組み合わせて使用すれば、これらのログから、エラーが発生した時点までデータベースの整合性をリカバリーさせることが可能です。

リカバリー履歴ファイル には、指定した時点までデータベースのすべてまたは一部をリカバリーする必要のある場合に使用できる、バックアップ情報のサマリーが含まれています。これはバックアップおよびリストア操作などのリカバリー関連のイベントを追跡するために使用します。このファイルは、データベース・ディレクトリー内にあります。

表スペース変更履歴ファイル (データベース・ディレクトリー内にもある) には、特定の表スペースのリカバリーに必要なログ・ファイルを判別するために使用可能な情報が含まれています。

リカバリー履歴ファイルまたは表スペース変更履歴ファイルを直接変更することはできません。ただし、PRUNE HISTORY コマンドを使用して、ファイルから項目を削除することは可能です。 *rec_his_retentn* データベース構成パラメーターを使用して、これらの履歴ファイルを保存する日数を指定することもできます。

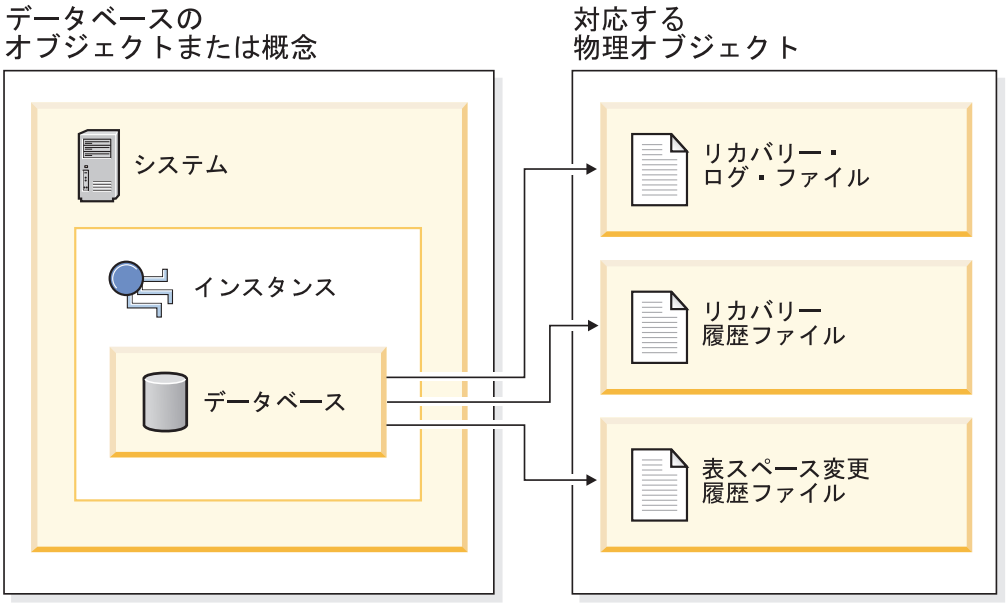

図 12. データベース・リカバリー・ファイル

簡単に再作成が可能なデータはリカバリー不能データベース内に保管できます。これには、読み取り専用アプリケーションに使用される外部ソースからのデータおよび頻繁に更新されない表が含まれており、少量のロギングであれば、ログ・ファイルの管理やリストア操作後のロールフォワードをさらに複雑化することはありません。*リカバリー不能データベース* は、 *logarchmeth1* および *logarchmeth2* データベース構成パラメーターが 「OFF」に設定されています。これは、保持されているログのみがクラッシュ・リカバリーに必要なログであることを意味しています。これらのログはアクティブ・ログとして知られており、現行のトランザクション・データが含まれています。オフライン・バックアップを使用したバージョン・リカバリーは、リカバリー不能データベースのリカバリーの基本的な方法です。 (オフライン・バックアップは、バックアップ操作が進行中にその他のアプリケーションがデータベースを使用できないことを意味しています。) そのようなデータベースは、オフラインのみをリストアできます。データベースは、バックアップ・イメージがとられ、ロールフォワード・リカバリーがサポートされないときの状態にリストアされます。

簡単に再作成できない データは、リカバリー可能データベース内に保管する必要があります。これには、データのロード後にソースが破壊されるデータ、表に手動で入力されるデータ、およびデータベースにロード後にアプリケーション・プログラムまたはユーザーによって変更されるデータが含まれます。*リカバリー可能データベース*は、 *logarchmeth1* または *logarchmeth2* データベース構成パラメーターが

「OFF」以外の値に設定されています。アクティブ・ログは、クラッシュ・リカバリーのために使用可能ですが、アーカイブ・ログ も持っています。これには、コミットされたトランザクション・データが含まれます。そのようなデータベースは、オフラインのみをリストアできます。データベースは、バックアップ・イメージがとられたときの状態にリストアされます。ただし、ロールフォワード・リカバリーでは、特定の時点 (つまり、バックアップ・イメージがとられた時点を通過した) またはアクティブ・ログの終わりのいずれかにアクティブおよびアーカイブ・ログを使用することにより、データベースをロールフォワード することができます。

リカバリー可能データベース・バックアップ操作は、オフラインまたはオンライン (オンラインとは、バックアップ操作中にその他のアプリケーションがデータベースに接続できることを意味している) のいずれかで実行できます。オンライン表スペースおよびロールフォワード操作は、データベースがリカバリー可能な場合のみサポートされます。データベースがリカバリー不能な場合、データベース・リストアおよびロールフォワード操作はオフラインで実行する必要があります。オンライン・バックアップ操作の際、ロールフォワード・リカバリーは、バックアップがリストアされる場合に、すべての 表の変更がキャプチャーされ、再適用されることを確認します。

リカバリー可能データベースを持っている場合、データベース全体ではなく、個々の表スペースをバックアップ、リストア、およびロールフォワードすることができます。表スペースをオンライン・バックアップするとき、表スペースは引き続き使用可能であり、同時更新がログ内に記録されます。表スペースでオンライン・リストアまたはロールフォワード操作を実行するとき、操作が完了するまで表スペース自体は使用できませんが、ユーザーはその他の表スペース内の表にアクセスすることができます。

自動バックアップ操作

バックアップ操作などの保守アクティビティーを実行するかどうか、およびいつ実行するかの決定には時間がかかるため、「自動保守の構成」ウィザードを使用してこれを行うことができます。自動保守では、自動保守をいつ実行するかなどの保守の目的を指定します。それによって、DB2 はそれらの目的に応じて、保守アクティビティーを実行する必要があるかどうかを判別し、次に保守を実行できる時間帯 (自動保守アクティビティーを実行するためのユーザー定義の時間帯) に必要な保守アクティビティーだけを実行します。

注: 自動保守を設定した場合でも、手動バックアップ操作は実行できます。 DB2 は必要な場合だけ、自動バックアップ操作を実行します。

関連概念:

- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『クラッシュ・リカバリー』
- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『バージョン・リカバリー』
- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『ロールフォワード・リカバリー』
- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『高可用性災害時リカバリーの概要』

- 「*DB2 Data Links Manager 管理ガイドおよびリファレンス*」の『データ・リンク・サーバー・ファイルのバックアップ』
- 「*DB2 Data Links Manager 管理ガイドおよびリファレンス*」の『障害とリカバリーの概要』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『rec_his_retentn - 「リカバリー履歴保存期間」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logarchmeth1 - 1 次ログ・アーカイブ方式構成パラメーター』
- 「*DB2 Data Links Manager 管理ガイドおよびリファレンス*」の『DB2 Data Links Manager システムのセットアップとバックアップに関する推奨事項』

自動保守

DB2® Universal Database (UDB) には、データベースのバックアップを実行したり、統計データを最新の状態に維持したり、必要に応じて表および索引を再編成したりするためのさまざまな自動保守機能が含まれています。

自動データベース・バックアップの機能を使用すると、バックアップのタイミングについて悩んだり、バックアップ・コマンドの知識を必要としたりすることなく、適切かつ定期的にデータベースのバックアップを取ることができます。

自動統計収集機能は、表に関する統計データを最新のものに維持することにより、データベースのパフォーマンスの向上を図ります。その目的は、オプティマイザーが正確な統計データに基づいてアクセス・プランを選択できるようにすることです。

自動統計プロファイルは、統計データのうち古くなったもの、欠落しているもの、そして指定が正しくないものを検出したり、照会フィードバックに基づいて統計プロファイルを生成したりすることによって、表統計データをいつどのように収集するかに関するアドバイスを提供します。

自動再編成機能は、表および索引のオフライン再編成を管理し、ユーザーがデータをいつどのように再編成するかを心配しなくてすむようにします。

自動保守の機能を有効にするかどうかは、自動保守データベース構成パラメーターを使用することによって制御します。それらのパラメーターは、自動保守機能を有効にするかどうかの管理を簡単かつ柔軟にするための、さまざまなスイッチの階層です。

自動データベース・バックアップ:

データベースは、ハードウェアまたはソフトウェアのさまざまな障害のために使用不可能になることがあります。自動データベース・バックアップの機能を使用すると、データベースの最新のフル・バックアップが必要に応じて確実に実行されるようになるため、 DBA がデータベース・バックアップを簡単に管理できるようになります。バックアップ操作の実行が必要かどうかは、次のことを基準にして判断されます。

- データベースのフル・バックアップを一度も実行したことがない
- 最後に実行したフル・バックアップ以降に経過した時間が、指定された時間を超えた
- 最後に実行されたバックアップ以降に消費されたトランザクション・ログ・スペースが指定された (4 KB) ページ数を超えた (アーカイブ・ロギング・モードの場合のみ)。

システムの災害時リカバリー・ストラテジーの計画を立ててインプリメントすることによって、データを保護してください。実際の必要によっては、自動データベース・バックアップ機能を、バックアップとリカバリーのストラテジーの一部として含めることもできます。

データベースがロールフォワード・リカバリー (アーカイブ・ロギング) 対応なら、オンライン・バックアップかオフライン・バックアップのいずれかについて、自動データベース・バックアップを有効にすることができます。そうでない場合に利用できるのは、オフライン・バックアップだけです。自動データベース・バックアップでは、ディスク、テープ、 Tivoli® Storage Manager (TSM)、およびベンダー固有の DLL メディア・タイプがサポートされています。

コントロール・センターまたはヘルス・センターの自動保守構成ウィザードを使用すると、次のものを構成できます。

- 前回のバックアップからの経過時間、または数のログ・ページ数
- バックアップ・メディア
- オンライン・バックアップか、それともオフライン・バックアップか

ディスクへのバックアップを選択した場合、自動バックアップ機能により、自動保守構成ウィザードで指定されたディレクトリーからバックアップ・イメージが定期的に削除されます。どの時点においても、最新のバックアップ・イメージだけが使用可能になります。そのディレクトリーは、自動バックアップ機能専用にし、他のバックアップ・イメージを格納するためには使用しないようにすることをお勧めします。

自動データベース・バックアップ機能を有効または無効にするには、**auto_db_backup** および **auto_maint** のデータベース構成パラメーターを使用します。複数データベース・パーティション環境においては、それらのデータベース構成パラメーターが有効になっているパーティションのそれぞれに対して、自動データベース・バックアップが実行されます。

自動統計収集:

SQL コンパイラーが SQL 照会プランを最適化する際には、データベースの表と索引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。さらに、表および索引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。オプティマイザーはその情報を使用することにより、各照会の代替アクセス・プランのコストを見積もります。表に対して追加または削除する行の数がかなり多い場合、または統計情報収集の対象となる列のデータが更新された場合には、 RUNSTATS ユーティリティーを再実行することによって、統計情報を更新する必要があります。

自動統計収集は、パフォーマンスが最高になる統計情報の最小セットを判別することにより動作します。統計情報を収集または更新するかどうかという決定は、表の変更頻度や表統計情報の変更の程度を監視することによってなされます。自動統計収集のアルゴリズムは、時間の経過と共に、表ごとに統計情報がどの程度の速さで変化するかを学習し、それに応じて内部で RUNSTATS の実行をスケジューリングします。

ユーザーが RUNSTATS や REORG を実行したり、表を変更またはドロップしたりするなどのデータベースの通常の保守活動は、この機能を有効かどうかによる影響を受けません。

データベースの表に関する統計情報をどの程度の頻度で収集するかがよくわからない場合には、データベース保守計画全体の一部として自動統計収集機能を含めることができます。

自動統計収集機能を有効または無効にするには、 **auto_runstats**、**auto_tbl_maint**、および **auto_maintdatabase** の構成パラメーターを使用します。

複数データベース・パーティション環境の場合、自動統計収集を実行するかどうかの決定や自動統計収集の開始は、カタログ・パーティション上でなされます。**auto_runstats** 構成パラメーターを有効にすることが必要なのは、カタログ・パーティションにおいてだけです。実際の統計収集は RUNSTATS によって実行され、次のように収集されます。

1. カタログ・パーティションに表データが含まれているなら、そのカタログ・パーティション上の統計情報が収集されます。表データが含まれているパーティションにおいて RUNSTATS が開始された場合、そのパーティションの統計情報は常に収集されます。
2. それ以外の場合、統計収集は、表パーティション・リストの最初のパーティションに対してなされます。

自動統計収集を考慮した表は、コントロール・センターまたはヘルス・センターから自動保守ウィザードを使用することによって構成できます。

自動統計プロファイル:

統計情報がない場合や古い場合には、オプティマイザーが遅い照会プランを選択する可能性があります。特定のワークロードに対してすべての統計情報が重要な意味を持つとは限らないという点に注意することは重要です。たとえば、どの照会述部にも出現しない列に関する統計情報には、普通、何の影響もありません。列と列の間の相関に関する調整のために、複数の列に関する統計情報 (列グループ統計) が必要になる場合があります。

自動統計プロファイルでは、前回の照会で使用された列だけを考慮し、さらに見積もり誤差が発生した列または列の組み合わせを認識することによって、オプティマイザーの動作を分析します。誤差を検出して統計プロファイルを推奨または変更するため、統計プロファイル・ジェネレーターは、照会実行時に蓄積された情報に加えて、照会のコンパイル時に収集された情報も調べます。照会が調べられ、プランが選択および実行された後でアクションが実行されると、このアプローチはそれに反作用します。

自動統計プロファイルは、統計データのうち古くなったもの、欠落しているもの、そして指定が正しくないものを検出したり、照会フィードバックに基づいて統計プロファイルを生成したりすることによって、 RUNSTATS ユーティリティーを使用した統計情報収集をどのように実行するかに関するアドバイスを提供します。

実際の必要によっては、自動統計プロファイル機能を、データベース保守プランの一部として含めることもできます。

自動統計プロファイルは、自動統計収集機能との相互作用により、統計情報の収集のタイミングに関するアドバイスを提供します。

自動統計プロファイル機能を有効または無効にするには、 **auto_stats_prof**、**auto_tbl_maint**、および **auto_maintdatabase** の構成パラメーターを使用します。**auto_prof_upd** データベース構成パラメーターも有効になっている場合には、生成される統計プロファイルを使用して RUNSTATS ユーザー・プロファイルが更新されます。自動統計プロファイルは、複数データベース・パーティション環境では無効です。また、対照型マルチプロセッサー (SMP) 並列処理 (パーティション内並列処理) が有効な場合も、自動統計プロファイルは無効です。

自動再編成:

表データに対して数多くの変更がなされると、論理的には連続しているデータが、連続していない物理ページ上に格納され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。

情報の中でも RUNSTATS によって収集される統計情報には、表の中のデータ分布が示されます。特に、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ていることがあります。自動再編成は、REORGCHK のさまざまな公式を使用することによって、表の再編成が必要かどうかを判定します。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が定期的に評価されます。再編成が必要なら、表の古典的な再編成が内部でスケジュールに入れられます。そのためには、再編成する表に対する書き込みアクセス権なしでアプリケーションが機能しなければなりません。

自動再編成機能を有効または無効にするには、 **auto_reorg**、**auto_tbl_maint**、および **auto_maint** のデータベース構成パラメーターを使用します。

複数データベース・パーティション環境の場合、自動再編成を実行するかどうかの決定や自動再編成の開始は、カタログ・パーティション上でなされます。データベース構成パラメーターを有効にすることが必要なのは、カタログ・パーティションにおいてだけです。再編成は、ターゲット表の存在しているすべての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。

自動再編成を考慮した表は、コントロール・センターまたはヘルス・センターから自動保守ウィザードを使用することによって構成できます。

自動化のための保守ウィンドウ:

前述の自動保守機能はシステム上のリソースを消費するため、それを実行すると、データベースのパフォーマンスに影響する可能性があります。自動再編成とオフライン・データベース・バックアップを実行すると、それらのユーティリティーの実行時に表とデータベースへのアクセスも制限されます。したがって、それらの保守活動の DB2 UDB での実行を内部でスケジュールに入れることが可能な、適切な期間を提供することが必要です。それらは、コントロール・センターまたはヘルス・センターから自動保守ウィザードを使用することによって、オフラインおよびオンラインの保守期間として指定できます。

オフライン・データベース・バックアップと、表および索引の再編成は、オフライン保守期間中に実行されます。それらの機能は、指定された期間を超えるとしても完了するまで実行されます。内部スケジューリング・メカニズムは、時間経過と共に学習し、ジョブの完了時刻を見積もります。オフライン期間が短すぎるために、特定のデータベース・バックアップまたは再編成活動を実行できない場合には、その次の期間でスケジューラーはそのジョブを開始せず、ヘルス・モニターを通じてオフライン保守期間を長くすることが必要であるということを通知します。

自動統計収集およびプロファイル、およびオンライン・データベース・バックアップは、オンライン保守期間中に実行されます。システムへの影響を最小限に抑えるため、それらは最適なユーティリティー・スロットル・メカニズムによってスロットルされます。内部スケジューリング・メカニズムでは、オンライン保守期間を使用することによりオンライン・ジョブが開始されます。それらの機能は、指定された期間を超えるとしても完了するまで実行されます。

ストレージ:

自動統計収集および再編成の機能は、作業データをデータベース中の表に格納します。それらの表は、SYSTOOLSPACE 表スペースの中に作成されます。SYSTOOLSPACE 表スペースは、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 として計算する必要があります。それがデータベースの重大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とよいでしょう。表スペースの中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。表スペースがドロップされると、それらの表にキャプチャーされた履歴は失われます。

モニターと通知:

ヘルス・モニターには、自動データベース・バックアップ、統計収集、および再編成の機能のモニターおよび通知の機能が含まれています。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『カタログ統計』
- 「*管理ガイド: パフォーマンス*」の『表の再編成』
- 「*管理ガイド: パフォーマンス*」の『標準表における表および索引の管理』
- 17 ページの『バックアップおよびリカバリー・ストラテジーの開発』
- 「*管理ガイド: パフォーマンス*」の『MDC 表のための表および索引管理』
- 「*システム・モニター ガイドおよびリファレンス*」の『ヘルス・モニター』
- 「*管理ガイド: パフォーマンス*」の『自動統計収集』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『autonomic_switches - 自動保守スイッチ構成パラメーター』

高可用性災害時リカバリー (HADR) フィーチャーの概要

DB2® Universal Database (DB2 UDB) の高可用性災害時リカバリー (HADR) は、サイトの部分的障害または全体的障害のための高可用性ソリューションを提供するデータベース・レプリケーション・フィーチャーです。 HADR は、ソース・データベース (主) からターゲット・データベース (スタンバイ) にデータ変更を複製する際に、データ損失が発生しないよう保護します。

部分サイト障害は、ハードウェア、ネットワーク、またはソフトウェア (DB2 UDB またはオペレーティング・システム) の障害によって発生することがあります。 HADR を使用しない場合には、データベース管理システム (DBMS) サーバーまたはデータベースが含まれているマシンをリブートするか、または再始動することが必要になります。その場合、完了するまでに数分間かかることがあります。 HADR を使用すれば、数秒間のうちにスタンバイ・データベースが、主データベースの役割を引き継ぎます。

サイト全体の障害は、火災などの災害のためにサイト全体が破壊された場合に発生することがあります。 HADR では、主データベースとスタンバイ・データベースの間の通信に TCP/IP を使用するため、 2 つのデータベースは異なる場所に配置できます。たとえば、主データベースをある都市にある本社に配置し、スタンバイ・データベースを別の都市にある営業所に配置することが可能です。主サイトで災害が発生した場合、リモートのスタンバイ・データベースが主データベースの DB2 UDB の機能をすべて引き継ぐようにすることによって、データ使用可能性が維持されます。引き継ぎ処理の後、オリジナルの主データベースのバックアップを復元してから、それを主データベースの現行の状況に戻すことができます。この操作は、*フェイルバック* と呼ばれます。

HADR を使用すると、潜在的なデータ損失からの保護レベルを選択できます。そのためには、同期 (SYNC)、近同期 (NEARSYNC)、および非同期 (ASYNC) の 3 種類の同期モードのうち 1 つを指定します。それらのモードは、2 つのシステムの間でのデータ変更の伝搬方法を指定するものです。選択する同期モードは、スタンバイ・データベースがレプリカとして主データベースにどれだけ近いものになるかを決めるものとなります。たとえば、同期モードを使用することにより HADR は、主データベースでコミットされたトランザクションがスタンバイでも確実にコミットされるようにすることができます。

同期により、2 つのシステムの間のフェイルオーバーとフェイルバックの機能が可能になります。

データの変更内容は、主システムからスタンバイ・システムに送られるデータベース・ログ・レコードに記録されます。 HADR は、DB2 UDB ロギングおよびリカバリーと密結合されています。

HADR では、2 つのシステムのハードウェア、オペレーティング・システム、および DB2 UDB ソフトウェアが同じであることが必要です。 (システムのアップグレード中に若干の相違が発生する可能性はあります。)

HADR スタンバイ・データベースは、それら主データベースのバックアップからリストアすることによって、または主データベースのスプリット・ミラー・コピーから初期化することによって確立されます。 HADR が起動すると、スタンバイ・データベースは主データベースからログ・レコードを取り出し、それをデータベースのスタンバイ側コピーに対して再生します。ログ・レコードは、スタンバイ・データベースが主データベースのメモリー内のログ・セットに「追いつく」まで、スタンバイ・データベースに適用されます。その時点で HADR の対は PEER 状態になります。つまり、主データベースは、新しいログ・ページをローカル・ディスクに書き込むだけでなく、スタンバイ・データベースにも送ります。それらのログ・ページがスタンバイ・データベースに到着すると、ただちに再生されます。継続的なログの再生により、スタンバイ・データベースは、主データベースの時間的に少し遅れたレプリカとして維持されます。

主データベースで障害が発生しても、スタンバイ・データベースに容易にフェイルオーバーできます。スタンバイにフェイルオーバーすると、今度はそれが新たに主データベースになります。スタンバイ・データベース・サーバーは既にオンラインになっているため、フェイルオーバーは非常に短時間で完了します。それで、データベース・アクティビティーが停止する時間は最小限で済みます。

HADR を使用すれば、特定のハードウェアまたはソフトウェア・リリースのアップグレードにおいてもデータベースを使用可能な状態に維持できます。主データベース側をアプリケーションから利用可能な状態にしたまま、スタンバイ側のハードウェア、オペレーティング・システム、または DB2 UDB フィックスパック・レベルをアップグレードできます。次に、アップグレードしたシステムにアプリケーションを移して、元の主データベースをアップグレードすることができます。

フェイルオーバーの直後の新しい主データベースのパフォーマンスは、障害発生前の主データベースとまったく同じにならない場合があります。新しい主データベースでは、データベース・マネージャーの使用するステートメント・キャッシュ、バッファー・プール、その他のメモリー・ロケーションにデータを入れることがしばしば必要になります。古い主データベースのログ・データを再生すると、バッファー・プールとシステム・カタログ・キャッシュの中のデータの一部が置換されますが、それは書き込みアクティビティーのみに基づくものなので、それだけでは不十分です。頻繁にアクセスされる索引ページ、照会されるが更新はされない表のカタログ情報、ステートメント・キャッシュ、およびアクセス・プランは、いずれもキャッシュに入っていません。しかし、プロセス全体としては、新しい DB2 UDB データベースの起動時よりも高速になります。

障害の発生した主サーバーが修復されたなら、データベースの 2 つのコピーを一貫性のある状態にすることができる場合には、主サーバーをスタンバイ・データベースとして再び統合できます。再統合の後、フェイルバック操作を実行することにより、オリジナルの主データベースを再度主データベースとして使用できるようにすることができます。

HADR フィーチャーを利用できるのは、DB2 UDB Enterprise Server Edition (ESE) においてだけです。 Personal Edition など、その他のエディションでは使用不可です。また、データベース・パーティション・フィーチャー (DPF) 付きの ESE でも使用不可です。

HADR は、インスタンス・レベルではなくデータベース・レベルで実施されます。したがって、単一のインスタンスの中に、主データベース (A)、スタンバイ・データベース (B)、および標準の (非 HADR) データベース (C) が含まれるということがあり得ます。しかし、1 つのインスタンスの中に、単一のデータベースの主とスタンバイの両方を含めることはできません。 HADR では、データベースの各コピーのデータベース名が同じでなければならないからです。

関連概念:

- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『高可用性』

セキュリティー

データベース・サーバーに関連するデータおよびリソースを保護するために、DB2® Universal Database は、外部セキュリティー・サービスと内部アクセス・コントロール情報の組み合わせを使用します。データベース・サーバーにアクセスするには、データベースのデータまたはリソースに対するアクセス権が与えられるまでに、いくつかのセキュリティー検査にパスしなければなりません。データベース・セキュリティーの最初のステップは*認証* と呼ばれ、ユーザーは、自分が言っているとおりの人物であることを証明しなければなりません。 2 番目のステップは*許可* と呼ばれ、そこでは、検査されているユーザーについて、要求したアクションの実行または要求したデータのアクセスが許されるかどうかを、データベース・マネージャーが判断します。

関連概念:

- 28 ページの『認証』
- 29 ページの『許可』

認証

ユーザーの認証は、DB2® Universal Database (DB2 UDB) の外部のセキュリティー機能を使用して完了します。セキュリティー機能は、オペレーティング・システムの一部であるか、別個の製品であるか、または、場合によってはまったく存在しない場合があります。 UNIX® ベースのシステムでは、セキュリティー機能は、オペレーティング・システムそのものの中にあります。

セキュリティーは、ユーザーを認証するのに 2 つの項目を必要とします。それは、ユーザー ID とパスワードです。ユーザー ID は、セキュリティー機能にユーザーを知らせます。正しいパスワード (ユーザーおよびセキュリティー機能にのみ認識されている情報) を提供すれば、ユーザーの身元 (ユーザー ID に対応) が検証されます。

認証された後、

- ユーザーは SQL 許可名または *authid* を使用して、DB2 UDB に識別されなければなりません。この名前は、ユーザー ID と同じものか、またはマップ値にすることができます。たとえば、UNIX ベースのシステムでは、DB2 UDB *authid* は、 DB2 UDB の命名規則に従った UNIX ユーザー ID を大文字に変換することによって得られます。
- そのユーザーが属しているグループのリストが取得されます。グループ・メンバーシップは、ユーザーを許可するときに使用されます。グループは、DB2 UDB 許可名にもマップする必要がある、セキュリティー機能のエンティティーです。このマッピングは、ユーザー ID のために使用される方法と類似した方法で行われます。

DB2 UDB は、セキュリティー機能を使用して、以下の 2 つの方法のうちの 1 つでユーザーを認証します。

- DB2 UDB は成功したセキュリティー・システム・ログインを識別の証拠として使用し、次のことを可能にします。
  - ローカル・データをアクセスするためのローカル・コマンドの使用
  - サーバーがクライアント認証を委託しているリモート接続の使用
- DB2 UDB はユーザー ID とパスワードの組み合わせを受け入れます。この組み合わせの妥当性検査がセキュリティー機能によって成功すると、それをユーザーのアイデンティティーの証拠として使用して、以下のことを許します。
  - サーバーが認証の検査を必要とするリモート接続の使用
  - ログインに使用されたアイデンティティー以外のアイデンティティーの下でユーザーがコマンドを実行したい場合の操作の使用

DB2 UDB for AIX® では、オペレーティング・システムで失敗したパスワード入力をロギングし、 LOGINRETRIES パラメーターで指定されたログイン試行の許可回数をクライアントが超過したときを検出します。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『サーバーでの認証メソッド』
- 「*管理ガイド: インプリメンテーション*」の『特権、権限レベル、およびデータベース権限』
- 28 ページの『セキュリティー』
- 29 ページの『許可』

許可

許可とは、それによって DB2® が、認証された DB2 ユーザーに関する情報 (ユーザーが実行できるデータベース操作、およびアクセスできるオブジェクトを示します) を取得するプロセスのことです。それぞれのユーザー要求について、オブジェクトおよび関係する操作によっては、複数の許可検査が行われる場合があります。

許可は、DB2 の機能を使用して実行されます。それぞれの許可名に関連する許可事項を記録するために、 DB2 表と構成ファイルが使用されます。認証ユーザーの許

可名、およびそのユーザーが属しているグループの許可名が、記録されている許可事項と比較されます。この比較に基づいて、DB2 は、要求されたアクセスを許すかどうかを判断します。

DB2 Universal Database™ (DB2 UDB) によって記録された許可事項のタイプには、「特権」と「権限レベル」の 2 つのタイプがあります。*特権* は、 1 ユーザーがデータベース・リソースを作成またはアクセスできるようにするために、 1 つの許可名に対して単一の許可事項を定義します。特権は、データベース・カタログに保管されます。*権限レベル* は、特権をグループ化する方法を提供し、より高いレベルで、データベース・マネージャーの保守とユーティリティー操作を制御します。データベース固有の権限は、データベース・カタログに保管されます。また、システム権限は、グループ・メンバーシップと関連付けられ、権限レベルに関連するグループ名は、特定のインスタンスについて、データベース・マネージャー構成ファイルの中に保管されます。

グループは、それぞれのユーザーに個別に特権の付与または取り消しを行うことを必要とせずに、ユーザーの集合に対して許可を実行するための便利な手段を提供します。特に異なる指定がなければ、グループ許可名は、許可名が許可の目的で使用されるところであれば、どこでも使用することができます。一般に、グループ・メンバーシップは、動的 SQL およびデータベース以外のオブジェクト (インスタンス・レベルのコマンドおよびユーティリティーなど) の許可のためのものと考えられ、静的 SQL のためのものとは考えられません。この一般的なケースの例外は、特権が PUBLIC に与えられるときに生じ、この場合は静的 SQL が処理されるときに考慮されます。グループ・メンバーシップが適用されない特定のケースについては、DB2 UDB の資料全体を通して、該当する場合にその旨の注が付いています。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『許可と特権』
- 「*管理ガイド: インプリメンテーション*」の『特権、権限レベル、およびデータベース権限』
- 28 ページの『セキュリティー』

作業単位

通常 DB2® Universal Database (DB2 UDB) では、トランザクションのことを*作業単位* といいます。作業単位とは、 1 つのアプリケーション・プロセスの中でリカバリー可能な一連の操作です。データベース・マネージャーは作業単位を使って、データベースを一貫した状態にします。データベースとの間の読み書きは、1 つの作業単位内で行われます。

たとえば、銀行トランザクションでは、資金を普通預金から当座預金に移す場合があります。アプリケーションにより該当金額を普通預金から減算した段階では、 2 つの預金の金額は矛盾した状態になり、この金額が当座預金に加算される時点まで、一貫していない状態が続きます。*両方の* ステップが完了した時点で、整合点に達します。そして変更がコミットされ、他のアプリケーションから使用できるようになります。

作業単位は、最初の SQL ステートメントがデータベースに対して発行される時点で開始します。作業単位は、 COMMIT または ROLLBACK ステートメントを発行することによってアプリケーションから終了させる必要があります。 COMMIT ステートメントを使うと、作業単位内でなされたすべての変更内容が固定されます。ROLLBACK ステートメントを使うと、これらの変更内容がデータベースから除去されます。アプリケーションがどちらのステートメントも明示的に発行しないまま正常終了すると、作業単位は自動的にコミットされます。アプリケーションが作業単位の途中で異常終了すると、作業単位は自動的にロールバックされます。いったん COMMIT または ROLLBACK を発行すると、それを停止することはできません。一部のマルチスレッド・アプリケーションやオペレーティング・システム (たとえば Windows®) では、アプリケーションがこのどちらのステートメントも明示的に発行せずに正常終了すると、作業単位は自動的にロールバックされます。アプリケーションでは、完了した作業単位を常に明示的にコミットまたはロールバックすることをお勧めします。作業単位の一部が正常に完了しない場合、更新内容はロールバックされ、処理されていた表はトランザクション開始前の状態に戻ります。このようにして、要求が失われたり、重複したりしないようになっています。

関連資料:

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

第 2 章 並列データベース・システム

データのパーティション化

DB2® Universal Database (DB2 UDB) は、データベース・マネージャーの機能を、並列、マルチノードの環境に拡張します。データベース・パーティション は、データベースの一部であり、それ自体のデータ、索引、構成ファイル、およびトランザクション・ログからなります。データベース・パーティションは、ノードまたはデータベース・ノードと呼ばれる場合があります。

単一パーティション・データベース は、 1 つだけのデータベース・パーティションを持つデータベースです。データベース内のすべてのデータが、そのパーティションに保管されます。この場合、データベース・パーティション・グループがあっても、追加の機能は提供されません。

パーティション・データベース は、 2 つ以上のデータベース・パーティションを持つデータベースです。表は、1 つ以上のデータベース・パーティションに配置することができます。表が複数のパーティションからなるデータベース・パーティション・グループ内にある場合、その行の一部が 1 つのパーティションに保管され、その他の行は他のパーティションに保管されます。

通常、単一のデータベース・パーティションが各物理ノードに存在し、データベース全体のデータのうちの一部を管理するために、各システムのプロセッサーが各データベース・パーティションのデータベース・マネージャーによって使用されます。

データは複数のデータベース・パーティションに分割されているので、複数の物理ノード上にある複数のプロセッサーの力を使用して、情報に対する要求を処理することができます。データ検索と更新の要求は自動的にサブの要求に分解され、適用可能なデータベース・パーティション内で並列に実行されます。データベースが複数のデータベース・パーティションに分割されているという事実を、 SQL ステートメントを発行しているユーザーが認識する必要はありません。

ユーザーとの接続は、そのユーザーに対する *コーディネーター・ノード* として知られているデータベース・パーティションを介して生じます。コーディネーター・ノードは、アプリケーションと同じデータベース・パーティションで実行されるか、またはリモート・アプリケーションの場合、そのアプリケーションが接続されるデータベース・パーティションで実行されます。任意のデータベース・パーティションをコーディネーター・ノードとして使用することができます。

DB2 UDB は、データベース内の複数のデータベース・パーティションにわたってデータを保管できるようにする、パーティション化ストレージ・モデルをサポートしています。つまり、データが物理的には複数のデータベース・パーティションにわたって保管されていても、 1 つの同じ場所に置かれているかのようにアクセスできます。パーティション・データベースのデータにアクセスするアプリケーションやユーザーは、データが物理的にどこにあるかを認識する必要がありません。

データは、物理的には分割されていますが、1 つの論理的な統一体として使用および管理されます。ユーザーは、パーティション・キーを定義することによって、自分のデータをパーティション化する方法を選択することができます。ユーザーはまた、データを保管する表スペースと関連するデータベース・パーティション・グループを選択することによって、自分のデータを展開できるデータベース・パーティションを指定したりデータベース・パーティションの数を決定することもできます。「設計アドバイザー」を使用することにより、パーティション化とレプリケーションに関する提案をすることができます。さらに、更新可能なパーティション・マップをハッシュ・アルゴリズムとともに使用して、データベース・パーティションへのパーティション・キー値のマッピングを指定します (これによってデータの各行の配置と検索が決まります)。その結果、大きな表のためのワークロードをパーティション・データベース全体に分散させると同時に、小さい表を 1 つまたは複数のデータベース・パーティションに保管することができます。それぞれのデータベース・パーティションは保管するデータのローカル索引を持っており、その結果、ローカルのデータ・アクセスのパフォーマンスが向上します。

すべての表を、データベース内のすべてのデータベース・パーティションに分割しなければならないという設計上の制限はありません。 DB2 UDB は*部分デクラスタリング* をサポートします。これによって、表および表スペースをシステム内のデータベース・パーティションのサブセットに分割できます。

それぞれのデータベース・パーティションに表を置きたいときに考慮できる別の方法は、マテリアライズ照会表を使用してからそれらの表を複製するというものです。まず必要な情報を含むマテリアライズ照会表を作成して、それを各ノードに複製します。

並列処理

データベース照会などの作業のコンポーネントは、並列に実行することにより、パフォーマンスを大幅に強化できます。作業の性質、データベース構成、およびハードウェア環境すべてによって、 DB2® Universal Database (DB2 UDB) が作業を並列に実行する方法は異なります。これらの考慮事項は相互に関連しているため、データベースの物理的および論理的な設計の作業をする際に、これらを一緒に検討する必要があります。 DB2 UDB では、以下のタイプの並列処理がサポートされています。

- 入出力
- 照会
- ユーティリティー

入出力の並列処理

表スペースに複数のコンテナーがある場合、データベース・マネージャーは*並列入出力* を利用できます。並列入出力とは、 2 つまたはそれ以上の入出力装置への書き込み処理またはそこからの読み取り処理を同時に行うことです。この結果、スループットが大幅に向上します。

照会並列処理

照会並列処理のタイプには、照会間並列処理と照会内並列処理の 2 つがあります。

照会間並列処理 とは、同時に複数のアプリケーションからの照会を受け付けるという、データベースの機能です。各照会はそれぞれ他の照会と独立して実行されますが、 DB2 UDB はそれらのすべてを同時に実行します。 DB2 UDB は、このタイプの並列処理を常にサポートします。

照会内並列処理 とは、*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* (あるいはその両方) を使用して、単一の照会の一部分を同時に処理することです。

パーティション内並列処理

パーティション内並列処理 とは、 1 つの照会を複数の部分に分割する機能のことです。一部の DB2 UDB ユーティリティーも、このタイプの並列処理を実行します。

パーティション内並列処理では、索引の作成、データベースのロード、または SQL 照会など、通常は単一データベース操作と考えられている操作を複数の部分に分割して、それらの部分の多くまたはすべてが単一のデータベース・パーティション内で 並列して実行できるようにします。

36 ページの図 13 は、3 つのピース (部分) に分割して並列に実行できるようにする照会を示したものであり、照会が順次に実行された場合に比べてより速く結果が戻されます。これらのピースは、お互いのコピーです。パーティション内並列処理を利用するためには、データベースを適切に構成する必要があります。並列処理の度合いは自分で選択するか、またはシステムに選択させるようにすることができます。並列処理の度合いは、並列に実行する照会のピースの数を表しています。

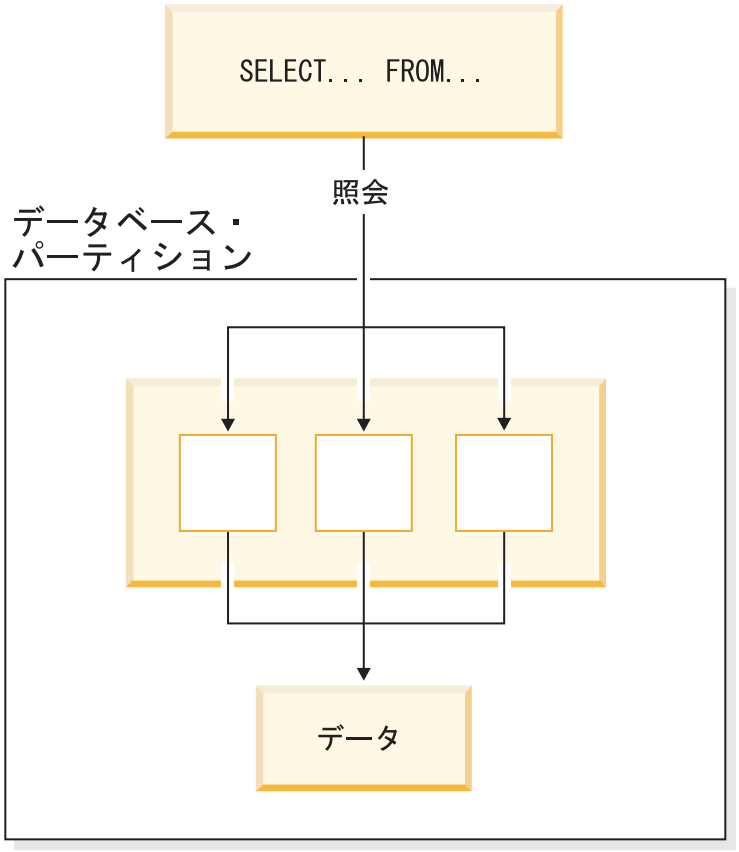

図 13. パーティション内並列処理

パーティション間並列処理

パーティション間並列処理 とは、1 つのマシンまたは複数のマシン上で、パーティション・データベースの複数のパーティションに渡って 1 つの照会を複数の部分に分割する機能のことです。照会は並列に実行されます。一部の DB2 UDB ユーティリティーも、このタイプの並列処理を実行します。

パーティション間並列処理では、索引の作成、データベースのロード、または SQL 照会など、通常は単一データベース操作と考えられている操作を複数の部分に分割して、それらの部分の多くまたはすべてが、 *1* つのマシンまたは複数のマシンで、パーティション・データベースの複数のパーティションに渡って 並列して実行できるようにします。

37 ページの図 14 は、3 つのピースに分割して並列に実行できるようにする照会を示したものであり、照会が単一のパーティション内で順次に実行された場合に比べてより速く結果が戻されます。

並列処理の度合いは、作成したパーティションの数とデータベース・パーティション・グループを定義した方法によって大部分が決まります。

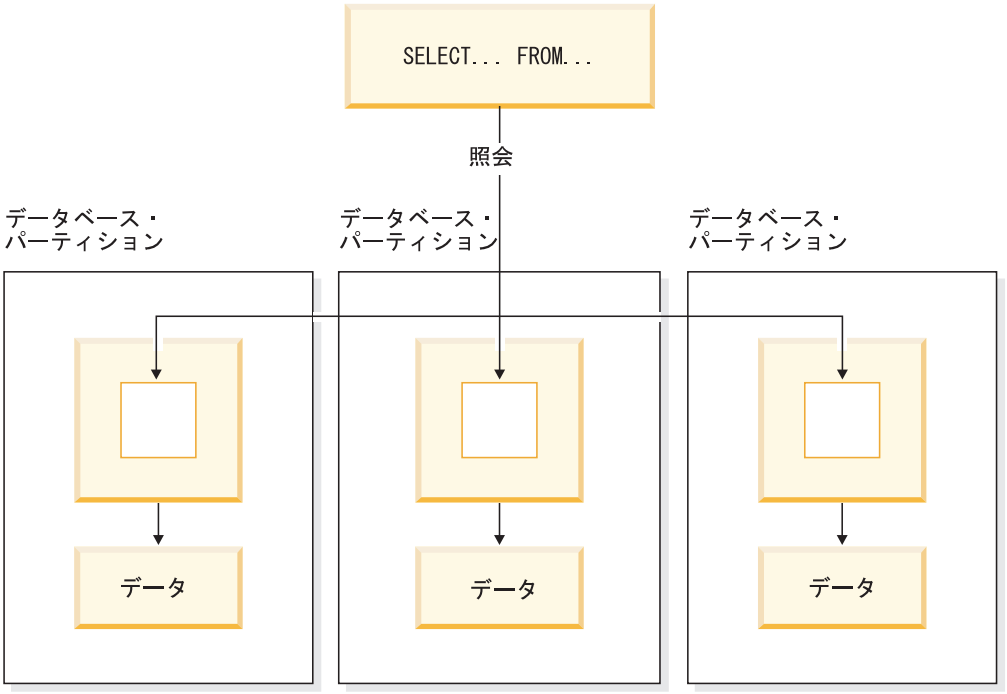

図 14. パーティション間並列処理

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同時使用

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を同時に使用することができます。この組み合わせにより 2 段階で並列処理が行われるため、この結果、照会の処理スピードが劇的に速くなります。

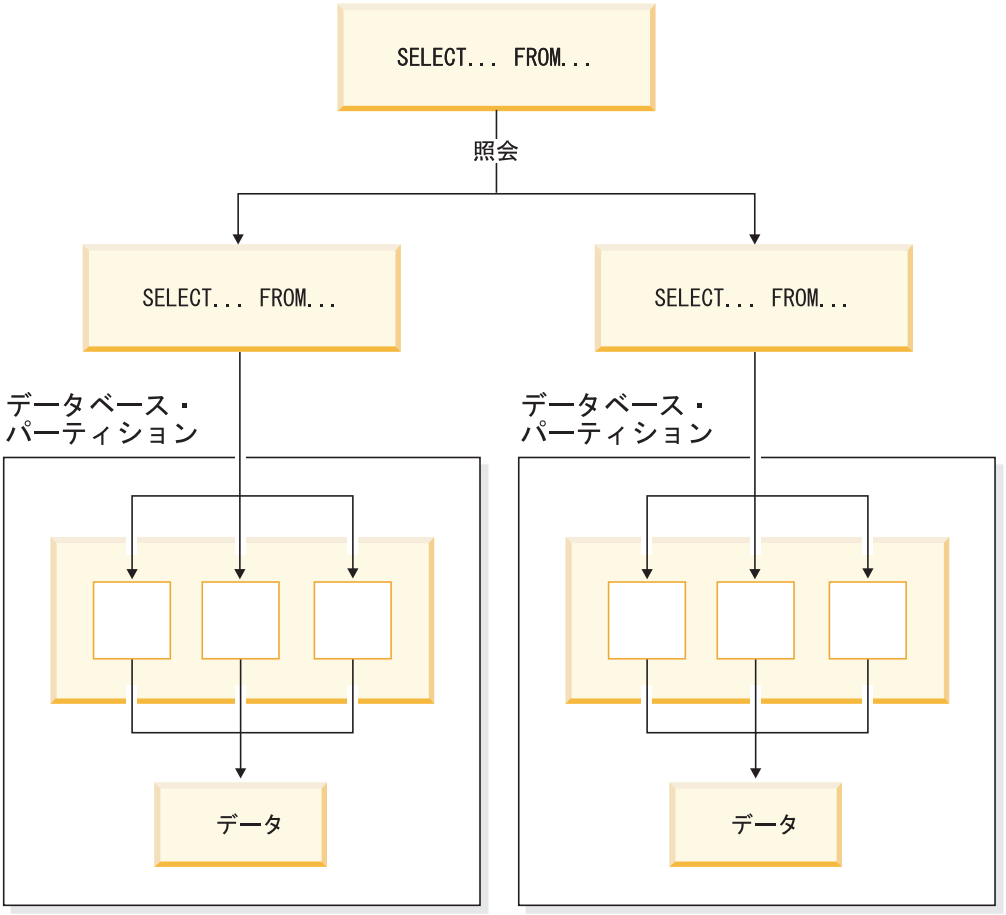

図 15. パーティション間並列処理とパーティション内並列処理の同時使用

ユーティリティー並列処理

DB2 UDB のユーティリティーは、パーティション内並列処理を利用できます。これらのユーティリティーはパーティション間並列処理も利用しており、複数のデータベース・パーティションが存在していると、ユーティリティーはそれぞれのパーティションで並列に実行されます。

ロード・ユーティリティーは、パーティション内並列処理と入出力並列処理を利用することができます。データのロードは、CPU 集中型のタスクです。ロード・ユーティリティーは、データの解析および形式設定などのタスクに、複数のプロセッサーを利用します。また、ロード・ユーティリティーは、並列入出力サーバーを使用して、コンテナーにデータを並列に書き込むことができます。

パーティション・データベース環境において、LOAD コマンドは、表が存在する各データベース・パーティションで並列に呼び出され、パーティション内、パーティション間、および入出力並列処理を実行します。

索引の作成時には、データのスキャンとその後のソートが並列して実行されます。DB2 UDB では、索引を作成するときに、入出力並列処理とパーティション内並列処理の両方を利用しています。これは、再始動時 (索引が無効としてマーク付けさ

れている場合) およびデータの再編成時に、 CREATE INDEX ステートメントが出されたときの索引作成のスピードアップに役立ちます。

データのバックアップとリストアは、入出力制約の大きいタスクです。 DB2 UDB では、バックアップ操作とリストア操作を実行するときに、入出力並列処理とパーティション内並列処理の両方を利用しています。バックアップでは、複数の表スペース・コンテナーから並列に読み取り、複数のバックアップ・メディアに非同期的に並列に書き込みを行うことによって、入出力並列処理を利用しています。

関連概念:

- 39 ページの『パーティションおよびプロセッサー環境』

パーティションおよびプロセッサー環境

このセクションでは、以下のハードウェア環境についての概要を説明します。

- 単一プロセッサー (ユニプロセッサー) 上での単一パーティション
- 複数プロセッサーを備えた単一パーティション (SMP)
- 複数パーティション構成
  - 1 つのプロセッサーを備えたパーティション (MPP)
  - 複数のプロセッサーを備えたパーティション (SMP のクラスター)
  - 論理データベース・パーティション (AIX® 用 DB2® パラレル・エディションのバージョン 1 では、複数論理ノードまたは MLN とも呼ばれる)

容量および拡張容易性は、それぞれの環境ごとに説明します。*容量* とは、データベースをアクセスできるユーザーおよびアプリケーションの数のことです。その大部分は、メモリー、エージェント、ロック、入出力、およびストレージ管理によって決まります。*拡張容易性* とは、データベースが拡張されても、同じ操作特性と応答時間を示し続ける能力のことです。

単一プロセッサー上での単一パーティション

この環境は、メモリーとディスクからなりますが、単一の CPU しか含まれていません (40 ページの図 16 を参照)。これはスタンドアロン・データベース、クライアント/サーバー・データベース、シリアル・データベース、単一プロセッサー・システム、および単一ノードまたは非並列環境など、多くの名前で呼ばれています。

この環境におけるデータベースは、部門または小さなオフィスのニーズを満たすもので、そこでは、データおよびシステム・リソース (単一プロセッサーまたは CPU を含む) が単一のデータベース・マネージャーによって管理されます。

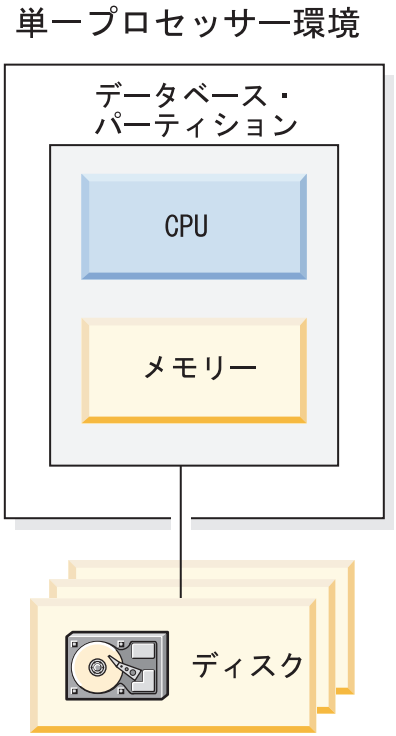

図 16. 単一プロセッサー上での単一パーティション

容量および拡張容易性

この環境では、さらにディスクを追加することができます。各ディスクごとに 1 つ以上の入出力サーバーを持つことによって、同時に複数の入出力操作を行うことができます。

単一プロセッサー・システムは、プロセッサーが処理できるディスク・スペースの量によって制限されます。ワークロードが増加すると、メモリーやディスクなどのコンポーネントにかかわりなく、単一 CPU ではユーザー要求をそれ以上速く処理することはできなくなる場合があります。容量または拡張容易性の最大に到達してしまった場合は、複数プロセッサーを備えた単一パーティション・システムに移行することを検討します。

複数プロセッサーを備えた単一パーティション

この環境は通常、同じマシン内の複数の等価の処理能力を持つプロセッサーからなり (41 ページの図 17 を参照)、*対称型マルチプロセッサー (SMP)* システムと呼ばれます。ディスク・スペースおよびメモリーなどのリソースは、*共有* されます。

複数のプロセッサーが使用可能なので、異なるデータベースの操作をより速く完了させることができます。また DB2 Universal Database™ (DB2 UDB) では、処理スピードを向上させるために、単一の照会の作業を、使用可能な複数のプロセッサーに分割することもできます。他のデータベース操作、たとえばデータのロード、表スペースのバックアップおよびリストア、および既存のデータの索引の作成などでも、複数のプロセッサーを利用できます。

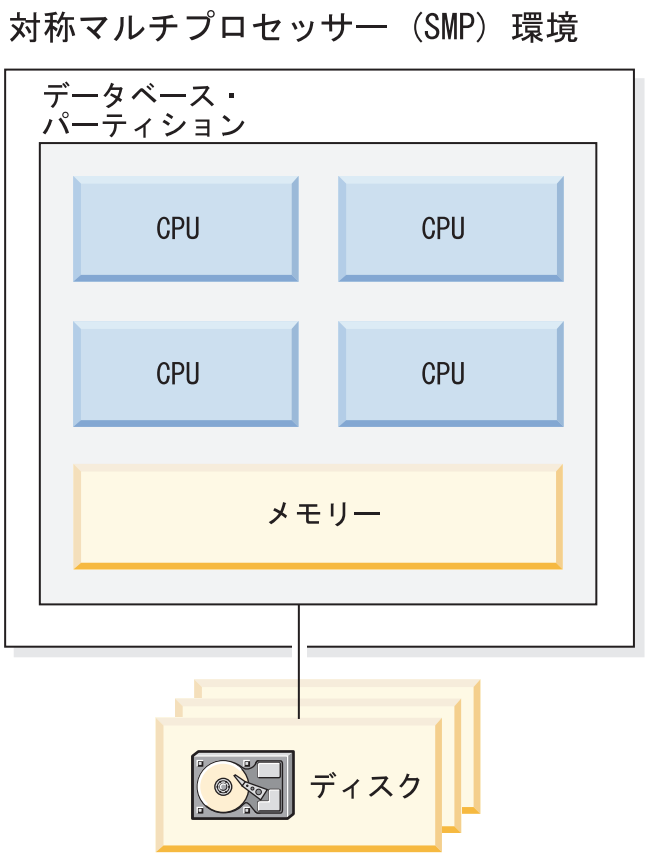

図 17. 単一パーティション・データベース対称型マルチプロセッサー環境

容量および拡張容易性

この環境では、さらにプロセッサーを追加することができます。ただし、異なるプロセッサーが同じデータのアクセスを試みる可能性があるため、業務の操作が拡大するにつれて、この環境の制約が現れてくる可能性があります。共有メモリーと共有ディスクを使用すれば、すべてのデータベース・データを効率的に共有することができます。

ディスクの数を増やすことにより、プロセッサーに関連するデータベース・パーティションの入出力容量を増やすことができます。特に入出力要求を処理するために、入出力サーバーを設定することができます。各ディスクごとに 1 つ以上の入出力サーバーを持つことによって、同時に複数の入出力操作を行うことができます。

容量または拡張容易性の最大に到達してしまった場合は、複数パーティションを備えたシステムに移行することを検討します。

複数パーティション構成

1 つのデータベースを複数のパーティションに分割して、それぞれのパーティションが独自のマシン上にあるようにすることができます。複数データベース・パーティションを備えた複数マシンを一緒にグループ化することができます。このセクションでは、以下のパーティション構成について説明します。

- 1 つのプロセッサーを備えたシステムのパーティション

- 複数のプロセッサーを備えたシステムのパーティション
- 論理データベース・パーティション

1 つのプロセッサーを備えたパーティション

この環境では、多くのデータベース・パーティションがあります。それぞれのパーティションは独自のマシン上に常駐しており、独自のプロセッサー、メモリー、およびディスクを持っています (図 18)。すべてのマシンは通信機能によって接続されています。この環境は、クラスター、単一プロセッサー・クラスター、超並列処理 (MPP) 環境、およびシェアード・ナッシング (shared-nothing) などの多くの名前で呼ばれています。後の方の名前は、この環境におけるリソースの配置を反映したものです。 SMP 環境と異なり、MPP 環境には共有のメモリーまたはディスクはありません。 MPP 環境は、メモリーおよびディスクの共有によって発生する制約を除去します。

パーティション・データベース環境によって、データベースは、物理的には複数のパーティション化されていても、 1 つの完全な論理的なものとして扱えます。データがパーティション化されているという事実をほとんどのユーザーは認識する必要がありません。作業は、データベース・マネージャー間で分割できます。各パーティションのそれぞれのデータベース・マネージャーは、自分の分担する部分のデータベースに対して作業を行います。

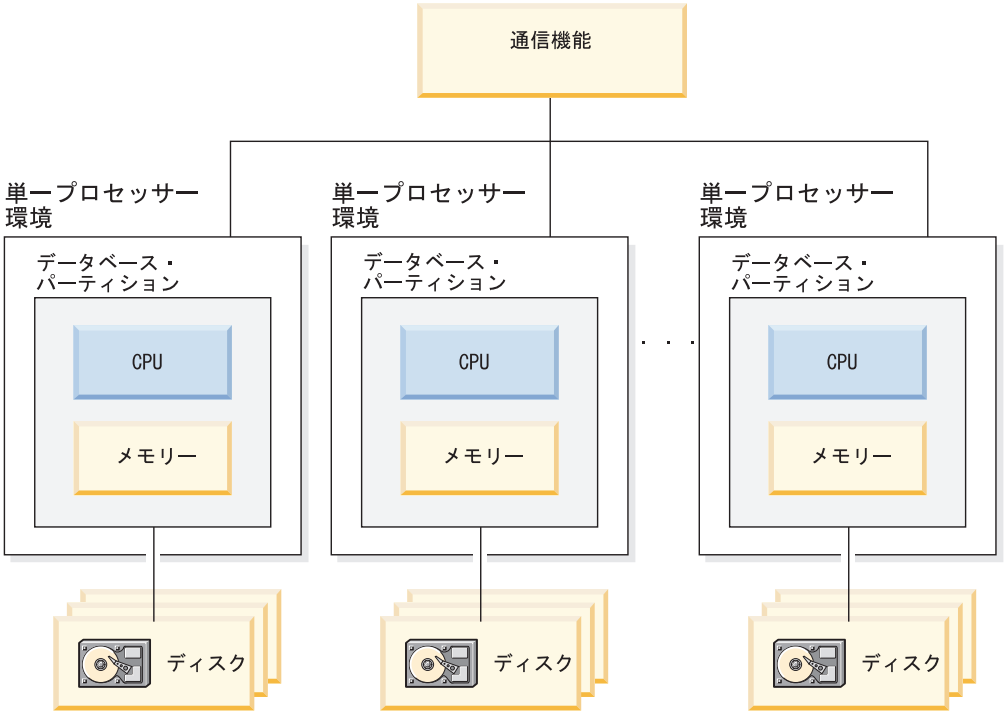

図 18. 超並列処理 (MPP) 環境

容量および拡張容易性: この環境では、さらにデータベース・パーティション (ノード) を構成に追加することができます。一部のプラットフォーム (たとえば RS/6000® SP™) では、最大は 512 ノードです。ただし、多数のマシンとインスタンスを管理することに関して実行上の制限がある場合があります。

容量または拡張容易性の最大に到達してしまった場合は、各パーティションが複数のプロセッサーを備えたシステムに移行することを検討します。

複数プロセッサーを備えたパーティション

各パーティションが単一プロセッサーを持つ構成に対する代替の構成は、 1 つのパーティションが複数のプロセッサーを持つ構成です。これは、*SMP* クラスター と呼ばれます (図 19)。

この構成は、SMP 並列処理と MPP 並列処理の利点を組み合わせたものになります。これは、1 つの照会を複数プロセッサーにわたって単一パーティションで実行できることを意味します。また、1 つの照会を複数パーティションにわたって並列に実行できることも意味します。

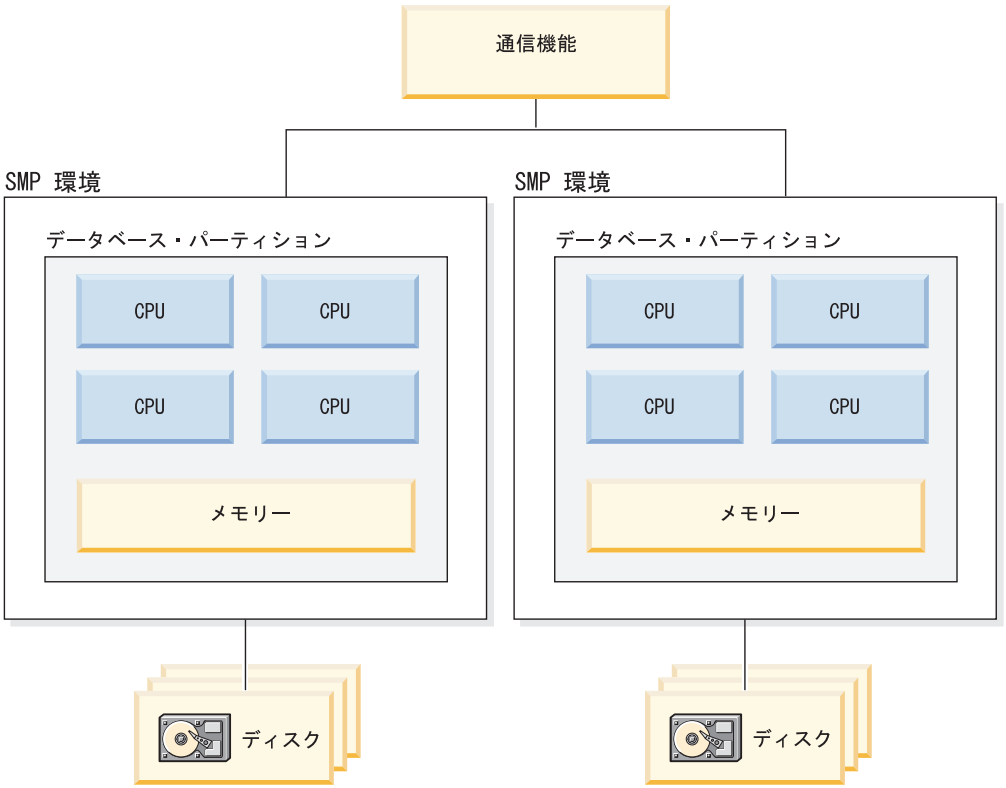

図 19. 複数の対称型マルチプロセッサー (SMP) 環境からなるクラスター

容量および拡張容易性: この環境では、さらにデータベース・パーティションを追加したり、既存のデータベース・パーティションにさらにプロセッサーを追加したりすることができます。

論理データベース・パーティション

論理データベース・パーティションは、マシン全体の制御権が与えられていないところが物理パーティションと異なります。マシンは共有リソースを持っていますが、データベース・パーティションはリソースを共有しません。プロセッサーは共有されますが、ディスクとメモリーは共有されません。

論理データベース・パーティションは拡張容易性を提供します。複数の論理パーティションで実行している複数のデータベース・マネージャーは、単一のデータベー

ス・マネージャーが利用可能なリソースよりも、使用可能なリソースの全てを利用することができる場合があります。 図 20 は、特に多くのプロセッサーを備えたマシンについて、さらにパーティションを追加することによって、 SMP マシン上でさらに拡張容易性を獲得できることを示したものです。データベースをパーティション化することによって、それぞれのパーティションを個別に管理およびリカバリーすることができます。

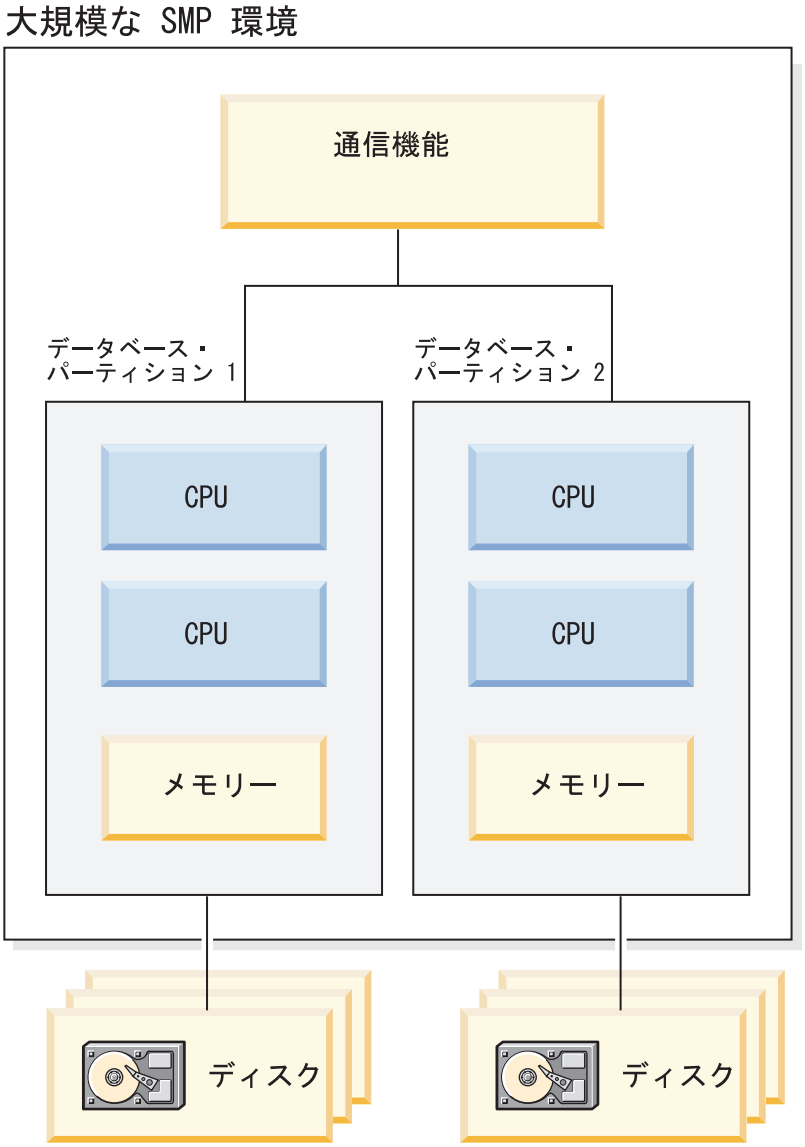

図 20. 対称型マルチプロセッサー環境でのパーティション・データベース

45 ページの図 21 は、処理能力を高めるために、 図 20 に示された構成を増やすことができることを示したものです。

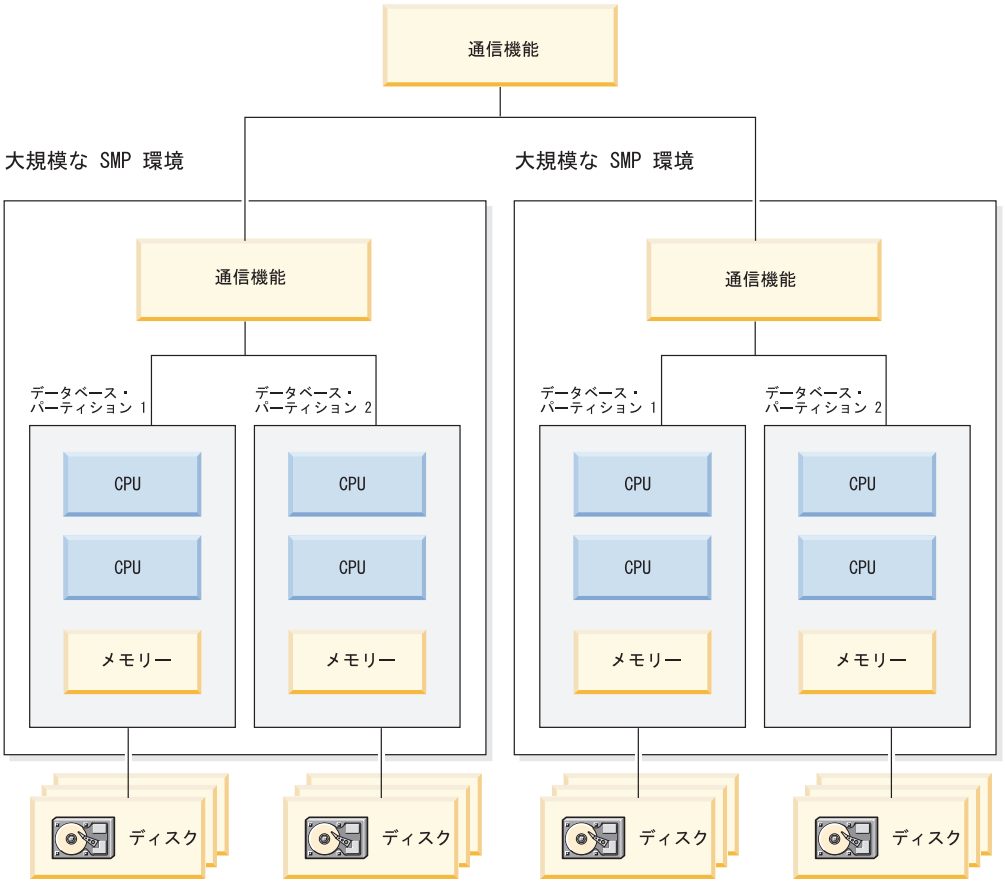

図 21. クラスターを構成する複数の対称型マルチプロセッサー環境を含むパーティション・データベース

注: また、2 つ以上のパーティションを、(プロセッサーの数には無関係に) 同じマシン上に共存させる機能によって、高可用性構成とフェイルオーバー対策を設計する上でより大きい柔軟性が得られます。マシン故障の際に、同じデータベースの別のパーティションがすでに含まれている 2 番目のマシンに自動的にデータベース・パーティションを移動して再始動できます。

各ハードウェア環境に最適な並列処理のサマリー

以下の表は、さまざまなハードウェア環境を活用するのに最適な並列処理のタイプをまとめたものです。

表 3. それぞれのハードウェア環境ごとの可能な並列処理のタイプ

ハードウェア環境	入出力並列処理	照会内並列処理	
		パーティション内並列処理	パーティション間並列処理
単一パーティション、シングル・プロセッサー	可	不可 注(1)	不可
単一パーティション、複数プロセッサー (SMP)	可	可	不可
複数パーティション、1 プロセッサー (MPP)	可	不可 注(1)	可

表 3. それぞれのハードウェア環境ごとの可能な並列処理のタイプ (続き)

ハードウェア環境	入出力並列処理	照会内並列処理	
		パーティション内並列処理	パーティション間並列処理
複数パーティション、複数プロセッサー (SMP のクラスター)	可	可	可
論理データベース・パーティション	可	可	可
注: (1) 特に実行する照会が完全に CPU を利用していない場合 (たとえば入出力制約型である場合) では、シングル・プロセッサー・システムの場合でも、並列処理の度合いを 1 よりも大きな値に (構成パラメーターの 1 つを使用して) 設定すると好結果が得られる場合があります。			

関連概念:

- 34 ページの『並列処理』

第 3 章 データウェアハウジングについて

データウェアハウジングが提供するソリューション

運用データ (業務の日常トランザクションを実行するデータ) を含むシステムには、業務分析に便利な情報が含まれています。たとえば、分析者はどの製品がどの地域で、年のどの時期に売れたかに関する情報を使用して、例外を探したり、将来の売上を予測したりできます。

しかし、分析者が運用データに直接アクセスすると、いくつかの問題が起こる可能性があります。

- 分析者が運用データベースを照会するための専門技術を持っていない可能性がある。たとえば、IMS™ データベースの照会には、特殊なタイプのデータ操作言語を使用するアプリケーション・プログラムが必要です。一般に、運用データベースを照会するための専門技術を持つプログラマーは、データベースおよびそのアプリケーションを保守するのに業務の全時間を費やしています。
- 銀行のデータベースなどの多くの運用データベースでは、パフォーマンスが重要です。このようなシステムでは、随時照会を作成するユーザーを処理できません。
- 一般に運用データは、業務分析者が使用するための最善の形式になっていません。たとえば、製品、地域、および季節ごとに要約されている販売データは、そのままのデータよりも分析者にとって役立ちます。

データウェアハウジングは、これらの問題を解決します。データウェアハウジングでは、*情報データ* のストアを作成します。情報データとは、意思決定作業のために運用データから抽出されてトランスフォームされたデータのことです。たとえば、データウェアハウジング・ツールでは、すべての販売データを運用データベースからコピーして、データを整理し、データの要約を計算し、運用データとは別のデータベースにあるターゲットに要約データを書き込むことができます。ユーザーは、運用データベースに影響を与えることなく、この個別のデータベース (ウェアハウス) を照会できます。

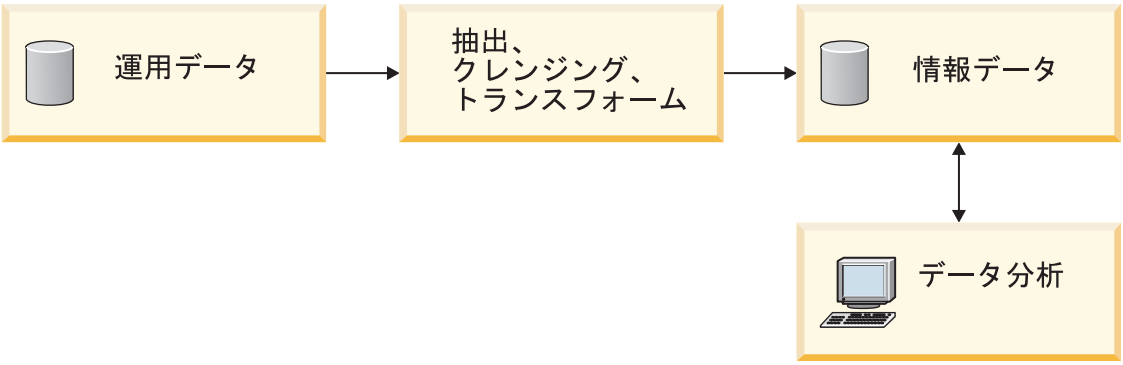

図 22. 運用データからデータ分析へのパス

関連概念:

- 48 ページの『データウェアハウス・オブジェクト』

データウェアハウス・オブジェクト

以降のセクションでは、データウェアハウスの作成と保守に使用するオブジェクトについて説明します。

サブジェクト・エリア

サブジェクト・エリアは、業務の論理領域と関連するプロセスを識別し、グループ化します。たとえば、マーケティングおよび売上データのウェアハウスを作成している場合は、「売上 (Sales)」サブジェクト・エリアと「マーケティング (Marketing)」サブジェクト・エリアを定義します。次に、「売上 (Sales)」サブジェクト・エリアの下に、販売に関連するプロセスを追加できます。同様に、「マーケティング (Marketing)」サブジェクト・エリアの下に、マーケティング・データと関連する定義を追加できます。

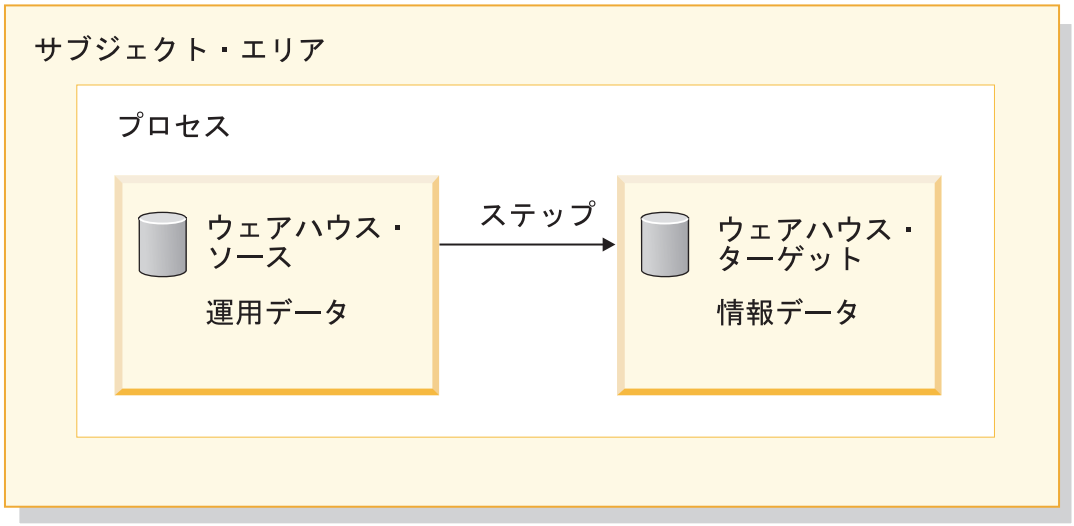

図 23. サブジェクト・エリアの階層

ウェアハウス・ソース

ウェアハウス・ソースは、ウェアハウスにデータを提供する表とファイルです。データウェアハウス・センターはウェアハウス・ソースの指定を使用して、データにアクセスします。ソースには、ネットワークに接続可能なほとんどすべてのリレーショナル・ソースまたは非リレーショナル・ソース (表、ビュー、またはファイル)、または WebSphere® Site Analyzer ソースを使用できます。

ウェアハウス・ターゲット

ウェアハウス・ターゲットは、トランスフォームされたデータを含むデータベース表またはファイルです。ユーザーはウェアハウス・ターゲットを使用して、他のウェアハウス・ターゲットにデータを提供することができます。たとえば、中央のウ

ェアハウスから部門サーバーにデータを提供したり、ウェアハウス内のメイン・ファクト表からサマリー表にデータを提供したりすることができます。

ウェアハウス・コントロール・データベース

ウェアハウス・コントロール・データベースには、データウェアハウス・センター・メタデータの保管に必要なコントロール表が入っています。データウェアハウス・センターのバージョン 8.2 から、ウェアハウス・コントロール・データベースは UTF-8 (Unicode Transformation Format または Unicode) データベースである必要があります。これにより、データウェアハウス・センターの言語サポートが拡張されます。 Unicode 形式でないデータベースを使用してデータウェアハウス・センターへのログオンを試行すると、ログオンできないことを示すエラー・メッセージが表示されます。ウェアハウス・コントロール・データベース管理ツールを使用すると、以前のデータベースから新しい Unicode データベースへメタデータを移行できます。

ウェアハウス・エージェントおよびエージェント・サイト

ウェアハウス・エージェントは、データ・ソースとターゲット・ウェアハウス間の、データの流れを管理します。ウェアハウス・エージェントは、AIX®、Linux、iSeries™、 z/OS™、Windows® NT、Windows 2000、および Windows XP オペレーティング・システムで、また Solaris™ オペレーティング環境で使用できます。エージェントは Open Database Connectivity (ODBC) ドライバーまたは DB2® CLI を使用して、さまざまなデータベースと通信します。

複数のエージェントが、ソース・ウェアハウスとターゲット・ウェアハウス間のデータの転送を処理できます。使用するエージェントの数は、既存の接続構成と、ウェアハウスに移動することを計画しているデータのボリュームによって異なります。同じエージェントを必要とする複数のプロセスを同時に実行する場合は、そのエージェントの追加インスタンスを生成できます。

エージェントには、ローカル・エージェントとリモート・エージェントがあります。ローカル・ウェアハウス・エージェントは、ウェアハウス・サーバーと同じワークステーションにインストールされているエージェントです。リモート・ウェアハウス・エージェントは、ウェアハウス・サーバーに接続できる別のワークステーションにインストールされているエージェントです。

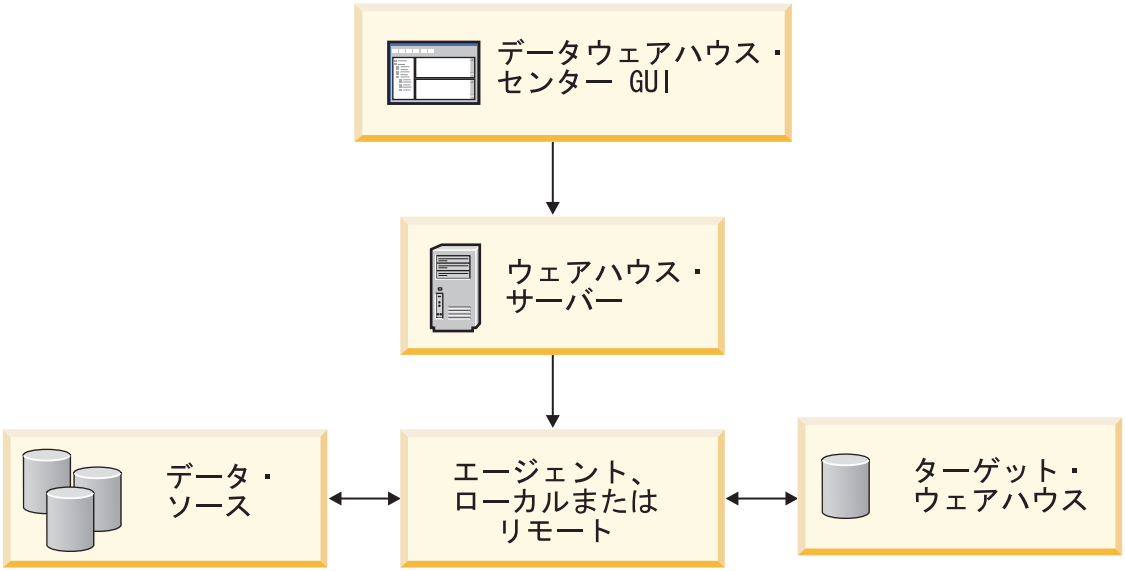

図 24. エージェント、データ・ソース、ターゲット・ウェアハウス、およびウェアハウス・サーバーの間の関係

エージェント・サイトとは、エージェント・ソフトウェアがインストールされているワークステーションの論理名です。エージェント・サイトの名前は、TCP/IP ホスト名と同じではありません。単一のワークステーションは、1 つの TCP/IP ホスト名しか持つことができませんが、単一のワークステーションに対して複数のエージェント・サイトを定義することが可能です。それぞれのエージェント・サイトは論理名で識別されます。

デフォルトのエージェント・サイトの名前は Default DWC Agent Site で、ウェアハウス・コントロール・データベースの初期化時にデータウェアハウス・センターが定義するローカル・エージェントです。

プロセスとステップ

プロセスには、特定のウェアハウス用にデータのトランスフォーメーションと移動を実行する一連のステップが含まれています。一般に、プロセスはソース・データをウェアハウスに移動します。その後、データはウェアハウス用に集約および要約されます。プロセスでは、単一のフラット表または一連のサマリー表を生成できます。また、特定タイプのデータ・トランスフォーメーションを実行することもできます。

ステップは、ウェアハウス内の単一操作の定義です。 SQL ステートメントを使用したりプログラムを呼び出したりすることにより、ステップはデータの移動およびトランスフォームの方法を定義します。ステップを実行すると、ウェアハウス・ソースとウェアハウス・ターゲット間のデータ転送、またはそのデータのトランスフォーメーションが発生します。

ステップは、データウェアハウス・センターにある論理エンティティーであり、次のものを定義します。

- ソース・データへのリンク
- 出力表またはファイルの定義、またはそれらへのリンク

- 出力表またはファイルにデータを入れるためのメカニズム (SQL ステートメントまたはプログラム) および定義
- 出力表またはファイルにデータを入れるための処理オプションおよびスケジュール

データウェアハウス・センターに次のタスクを実行させるとします。

1. 各種のデータベースからデータを抽出する。
2. データを単一の形式に変換する。
3. データをデータウェアハウスの表に書き込む。

いくつかのステップを含むプロセスを作成します。データベースからのデータの抽出、正しい形式への変換などのように、ステップごとに別々のタスクが実行されます。データを完全にトランスフォームおよび形式設定し、それを最後の表に入れるための、いくつかのステップを作成する必要があるかもしれません。

ステップまたはプロセスが実行されると、次のようにターゲットに影響します。

- ウェアハウス・ターゲット内のすべてのデータを新しいデータと置換する。
- 既存のデータに新しいデータを追加する。
- データの別個のエディションを追加する。
- 既存のデータを更新する。

ステップは、必要に応じて実行したり、以下のスケジュールで実行したりすることができます。

- 設定時刻
- 一度だけ
- 繰り返し (毎週の金曜日など)
- 順番に (1 つのステップの実行が終了したら、次のステップの実行が開始される)
- 完了時 (他のステップの成功時または失敗時)

プロセスをスケジュールする場合、スケジュールした時間にプロセスの最初のステップが実行されます。

以降のセクションでは、データウェアハウス・センターで使用できるさまざまなタイプのステップについて説明します。

SQL ステップ

データウェアハウス・センターには、2 つのタイプの SQL ステップがあります。「SQL 選択および挿入 (Select and Insert)」ステップでは、SQL SELECT ステートメントを使用して、ウェアハウス・ソースからデータを抽出し、そのデータをウェアハウス・ターゲット表に挿入するための INSERT ステートメントを生成します。「SQL 選択および更新 (Select and Update)」ステップでは、SQL SELECT ステートメントを使用して、ウェアハウス・ソースからデータを抽出し、ウェアハウス・ターゲット表内の既存のデータを更新します。

プログラム・ステップ

データウェアハウス・センターには、いくつかのプログラム・ステップがあります。 DB2 for iSeries プログラム、DB2 for z/OS プログラム、DB2 Universal

Database™ プログラム、 Visual Warehouse™ 5.2 DB2 プログラム、OLAP Server プログラム、ファイル・プログラム、およびレプリケーション・プログラムです。これらのステップは、事前定義されたプログラムとユーティリティーを実行します。特定のオペレーティング・システムのウェアハウス・プログラムは、オペレーティング・システムのエージェントと共にパッケージされています。ウェアハウス・プログラムは、エージェント・コードのインストール時にインストールしてください。

トランスフォーマー・ステップ

トランスフォーマー・ステップは、ストアード・プロシージャーとユーザー定義関数であり、データをトランスフォームするために使用できる統計またはウェアハウス・トランスフォーマーを指定します。トランスフォーマーを使用して、データのクリーニング、逆転、およびピボット、主キーおよび期間表の生成、および各種統計の計算を行うことができます。

トランスフォーマー・ステップでは、統計またはウェアハウス・トランスフォーマーの 1 つを指定します。このプロセスを実行すると、トランスフォーマー・ステップは 1 つまたは複数のウェアハウス・ターゲットにデータを書き込みます。

ユーザー定義プログラム・ステップ

ユーザー定義プログラム・ステップは、データウェアハウス・センターにある論理エンティティーであり、データウェアハウス・センターが開始するビジネス固有のトランスフォーメーションを表します。いかなるビジネスにもユニークなデータ・トランスフォーメーション要件があるので、ビジネスでは、独自のプログラム・ステップを作成するか、または ETI や Vality などの他社が提供するツールを使用するかを選択できます。

たとえば、以下の関数を実行するユーザー定義プログラムを作成できます。

1. 表からデータをエクスポートする。
2. そのデータを操作する。
3. データを一時的な出力リソースまたはウェアハウス・ターゲットに書き込む。

関連概念:

- 47 ページの『データウェアハウジングが提供するソリューション』
- 52 ページの『ウェアハウス・タスク』
- 「データウェアハウス・センター 管理ガイド」の『ユーザー定義プログラムとは?』
- 「データウェアハウス・センター 管理ガイド」の『プログラム・グループとは?』

ウェアハウス・タスク

データウェアハウスの作成には、次のタスクが関係しています。

- ソース・データ (または運用データ) を識別し、ウェアハウス・ソースとして使用するために定義する。
- ウェアハウスとして使用するデータベースを作成し、ウェアハウス・ターゲットを定義する。

- ウェアハウスで定義するプロセスのグループ用に、サブジェクト・エリアを定義する。
- プロセス内のステップを定義することにより、ウェアハウス・データベースにソース・データを移動して形式をトランスフォームする方法を指定する。
- 定義したステップをテストし、それらが自動的に実行されるようにスケジュールする。
- セキュリティーを定義し、「進行中の作業 (Work in Progress)」ノートブックを使用して、データベース使用をモニターすることにより、ウェアハウスを管理する。

DB2® Warehouse Manager がある場合は、ウェアハウス内にデータのインフォメーション・カタログを作成できます。インフォメーション・カタログは、業務メタデータを含むデータベースです。業務メタデータは、ユーザーが組織内で使用可能なデータと情報を識別し、見つけるのに役立ちます。データウェアハウス・メタデータはインフォメーション・カタログに公開できます。インフォメーション・カタログを検索して、ウェアハウス内で使用可能なデータを判別できます。

ウェアハウス内にデータのスタースキーマ・モデルを定義することもできます。スタースキーマとは、業務の各性質を説明する複数のディメンション表と、業務に関するファクトを含む 1 つのファクト表によって構成される特殊な設計です。たとえば、製造会社の場合、いくつかのディメンション表は、製品、マーケット、および時間を表します。ファクト表には季節ごとにそれぞれの地域で発注された製品の取引情報が含まれます。

関連概念:

- 「データウェアハウス・センター *管理ガイド*」の『ウェアハウス・ステップ』

関連タスク:

- 『ステップとタスク: データウェアハウス・センター・ヘルプ』

第 2 部 データベースの設計

第 4 章 論理データベースの設計

データベース設計の目標となるのは、自分の環境を、理解しやすくしかも将来の拡張の基礎となるよう表現したものを作ることです。また、データの一貫性や整合性を保ちやすいデータベース設計が望ましいと言えます。そのためには、設計の段階で冗長性を少なくし、データベースの更新時に発生する異常をなくす必要があります。

データベースの設計は、一度で終了するものではありません。多くの場合、何度かやり直すことが必要になります。

データベースに記録されること

データベース設計の最初の段階は、データベースの表に格納するデータの種類を決めることです。データベースには、組織や事業の中のエンティティー、またそれらの相互のリレーションシップを含めます。リレーショナル・データベースの場合、エンティティー は*表* として表されます。

エンティティー とは、格納する情報の基になる人、物、または概念のことです。サンプル表に示されているエンティティーには、従業員、部署、プロジェクトなどがあります。

サンプルの従業員表の場合、「従業員」というエンティティーには、従業員番号、名前、所属部署、給与といった*属性*、またはプロパティーが含まれています。こうしたプロパティーは、EMPNO、FIRSTNME、LASTNAME、WORKDEPT、および SALARY などの*列* で表します。

「従業員」というエンティティーの*オカレンス* は、一従業員に関するすべての列の値で成っています。各従業員にはユニークな従業員番号 (EMPNO) が付けられています。これは、「従業員」というエンティティーの各オカレンスを識別するためのものです。表の各行は、エンティティーまたはリレーションシップの各オカレンスを表します。たとえば、次の表の最初の行の値は、Haas という名前の従業員を表します。

表 4. 従業員のエンティティーとその属性のオカレンス

EMPNO	FIRSTNME	LASTNAME	WORKDEPT	JOB
000010	Christine	Haas	A00	President
000020	Michael	Thompson	B01	Manager
000120	Sean	O’Connell	A00	Clerk
000130	Dolores	Quintana	C01	Analyst
000030	Sally	Kwan	C01	Manager
000140	Heather	Nicholls	C01	Analyst
000170	Masatoshi	Yoshimura	D11	Designer

マルチメディアなど、従来なかったデータベース・アプリケーションをサポートする必要性が高まりつつあります。このため、文書、ビデオまたは混合メディア、イメージ、音声などのマルチメディア・オブジェクトをサポートする属性について考慮しておくとよいでしょう。

表の中で、ある行の各列はその行の他のすべての列と何らかの関連があります。サンプル表の中でのリレーションシップには、次のものがあります。

- 従業員は各部署に配属されている
  - Dolores Quintana は C01 部に配属されています。
- 従業員は何らかの仕事を行う
  - Dolores はアナリスト (Analyst) として働いています。
- 従業員は部署を管理している
  - Sally は C01 部を管理しています。

「従業員」と「部署」はエンティティー、Sally Kwan は「従業員」のオカレンスの 1 つ、 C01 は「部署」のオカレンスの 1 つです。表の各行の同じ列には同じリレーションシップが適用されます。たとえば、表の 1 つの行は Sally Kwan が C01 部を管理するリレーションシップを表し、別の行は、Sean O'Connell が A00 部の事務員 (clerk) であるリレーションシップを表しています。

表にどんな情報が入れられるかは、それによって表されるリレーションシップ、必要とされる柔軟性、および必要とされるデータ検索速度によって異なります。

企業内に存在するエンティティー・リレーションシップの識別に加えて、データに適用するビジネス・ルールなど、他のタイプの情報も識別する必要があります。

関連概念:

- 58 ページの『データベースのリレーションシップ』
- 61 ページの『列定義』

データベースのリレーションシップ

データベースでは、いくつかのタイプのリレーションシップが定義できます。従業員と部署との間のいろいろなリレーションシップについて考えてみましょう。

1 対多および多対 1 のリレーションシップ

従業員が 1 つの部署だけでしか働かない場合、このリレーションシップは従業員にとって*単一値* です。一方、1 つの部署には多数の従業員が含まれますので、このリレーションシップは部署にとって*複数値* になります。従業員 (単一値) と部署 (複数値) との間のリレーションシップは、 *1 対多* のリレーションシップになります。

1 対多および多対 1 のリレーションシップごとに、それぞれ表を定義するには、次のようにします。

1. 「多」の側のエンティティーが同じであるリレーションシップをすべて 1 つのグループにまとめます。
2. グループ内のすべてのリレーションシップをまとめて 1 つの表を定義します。

次の例では、1 番目と 2 番目のリレーションシップの「多」の側が「従業員」なので、従業員の表 EMPLOYEE が定義されます。

表 5. 多対 1 のリレーションシップ

エンティティー	リレーションシップ	エンティティー
従業員 (Employees)	割り当てられる (are assigned to)	部門 (departments)
従業員 (Employees)	働く (work at)	仕事 (jobs)
部門 (Departments)	報告する (report to)	(管理) 部門 ((administrative) departments)

3 番目のリレーションシップの場合、「部署」が「多」の側なので、部署の表 DEPARTMENT が定義されます。

これらさまざまなリレーションシップについて、次の表に示します。

EMPLOYEE 表:

EMPNO	WORKDEPT	JOB
000010	A00	President
000020	B01	Manager
000120	A00	Clerk
000130	C01	Analyst
000030	C01	Manager
000140	C01	Analyst
000170	D11	Designer

DEPARTMENT 表:

DEPTNO	ADMRDEPT
C01	A00
D01	A00
D11	D01

多対多のリレーションシップ

多対多のリレーションシップは、双方向に複数値のリレーションシップです。 1 人の従業員が複数のプロジェクトで働き、 1 つのプロジェクトに複数の従業員が加わっている場合がそれに当たります。「Dolores Quintana の仕事はなにか?」および「プロジェクト IF1000 でだれが働いているか?」という質問には、両方とも複数の答えがあります。多対多のリレーションシップは、エンティティー (「従業員」と「プロジェクト」) ごとに 1 つの列を割り当てた表にすることができます。次の例をご覧ください。

多対多のリレーションシップ (1 人の従業員が複数のプロジェクトで働き、 1 つのプロジェクトで複数の従業員が働く) がどのように表されるかを、次の表に示します。

従業員活動 (EMP_ACT) 表:

EMPNO	PROJNO
000030	IF1000
000030	IF2000
000130	IF1000
000140	IF2000
000250	AD3112

1 対 1 のリレーションシップ

1 対 1 のリレーションシップは、双方向に単一値のリレーションシップです。 1 人のマネージャーは 1 つの部署を管理し、 1 つの部署には 1 人のマネージャーしかいません。「C01 部の管理者はだれか?」および「Sally Kwan はどの部署を管理しているか?」という質問は、両方とも答えは 1 つです。このリレーションシップは、DEPARTMENT 表と EMPLOYEE 表のどちらにでも割り当てることができます。すべての部署にマネージャーがいますが、すべての従業員がマネージャーではないため、マネージャーは DEPARTMENT に追加する方が論理的です。次の例をご覧ください。

次の表には、1 対 1 のリレーションシップが示されています。

DEPARTMENT 表:

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

等しい値が必ず同じエンティティーを表すようにする

一連のエンティティーからなる同じ集合の属性を表す表は複数個作成できます。たとえば、従業員表で従業員が配属されている部署の番号を示し、部署表で各番号の部署にどのマネージャーが配属されているかを示します。両方の属性を同時に検索するには、次の例にあるとおり、一致する列で 2 つの表を結合させます。WORKDEPT と DEPTNO の値は同じエンティティーを表すものであり、部署表と従業員表との間の*結合パス* になります。

DEPARTMENT 表:

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
D21	Administration Support	000070	D01

EMPLOYEE 表:

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	JOB
000250	Daniel	Smith	D21	Clerk

あるエンティティーに関する情報を複数の表から検索する場合、等しい値が必ず同じエンティティーを表すようにしておく必要があります。接続する列には、互いに違う名前を使用できます (前の例では「WORKDEPT」と「DEPTNO」)。また、同じ名前を使用しても問題ありません (部署表とプロジェクト表では「DEPTNO」という列)。

関連概念:

- 57 ページの『データベースに記録されること』
- 61 ページの『列定義』

列定義

関係表の列を定義するには、次のようにします。

1. 列の名前を決定します。

   表の各列の名前は、その表でユニークであることが必要です。

2. その列に対してどのようなデータが有効かを指定します。

   データ・タイプ と*長さ* に、その列に有効なデータの種類と最大長を指定します。データ・タイプは、データベース・マネージャーに用意されているデータ・タイプの中から選択することもできますし、独自にユーザー定義のタイプを作成することもできます。

   データ・タイプのカテゴリーの例としては、数値、文字ストリング、 2 バイト (すなわち GRAPHIC) 文字ストリング、日時、および バイナリー・ストリングがあります。

   ラージ・オブジェクト (LOB) データ・タイプは、文書、ビデオ、イメージ、音声などのマルチメディア・オブジェクトをサポートします。これらのオブジェクトは、次のデータ・タイプを使用して実現されます。

   - *バイナリー・ラージ・オブジェクト* (BLOB) ストリング。 BLOB の例としては、従業員の写真、音声、ビデオがあります。
   - *文字ラージ・オブジェクト* (CLOB) ストリング。文字のシーケンスを 1 バイト文字かマルチバイト文字 (あるいは、その両方の組み合わせ) にすることができます。 CLOB の例としては、従業員の履歴書があります。
   - *2 バイト文字ラージ・オブジェクト* (DBCLOB) ストリング。文字列が 2 バイト文字のものです。 DBCLOB の例としては、日本語の履歴書があります。

   ユーザー定義タイプ (UDT) は、既存のタイプから派生したタイプです。既存のタイプから派生してそれと似たような特性でありながら、別個の非互換タイプと見なされるものを定義することが必要となる場合があります。

構造型 は、データベースで定義される構造を持つユーザー定義タイプです。構造型には、それぞれがデータ・タイプを持つ名前付き*属性* の順序列が含まれています。構造型は、別の構造型のサブタイプ として定義されることがありますが、この場合この別の構造型をスーパータイプ と呼びます。サブタイプは、そのスーパータイプのすべての属性を継承し、追加の属性を定義することもできます。共通のスーパータイプに関連する構造型のセットはタイプ*階層* と呼ばれ、スーパータイプを持たないスーパータイプはそのタイプ階層のルート・タイプ と呼ばれます。

構造型は、表またはビューのタイプとして使用することができます。構造型の属性の名前とデータ・タイプは、オブジェクト ID と共に、この*型付き表* または*型付きビュー* の列の名前とデータ・タイプになります。型付き表または型付きビューの行は、構造型のインスタンスの表していると考えることができます。

構造型は、表またはビューの列のデータ・タイプとして使用することはできません。アプリケーション・プログラムのホスト変数に、構造型インスタンス全体をリトリーブさせることもサポートされておりません。

参照タイプ は、構造型のコンパニオン・タイプです。特殊タイプと同じように、参照タイプは共通の表示を組み込みデータ・タイプの 1 つと共用するスカラー・タイプです。この同じ表示は、タイプ階層のすべてのタイプで共有されます。参照タイプの表示は、タイプ階層のルート・タイプが作成される時に定義されます。参照タイプを使用する時は、構造型タイプはそのタイプのパラメーターとして指定されます。このパラメーターは、その参照のターゲット・タイプ と呼ばれます。

参照のターゲットは、常に型付き表またはビューの行です。参照タイプを使用する時は、*有効範囲* を定義します。有効範囲は、参照値のターゲット行を含む表 (*ターゲット表* と呼ばれる)、またはビュー (*ターゲット・ビュー* と呼ばれる) を識別します。ターゲット表またはビューは、参照タイプのターゲット・タイプと同じタイプでなければなりません。有効範囲付きの参照タイプのインスタンスは、型付き表または型付きビューの行 (*参照タイプのターゲット行* と呼ばれる) を一意的に識別します。

ユーザー定義タイプを互いに比較したり変換したりするためのルーチンを呼び出すには、*ユーザー定義関数* (UDF) を使用できます。 UDF は、SQL の組み込み関数によって提供されるサポートに対して拡張および追加を行い、また組み込み関数が使える所ならどこでも使用できます。 UDF には次の 2 つの種類があります。

- 外部関数。プログラム言語で作成されたもの。
- ソース関数。他の UDF を呼び出すときに使用するもの。

たとえば、「ヨーロッパ式靴サイズ」と「アメリカ式靴サイズ」という 2 つの数値データ・タイプがあるとします。どちらのタイプも靴のサイズを表していますが、大きさの単位が違うために互換性がなく、そのままでは比較できません。ユーザー定義の関数を呼び出して、靴のサイズをどちらかに変換できます。

3. どの列にデフォルト値が必要かを指定します。

すべての行に意味のある値を入れることのできない列もあります。これは次の理由によります。

- その列の値が行に適用されない場合。

  たとえば、従業員のミドルネームのイニシャルを入れる列は、ミドルネームのない従業員には適用されません。

- 値は適用されるが、まだ分からない場合。

  たとえば、ある部署の前マネージャーが配属替えになり、新しいマネージャーがまだ決まっていない場合、マネージャー番号の列 (MGRNO) に有効なマネージャー番号が入っていないことがあるかもしれません。

いずれの場合でも、NULL 値 (列の値が分からないか適用されないことを示す特殊値) を入れるか、データベース・マネージャーやアプリケーションによって NULL 値ではないデフォルト値が割り当てられるようにするか、どちらかを選択できます。

主キー

キー は、特定の行 (単数または複数) を識別したりそれらにアクセスしたりするために使用可能な一連の列です。キーは、表、索引、または参照制約の記述で識別されます。同じ列を複数のキーの一部にすることができます。

ユニーク・キー は、 2 つの値が同じにならないように制限されているキーです。ユニーク・キーの列に NULL 値を含めることはできません。たとえば、従業員番号の列の各値は 1 人の従業員だけを指すものなので、これをユニーク・キーとして定義することができます。 2 人の従業員が同じ従業員番号を共用することはできません。

キーの固有性を実施させるために使用する機構をユニーク*索引* と呼びます。表のユニーク索引とは、それぞれの値がユニークな行を識別する (機能的に判別する)、 1 つの列または複数の列の順序集合のことです。ユニーク索引には NULL 値を含めることができます。

主キー は、 1 つの表上に定義されたユニーク・キーの 1 つですが、 1 番目に重要なキーとして選択されたものです。 1 つの表上には 1 つだけの主キーが可能です。

主キーに対する *1 次索引* は自動的に作成されます。 1 次索引は、データベース・マネージャーが表の行に効率的にアクセスするために使用するもので、データベース・マネージャーが主キーを固有のものにするためのものです。 (さらに、非主キー列にも索引を定義して、照会の処理時に効率的にデータにアクセスできます。)

表に「本来の」ユニーク・キーがない場合や、到着順によってユニークな行を区別する場合は、タイム・スタンプをキーの一部として使用するのが効果的な場合もあります。

サンプル表の一部に使用されている主キーは、次のとおりです。

表	キー列

従業員 (EMPLOYEE) 表	EMPNO
部署 (DEPARTMENT) 表	DEPTNO
プロジェクト (PROJECT) 表	PROJNO

PROJECT 表の一部に主キー列を示した例を次に示します。

表 6. PROJECT 表の主キー

PROJNO(主キー)	PROJNAME	DEPTNO
MA2100	Weld Line Automation	D01
MA2110	Weld Line Programming	D11

表のすべての列に重複値が含まれる場合、 1 つの列だけに主キーを定義することはできません。複数の列を使ったキーは複合キー と呼ばれます。列の値の組み合わせは、ユニークなエンティティーを定義するものでなければなりません。複合キーを簡単に定義することができない場合、ユニークな値を持つ新しい列を作成するのも 1 つの方法です。

次の例は、複数の列を含む主キー (複合キー) を示すものです。

表 7. EMP_ACT 表の複合主キー

EMPNO (主キー)	PROJNO (主キー)	ACTNO (主キー)	EMPTIME	EMSTDATE (主キー)
000250	AD3112	60	1.0	1982-01-01
000250	AD3112	60	.5	1982-02-01
000250	AD3112	70	.5	1982-02-01

候補キー列の識別

キーの候補を識別するには、ユニークなエンティティーを定義する最小限の列を選択してください。キー候補は複数個あるかもしれません。 表 8 には、キーの候補となるものがたくさん示されています。EMPNO、 PHONENO、および LASTNAME 列は、それぞれ従業員を固有に識別するものとなります。

表 8. EMPLOYEE 表

EMPNO (主キー)	FIRSTNAME	LASTNAME	WORKDEPT (外部キー)	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O’Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

いくつかのキーの候補から主キーを選択するときには、持続性、固有性、安定性を基準にします。

- 持続性は、それぞれの行の主キー値が常に存在することを意味します。

- 固有性は、それぞれの行のキー値が他のすべてのものと異なることを意味します。
- 安定性は、主キーが決して変化しないことを意味します。

例の中の 3 つの候補キーのうち、上記の基準すべてにかなうのは EMPNO だけです。従業員は入社時に電話番号を持っていないかもしれません。名字は変わる可能性があるため、ある時点でユニークであっても、そのことが保証されているわけではありません。主キーとしては従業員番号列が最善であると言えます。従業員には一度だけユニークな番号が与えられ、大抵は会社をやめない限りその番号が変更になることはありません。各従業員は必ず番号を持つため、従業員番号列の値には持続性があります。

関連概念:

- 65 ページの『ID (Identity) 列』

ID (Identity) 列

ID 列 は、DB2® Universal Database (DB2 UDB) が自動的に表の各行にユニークな数値を生成する手段となります。表には、ID 属性が定義されている単一列を入れることができます。 ID 列の例には、オーダー番号、従業員番号、在庫番号、および問題番号があります。

ID 列の値は常に生成するか、デフォルトで生成することができます。

- *常に生成する (generated always)* として定義されている ID 列は、 DB2 UDB によってユニークであることが保証されています。この値は常に DB2 UDB によって生成されます。アプリケーションは明示的な値を提供することが許可されていません。
- *デフォルトで作成する (generated by default)* として定義されている ID 列は、アプリケーションが ID 列の値を明示的に提供する手段となります。値が指定されていない場合は DB2 UDB がその値を生成しますが、この場合は、値の固有性は保証されていません。 DB2 UDB が固有性を保証するのは、DB2 が生成した一連の値についてのみです。デフォルトによる生成は、データ伝搬 (このとき、既存の表の内容がコピーされる) か、表のアンロードおよび再ロードのために使用するものです。

ID 列は、ユニークな主キー値を生成するタスクに理想的なものです。アプリケーションは ID 列を使用することにより、データベースの外で独自のユニーク・カウンターが生成されたときに生じる可能性がある並列処理の問題やパフォーマンス上の問題を避けることができます。たとえば、一般的な 1 つのアプリケーション・レベル実装では、カウンターを含む 1 行の表を保守します。それぞれのトランザクションはこの表をロックし、数値を増分してから、コミットします。つまり、一度に 1 つのトランザクションのみがカウンターを増分できます。対照的に、カウンターが ID 列を介して保守されている場合、カウンターはトランザクションによりロックされないため、さらに高いレベルの並列処理を実現することができます。カウンターを増分したトランザクションがコミットされていなくても、続くトランザクションがカウンターを増分できなくなることはありません。

ID 列のカウンターは、トランザクションから独立して増分 (または減分) されます。特定のトランザクションが ID カウンターを 2 回増分した場合、同時に他のトランザクションが同じ ID カウンターを増分している (つまり、行を同じ表に挿入している) 可能性があるため、このトランザクションで生成された 2 つの数の間にギャップが検出されることがあります。アプリケーションに連続する一連の数が必要な場合、そのアプリケーションは ID 列がある表に排他ロックを行う必要があります。この決定は、結果として並列処理が失われることを比較検討して行わなければなりません。さらに、ID 列の値を生成したトランザクションがロールバックしたため、または値の範囲をキャッシュしたデータベースが、キャッシュされたすべての値が割り当てられる前に非活動化されたため、特定の ID 列の数値にギャップが生じたように見えることがあります。

ID 列によって生成された順次数値には、次の付加的な特性があります。

- 値は、スケールがゼロの厳密な数データ・タイプにすることができます。つまり、スケールがゼロの SMALLINT、INTEGER、BIGINT、または DECIMAL です。 (単精度および倍精度の浮動小数点数は、おおよその数値データ・タイプと見なされます。)
- 連続値は、指定した整数増分値によって異なる場合があります。デフォルトの増分値は 1 です。
- ID 列のカウンター値はリカバリー可能です。障害が生じた場合、カウンター値はログから再構成されるため、ユニークな値が引き続き生成されることが保証されます。
- ID 列値をキャッシュすると、パフォーマンスが向上します。

関連概念:

- 63 ページの『主キー』

正規化

正規化 は、表データの冗長度と矛盾を除去するのに役立ちます。これは、キー列以外の列が主キー列に依存したものになるような列の集合に表を整理するプロセスです。そうでない場合、データ更新時に矛盾が生じることがあります。

このセクションでは、第 1、第 2、第 3、第 4 正規形の規則について簡単に説明します。第 5 正規形の表については、データベース設計に関する多くの本の中で取り上げられており、ここでは説明しません。

形式	説明
第 1	表の各行-列の位置には、値の集合ではなく、*1* つの値が存在します。
第 2	キーの一部ではない各列は、キーに従属しています。
第 3	それぞれの非キー列は、他の非キー列からは独立しており、キーにのみ従属しています。
第 4	どの行にも、あるエンティティーに関する複数の独立した複数値情報は含まれません。

第 1 正規形

各セルに値が 1 つしかない (値のセットでない) 場合、その表は*第 1 正規形* です。第 1 正規形の表は、それより上位の正規形の基準を満たしているとは限りません。

たとえば、次の表の場合、「WAREHOUSE」列には「PART」の各オカレンスごとに複数の値が入っているため、第 1 正規形に違反しています。

表 9. 第 1 正規形に違反する表

PART (主キー)	WAREHOUSE
P0010	Warehouse A, Warehouse B, Warehouse C
P0020	Warehouse B, Warehouse D

次の例は、第 1 正規形と同じ表です。

表 10. 第 1 正規形に適合する表

PART (主キー)	WAREHOUSE (主キー)	QUANTITY
P0010	Warehouse A	400
P0010	Warehouse B	543
P0010	Warehouse C	329
P0020	Warehouse B	200
P0020	Warehouse D	278

第 2 正規形

キーでない各列に入っている情報がキー*全体* に依存している場合、その表は*第 2 正規形* です。

キー列以外の列が複合キーの*一部* に従属している場合、第 2 正規形に違反することになります。次の例をご覧ください。

表 11. 第 2 正規形に違反している表

PART (主キー)	WAREHOUSE (主キー)	QUANTITY	WAREHOUSE_ADDRESS
P0010	Warehouse A	400	1608 New Field Road
P0010	Warehouse B	543	4141 Greenway Drive
P0010	Warehouse C	329	171 Pine Lane
P0020	Warehouse B	200	4141 Greenway Drive
P0020	Warehouse D	278	800 Massey Street

主キーは複合キーであり、「PART」(部品) 列と「WAREHOUSE」(倉庫) 列によって構成されています。「WAREHOUSE_ADDRESS」列 (倉庫の住所) は「WAREHOUSE」の値だけに依存するものであるため、この表は第 2 正規形の規則に違反しています。

次の点が、この設計の問題点となっています。

- 倉庫の住所が、倉庫に保管されている部品のレコードごとに繰り返されています。
- 倉庫の住所が変更になった場合、その倉庫に保管されている部品に関係するすべての行を更新する必要があります。
- データのこの冗長性のために同じ倉庫のレコードでも住所が違う、というデータの矛盾が発生する可能性があります。
- ある時点で倉庫に保管されている部品がないということがあると、倉庫の住所を記録する行がなくなってしまいます。

解決策は、表を次の 2 つの表に分割することです。

表 12. 第 2 正規形に適合する部品在庫 (PART_STOCK) 表

PART (主キー)	WAREHOUSE (主キー)	QUANTITY
P0010	Warehouse A	400
P0010	Warehouse B	543
P0010	Warehouse C	329
P0020	Warehouse B	200
P0020	Warehouse D	278

表 13. 第 2 正規形に適合する倉庫 (WAREHOUSE) 表

WAREHOUSE (主キー)	WAREHOUSE_ADDRESS
Warehouse A	1608 New Field Road
Warehouse B	4141 Greenway Drive
Warehouse C	171 Pine Lane
Warehouse D	800 Massey Street

第 2 正規形の表が 2 つになると、パフォーマンスの面で考慮するべき点が出てきます。部品の保管場所のレポートを作成するアプリケーションでは、 2 つの表を結合させて関係する情報を検索することが必要になります。

第 3 正規形

キー列でない列に入っている情報が、キー列でない他の列とは無関係でキー列にしか依存しないものであるなら、その表は第 3 正規形です。

次の例の最初の表には、「EMPNO」列と「WORKDEPT」列が含まれています。ここに、「DEPTNAME」列を追加するとしましょう (69 ページの表 15 を参照)。新しい列は WORKDEPT に依存しますが、主キーは EMPNO です。この表は第 3 正規形に違反します。 John Parker という従業員の「DEPTNAME」を変えても、その部署の他の従業員の部署名は変わりません。部署番号 E11 に 2 つの部署名が使われることになっていることに注意してください。更新後の表に結果として矛盾が生じます。

表 14. 更新前の正規化されていない EMPLOYEE_DEPARTMENT 表

EMPNO (主キー)	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Operations
000320	Ramlal	Mehta	E21	Software Support
000310	Maude	Setright	E11	Operations

表 15. 更新後の正規化されていない EMPLOYEE_DEPARTMENT 表： 表の情報に矛盾が生じています。

EMPNO (主キー)	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Installation Mgmt
000320	Ramlal	Mehta	E21	Software Support
000310	Maude	Setright	E11	Operations

「WORKDEPT」と「DEPTNAME」を列とする表を新たに作成すれば、表を正規化できます。部署名の変更などの更新はさらに容易です。新しい表のみを更新する必要があります。

部署名と従業員名の両方を戻す SQL 照会は、 2 つの表を結合させることが必要になるため、少し複雑になります。またこの照会は、1 つの表の照会の場合よりも実行時間が長くなります。さらに、両方の表に「WORKDEPT」列が含まれることになるため、付加的なストレージ・スペースが必要になります。

次の表は、正規化した結果として定義されるものです。

表 16. 「従業員 - 部署」(EMPLOYEE_DEPARTMENT) 表を正規化した後の従業員 (EMPLOYEE) 表

EMPNO (主キー)	FIRSTNAME	LASTNAME	WORKDEPT
000290	John	Parker	E11
000320	Ramlal	Mehta	E21
000310	Maude	Setright	E11

表 17. 「従業員 - 部署」(EMPLOYEE_DEPARTMENT) 表を正規化した後の部署 (DEPARTMENT) 表

DEPTNO (主キー)	DEPTNAME
E11	Operations
E21	Software Support

第 4 正規形

どの行にも 1 つのエンティティーに関して 2 つ以上の独立した複数値情報が含まれていない場合、その表は第 4 正規形です。

従業員、技術、言語の 3 種類のエンティティーについて考えてみましょう。 1 人の従業員が複数の技術を持っていたり、複数の言語を知っていたりするかもしれません。この場合には、従業員と技術、従業員と言語という 2 通りのリレーションシ

ップがあります。 1 つの表でその両方のリレーションシップを表すとすれば、その表は第 4 正規形ではありません。次の例をご覧ください。

表 18. 第 4 正規形に違反する表

EMPNO (主キー)	SKILL (主キー)	LANGUAGE (主キー)
000130	Data Modelling	English
000130	Database Design	English
000130	Application Design	English
000130	Data Modelling	Spanish
000130	Database Design	Spanish
000130	Application Design	Spanish

むしろ、これらのリレーションシップは 2 つの表で表すべきです。

表 19. 第 4 正規形に適合している EMPLOYEE_SKILL 表

EMPNO (主キー)	SKILL (主キー)
000130	Data Modelling
000130	Database Design
000130	Application Design

表 20. 第 4 正規形に適合している EMPLOYEE_LANGUAGE 表

EMPNO (主キー)	LANGUAGE (主キー)
000130	English
000130	Spanish

しかし、属性が相互依存の関係にある場合 (つまり従業員がある言語を特定の技術だけに使用するような場合)、表を分割するべきではありません。

データベースを設計する際は、まずすべてのデータを第 4 正規形の表にまとめ、それからパフォーマンスが許容できるレベルかどうかを判断するのが得策です。パフォーマンスが許容できない場合は、表のデータを第 3 正規形に配列してパフォーマンスを再評価します。

制約

制約 は、データベース・マネージャーが実施する規則です。

制約には、以下の 4 つのタイプがあります。

- *ユニーク制約* は、表の中の 1 つまたは複数の列での重複値を禁止する規則です。ユニーク制約では、ユニーク・キーと主キーがサポートされています。たとえば、 2 つの製造業者に同じ製造業者 ID が絶対に与えられないようにするために、製造業者表の製造業者 ID にユニーク制約を定義することができます。
- *参照制約* は、 1 つ以上の表の中の 1 つ以上の列での値に関する論理規則です。たとえば、表集合は企業の製造業者に関する情報を共有します。ときには、製造業者の名前が変わることもあります。表の製造業者の ID が、製造業者情報の製

造業者 ID と一致していなければならないことを示す、参照制約を定義できます。これにより、そうしなければ製造業者情報の消失に至るような、挿入、更新、削除操作が防ぐことができます。

- *表チェック制約* は、特定の表に追加されるデータに制約を設定します。たとえば、表チェック制約では、個人情報を含む表でデータが追加または更新される場合に、必ず従業員の給与レベルが $20,000 より低くならないようにできます。
- *情報制約* は、 SQL コンパイラーによって使用されますが、データベース・マネージャーによっては施行されない規則です。

参照制約および表チェック制約は、オンまたはオフに切り替えることができます。たとえば、大量のデータがデータベースにロードされる場合、制約の実施をオフにする方が一般に賢明です。

ユニーク制約

ユニーク制約 は、キーの値が表内でユニークな場合にのみ、その値が有効になるという規則です。ユニーク制約はオプショナルであり、 PRIMARY KEY 文節または UNIQUE 文節を使用して、 CREATE TABLE または ALTER TABLE ステートメントで定義できます。ユニーク制約で指定される列は、 NOT NULL として定義されていなければなりません。データベース・マネージャーはユニーク索引を使用して、ユニーク制約の列への変更の際にキーのユニーク性を固持します。

1 つの表で、任意の数のユニーク制約を定義できます。ただし、複数のユニーク制約を主キーとして定義することはできません。 1 つの表で、同じ列のセット上に複数のユニーク制約を持つことはできません。

参照制約の外部キーによって参照されるユニーク制約を、*親キー* といいます。

ユニーク制約が CREATE TABLE ステートメントで定義されている場合、データベース・マネージャーによってユニーク索引が自動的に作成され、 1 次索引またはユニーク・システム必須索引として指定されます。

ユニーク制約が ALTER TABLE ステートメントで定義されており、索引が同じ列にある場合、その索引はユニークおよびシステム必須として指定されます。そのような索引が存在しない場合、データベース・マネージャーによってユニーク索引が自動的に作成され、 1 次索引またはユニーク・システム必須索引として指定されます。

ユニーク制約の定義とユニーク索引の作成には、違いがあるので注意してください。両方とも制約を実施しますが、ユニーク索引では NULL 可能列が使用できるため、一般に親キーとしては使用できません。

参照制約

参照保全 とは、すべての外部キーのすべての値が有効である、データベースの状態です。*外部キー* は、親表の行の 1 つ以上の主キーまたはユニーク・キー値と値が一致していなければならない、表内の列または列のセットです。*参照制約* は、以下の条件のいずれかが当てはまる場合だけ、外部キーの値が有効になる規則です。

- それらが親キーの値となっている。
- 外部キーの何らかの部分が NULL である。

親キーを含んでいる表を参照制約の*親表* といい、外部キーを含んでいる表を表の*従属* といいます。

参照制約はオプショナルであり、 CREATE TABLE ステートメントまたは ALTER TABLE ステートメントで定義できます。参照制約は、INSERT、UPDATE、DELETE、ALTER TABLE、ADD CONSTRAINT、および SET INTEGRITY ステートメントの実行中に、データベース・マネージャーによって実施されます。

RESTRICT の削除または更新規則が指定されている参照制約は、他のすべての参照制約の前に実施されます。 NO ACTION の削除または更新規則が指定されている参照制約は、ほとんどの場合、 RESTRICT のように動作します。

参照制約、チェック制約、およびトリガーは結合できることを覚えておいてください。

参照保全規則には、以下の概念および用語が関係します。

親キー 親キーとは、参照制約の主キーまたはユニーク・キーのことです。

親行 1 つ以上の従属行を持つ行。

親表 参照制約の親キーを含む表。 1 つの表が、任意の数の参照制約において親となることが可能です。参照制約の中の親である表が、参照制約の中の従属になることもできます。

従属表 定義で 1 つ以上の参照制約を含む表。 1 つの表が、任意の数の参照制約において従属となることが可能です。参照制約の中の従属である表が、参照制約の中の親になることもできます。

下層表 ある表が表 T の下層であるとは、それが T の従属であるか、または T の従属の下層であるということです。

従属行 1 つ以上の親行を持つ行。

下層行 ある行が行 r の下層であるとは、それが r の従属であるか、または r の従属の下層であるということです。

参照循環
セット内の各表がそれ自身の下層であるような、参照制約のセット。

自己参照表
同じ参照制約内の親および従属である表。この制約を、*自己参照制約* といいます。

自己参照行
それ自身の親である行。

挿入規則

参照制約の挿入規則とは、外部キーの非 NULL の挿入値が、親表の親キーの何らかの値に一致しなければならないというものです。複合外部キーの値は、値のいずれかの部分が NULL であれば、NULL になります。これは、外部キーが指定された場合は暗黙の規則になります。

更新規則

参照制約の更新規則は、参照制約の定義時に指定されます。選択項目は NO ACTION および RESTRICT です。更新規則は、親の行または従属表の行の更新時に適用されます。

親行の場合、親キーの列内の値の更新時に以下の規則が適用されます。

- 従属表のいずれかの行がキーのオリジナルの値と一致する場合、更新規則が RESTRICT であると更新はリジェクトされます。
- 更新ステートメントが完了したときに (トリガーの後を除く) 従属表のどの行も対応する親キーを持っていない場合、更新規則が NO ACTION であると更新はリジェクトされます。

従属行の場合、外部キーの指定時に NO ACTION 更新規則が暗黙的に適用されます。 NO ACTION とは、更新ステートメントの完了時に、外部キーの非 NULL の更新値が、親表の親キーの何らかの値に一致しなければならないというものです。

複合外部キーの値は、値のいずれかの部分が NULL であれば、NULL になります。

削除規則

参照制約の削除規則は、参照制約の定義時に指定されます。選択項目は NO ACTION、RESTRICT、CASCADE、または SET NULL です。 SET NULL を指定できるのは、外部キーの何らかの列で NULL 値が可能な場合だけです。

参照制約の削除規則は、親表の行の削除時に適用されます。より正確には、親表の行が削除または伝搬された削除操作 (以下で定義する) の対象であり、その行に参照制約の従属表がある場合に、その規則が適用されます。 P が親表、D が従属表、および p が削除または伝搬された削除操作の対象の親行である、以下の例を考慮してください。削除規則は以下のように機能します。

- RESTRICT または NO ACTION では、エラーが発生し、行が削除されません。
- CASCADE では、削除操作が表 D の従属に伝搬されます。
- SET NULL では、表 D 内の p の各従属の外部キーの各 NULL 可能列が、NULL に設定されます。

表が親である各参照制約にはそれ自体の削除規則があり、削除操作の結果を判別するために、すべての適用可能な削除規則が使用されます。そのため、行に RESTRICT または NO ACTION の削除規則が指定されている参照制約の従属がある場合、または RESTRICT または NO ACTION の削除規則が指定されている参照制約の従属である、いずれかの下層への削除カスケードがある場合、その行は削除できません。

親表 P からの行の削除には他の表も関係しており、以下の表の行に影響する場合があります。

- 表 D が P の従属であり、削除規則が RESTRICT または NO ACTION である場合、その操作に D が関係するものの、操作によって影響は受けません。
- D が P の従属であり、削除規則が SET NULL である場合、その操作に D が関係し、その操作中に D の行が更新される場合があります。

- D が P の従属であり、削除規則が CASCADE である場合、その操作に D が関係し、その操作中に D の行が削除される場合があります。

  D の行が削除される場合、 P での削除操作を D への伝搬といいます。 D が親表でもある場合、今度は、ここにリストされているアクションが D の従属にも適用されます。

P での削除操作に関係する可能性のあるすべての表を、 P への*連結削除* といいます。そのため、表が P の従属である場合、または P から削除操作のカスケードが行われる表の従属である場合、表は表 P への連結削除になります。

連結削除の関係には、以下の制約事項が適用されます。

- 表が複数の表の参照循環の中でそれ自身に対して連結削除されている場合、その循環には RESTRICT または SET NULL の削除規則が含まれていてはなりません。
- 表は、CASCADE 関係 (自己参照または他の表を参照) の従属表であってはならず、 RESTRICT または SET NULL の削除規則を持つ自己参照関係を持ってはなりません。
- 表が、外部キーが重複する複数の関係で他の表に連結削除されている場合、これらの関係は同じ削除規則を持たなければならず、 SET NULL に設定されたものがあってはなりません。
- 複数の関係の 1 つに削除規則 SET NULL が指定されているときに、それらの関係の中で表が他の表に連結削除されている場合、この関係の外部キーの定義にはパーティション・キーまたは MDC キー列が含まれていてはなりません。
- 2 つの表が同じ表に CASCADE 関係を通して連結削除されている場合、削除接続されているパスが削除規則 RESTRICT または SET NULL で終了しているときには、それら 2 つの表を相互に連結削除してはなりません。

表チェック制約

表チェック制約 は、表の各行にある 1 つまたは複数の列で許可される値を指定するための規則です。制約はオプショナルであり、 CREATE TABLE または ALTER TABLE ステートメントを使用して定義できます。表チェック制約の指定は、検索条件の制限付きフォームによって行われます。制約の 1 つは、表 T での表チェック制約の列名が、表 T の列を示していなければならないというものです。

1 つの表で、任意の数の表チェック制約を定義できます。表チェック制約は、挿入または更新される各行へ検索条件を適用することにより、実施されます。何らかの行で検索条件の結果が false の場合、エラー検索が発生します。

既存のデータがある表で、 ALTER TABLE ステートメントに 1 つ以上の表チェック制約が定義されている場合、 ALTER TABLE ステートメントが完了する前に、既存のデータが新しい条件と比べてチェックされます。 SET INTEGRITY ステートメントを使用して、表をチェック・ペンディング 状態にすることができます。これにより、データの検査を行わずに、 ALTER TABLE ステートメントを続行できます。

情報制約

情報制約 は、データへのアクセス・パスを改良するために SQL コンパイラーによって使用される規則です。情報制約は、データベース・マネージャーによって施行されるものではなく、データの追加の検査のために使用されるものではありません。むしろ、照会のパフォーマンスを改良するために使用されます。

CREATE TABLE または ALTER TABLE ステートメントを使用して参照または表チェック制約を定義する場合には、データベース・マネージャーが制約を施行するかどうか、および制約は照会の最適化に使用されるかどうかについて決定する制約属性を指定します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET INTEGRITY ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『トリガーと制約の相互作用』

トリガー

トリガー は、指定した表に対する挿入、更新、または削除操作への応答として実行されるアクションのセットを定義します。そのような SQL 操作が実行されるときに、トリガーが*活動化された* といいます。

トリガーはオプションであり、 CREATE TRIGGER ステートメントを使用して定義されます。

データ保全性規則を実施するために、参照制約およびチェック制約とともにトリガーを使用できます。また、トリガーを使用して、他の表への更新を行ったり、挿入または更新される行の値を自動的に生成またはトランスフォームできます。あるいは、関数を呼び出してタスク (アラートを発するなど) を実行することもできます。

トリガーは、*遷移* メカニズム・ルールを定義および実施するための便利な機能です。この規則は、さまざまな状態のデータ (たとえば、増加率が必ず 10 % 以下である給与など) を扱う規則です。

トリガーを使用すると、ビジネス規則を実施するロジックをデータベース内に置くことができます。つまり、アプリケーションがそれらの規則の実施を担当しないということです。ロジックを変更する場合に、アプリケーション・プログラムへの変更が必要ないため、すべての表に実施される集中化したロジックにより、保守が容易になります。

トリガーを作成する際に、以下を指定します。

- *サブジェクト表*。これは、トリガーが定義される表を指定します。
- *トリガー・イベント*。これは、サブジェクト表を変更する特定の SQL 操作を定義します。イベントには、挿入、更新、または削除操作があります。
- *トリガー起動時間*。これは、トリガー・イベントが発生する前か後のどちらで、トリガーを活動化するかを指定します。

トリガーを活動化するステートメントには、*影響を受ける行のセット* が含まれます。これらは、挿入、更新、または削除されるサブジェクト表の行です。*トリガー*

細分性 では、トリガーのアクションの実行をステートメントで 1 回か、または影響を受ける行ごとに 1 回かを指定します。

トリガー・アクション は、オプションの検索条件と、トリガーが活動化されるたびに実行される SQL ステートメントのセットから成り立っています。 SQL ステートメントが実行されるのは、検索条件が true として評価される場合だけです。トリガー起動時間がトリガー・イベントの前の場合、トリガー・アクションに、選択を行うステートメント、set 遷移変数、またはシグナル SQLSTATE を含めることができます。トリガー起動時間がトリガー・イベントの後の場合、トリガー・アクションに、選択、挿入、更新、削除を行うステートメント、またはシグナル SQLSTATE を含めることができます。

トリガー・アクションでは、*遷移変数* を使用して、影響を受ける行のセット内の値を参照できます。遷移変数は、サブジェクト表の列の名前を使用します。この名前は、参照が古い値 (更新前) か新しい値 (更新後) かを示す、指定された名前によって修飾されます。挿入または更新トリガーの前に、 SET Variable ステートメントを使用して新しい値を変更することもできます。

影響を受ける行のセット内の値を参照する別の方法は、*遷移表* を使用することです。遷移表では、サブジェクト表の列の名前も使用しますが、名前を指定することにより、影響を受ける行の完全なセットを表として扱うことができます。遷移表を使用できるのはトリガーの後だけであり、古い値と新しい値に、それぞれ別個の遷移表を定義できます。

表、イベント、起動時間の組み合わせで、複数のトリガーを指定できます。トリガーが活動化される順序は、作成された順序と同じです。そのため、一番あとに作成されたトリガーが、最後に活動化されます。

トリガーの活動化では、*トリガー・カスケード* が行われる場合があります。これは、 SQL ステートメントを実行するあるトリガーを活動化することにより、その SQL ステートメントによって、他のトリガーが活動化されるか、または同じトリガーが再度活動化された結果です。トリガー・アクションによって、削除の参照保全のアプリケーションの結果である更新が行われることもあります、これにより、今度は、追加トリガーの活動化が行われる場合があります。トリガー・カスケードにより、トリガーおよび参照保全の削除規則のチェーンが活動化され、単一の INSERT、UPDATE、または DELETE ステートメントの結果として、データベースへの大幅な変更が行われる場合があります。

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『トリガーと制約の相互作用』

追加のデータベース設計に関する考慮事項

データベースの設計時には、ユーザーがどの表にアクセスできるかを考えるのは重要なことです。表へのアクセス権の付与または取り消しは、権限許可によって行われます。最高の権限は、システム管理権限 (SYSADM) です。 SYSADM 権限のあるユーザーは、データベース管理者権限 (DBADM) を含め、その他の権限許可を割り当てることができます。

監査 を行うには、一定期間にデータに対して加えられた更新事項をすべて記録しておくことが必要になります。たとえば、従業員の給与が変更されるたびに、監査表を更新するのはよいことです。この表の更新は、適切なトリガーを定義しておけば、自動的に行われます。監査活動は、DB2® Universal Database (DB2 UDB) 監査機能でも行えます。

パフォーマンス上の理由で、*履歴* として基本データの維持を実行している間は、選択された量のデータだけをアクセスしたい場合があります。設計の中には、履歴データの維持に関して、データをどれくらいの期間保存しておくことが必要かという点など、必要な情報を含めるべきです。

さらに、*サマリー* 情報を利用することもできます。たとえば、すべての従業員の情報を含む表があるとします。しかし、この情報を部門や所属別に別々の表に分割したいこともあるでしょう。この場合、元の表のデータに基づく、部門ごとまたは所属ごとのマテリアライズ照会表が役立ちます。

セキュリティー の影響についても、設計中に識別するべきです。たとえば、ある種のデータに対するユーザー・アクセスをセキュリティー表によってサポートするとしましょう。各種のデータに対するアクセス・レベルやだれがアクセスできるかといった点を定義できます。従業員および給与計算データなどの機密データには、最も厳重なセキュリティー制限があります。

表に関連した*構造型* を持つ表を作成することができます。そのような型付き表については、*タイプ階層* と呼ばれる表の間で定義されたリレーションシップを使用して、階層構造を設定できます。タイプ階層は、単一のルート・タイプ、スーパータイプ、およびサブタイプで構成されます。

参照タイプ の表示は、タイプ階層のルート・タイプが作成される時に定義されます。参照のターゲットは、常に型付き表またはビューの行です。

高可用性災害時リカバリー (HADR) 環境で作業している場合、いくつかの推奨事項があります。

- HADR 環境の 2 つのインスタンスは、ハードウェアについてもソフトウェアについても同一でなければなりません。つまり、
  - HADR の主データベースのホスト・コンピューターとスタンバイ・データベースのホスト・コンピューターは、同一でなければなりません。
  - 主システムとスタンバイのオペレーティング・システムは、パッチも含めて同じバージョンでなければなりません。 (アップグレード中には、そのようにできない場合があります。しかし、相違が生じることによる問題をなるべく少なくするため、インスタンスが一致していない期間は可能な限り短くしてください。相違があることによって生じる可能性のある問題には、通常の非フェイルオーバー操作に影響する問題に加えて、フェイルオーバー中の新しいフィーチャーのサポートが失われることがあります。)
  - 2 つの HADR インスタンスの間で TCP/IP インターフェースが使用可能でなければなりません。
  - 高速大容量のネットワークをお勧めします。
- 考慮しなければならない DB2 UDB の要件があります。

- 主側とスタンバイ側で使用するデータベースのリリースは、同一でなければなりません。
- 主側とスタンバイ側の DB2 UDB ソフトウェアは、ビット・サイズが同じでなければなりません。 (つまり両方とも 32 ビットか、両方とも 64 ビットでなければなりません。)
- 表スペースとそれに対応するコンテナーは、主データベースとスタンバイ・データベースで同一でなければなりません。表スペースについて対称でなければならない特性には、表スペースのタイプ (DMS か SMS か)、表スペースのサイズ、コンテナーのパス、コンテナーのサイズ、コンテナーのファイル・タイプ (ロー・デバイスかファイル・システムか)、などがあります。相対コンテナー・パスも使用できますが、その場合、その相対パスは各インスタンスで同じでなければなりません。それらのマップ先の絶対パスは、同じであっても違っていても問題ありません。
- 主データベースとスタンバイ・データベースのデータベース・パスは、同じである必要はありません (CREATE DATABASE コマンド使用時の宣言による)。
- 主データベースでのバッファー・プールの活動が、スタンバイにおいても再現されます。このことは、主データベースとスタンバイ・データベースのメモリー量が同じであることの重要性を示唆しています。

第 5 章 物理データベースの設計

b 論理データベースの設計が完了したら、データベースや表を組み込む物理的な環境
b について考慮するべき点がいくつかあります。データベースのサポートや管理のた
b めに作成されるファイル、データを格納するのに必要なスペースの大きさ、データ
b の格納に必要な表スペースの使用法、さらにデータを入れるのに使用する表の構造
b について考慮しなければなりません。

データベース・ディレクトリーおよびファイル

データベースを作成するとき、デフォルトの情報を含むデータベースに関する情報は、ディレクトリー階層内に保管されます。階層ディレクトリー構造は、 CREATE DATABASE コマンド内に提供する情報によって判別されるロケーションに作成されます。データベースを作成するときにディレクトリー・パスまたはドライブのロケーションを指定しない場合、デフォルトのロケーションが使用されます。

データベースを作成したいロケーションを明示的に示すことをお勧めします。

CREATE DATABASE コマンドで指定するディレクトリー内に、インスタンスの名前を使用するサブディレクトリーが作成されます。このサブディレクトリーにより、同じディレクトリーの下に異なるインスタンスで作成されたデータベースが同じパスを使用しないようにします。インスタンス名のサブディレクトリーの下に、NODE0000 というサブディレクトリーが作成されます。このサブディレクトリーは、論理的にパーティション化されたデータベース環境内のパーティションを区別します。ノード名のディレクトリーの下に、SQL00001 というサブディレクトリーが作成されます。このサブディレクトリーのこの名前はデータベース・トークンを使用し、作成されているデータベースを表します。 SQL00001 には、作成された 1 番目のデータベースに関連するオブジェクトが含まれ、 2 番目以降のデータベースについては SQL00002 というように、より大きな数値が順次与えられます。これらのサブディレクトリーは、 CREATE DATABASE コマンドで指定したディレクトリー上のこのインスタンスで作成されたデータベースを区別します。

以下のように、ディレクトリー構造が表示されます。

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

データベース・ディレクトリーには、CREATE DATABASE コマンドの一部として作成される以下のファイルが含まれます。

- ファイル SQLBP.1 および SQLBP.2 には、バッファー・プール情報が含まれています。各ファイルには、バックアップを提供するための複製コピーがあります。
- ファイル SQLSPCS.1 および SQLSPCS.2 には、表スペース情報が含まれています。各ファイルには、バックアップを提供するための複製コピーがあります。
- SQLDBCON ファイルには、データベース構成情報が入っています。このファイルを編集しないでください。構成パラメーターを変更するには、コントロール・

センターまたはコマンド行ステートメント UPDATE DATABASE CONFIGURATION および RESET DATABASE CONFIGURATION のどちらかを使用します。

- DB2RHIST.ASC 履歴ファイルおよびそのバックアップ DB2RHIST.BAK には、バックアップ、リストア、表のロード、表の再編成、表スペースの変更、およびデータベースへの他の変更についての履歴情報が含まれています。

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

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

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

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

SMS データベース・ディレクトリーについての追加情報

SQLT* サブディレクトリーには、 1 つのデータベースが作動するために必要なデフォルトのシステム管理スペース (SMS) 表スペースが含まれます。デフォルトの表スペースは 3 つ作成されます。

- SQLT0000.0 サブディレクトリーには、システム・カタログ表のカタログ表スペースが含まれます。
- SQLT0001.0 サブディレクトリーには、デフォルト 一時表スペースが含まれます。

- SQLT0002.0 サブディレクトリーには、デフォルト・ユーザー・データ表スペースが含まれます。

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

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

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

関連概念:

- 126 ページの『SMS 表スペースと DMS 表スペースの比較』
- 「*管理ガイド: パフォーマンス*」の『DMS 装置に関する考慮事項』
- 「*管理ガイド: パフォーマンス*」の『SMS 表スペース』
- 「*管理ガイド: パフォーマンス*」の『DMS 表スペース』
- 「*管理ガイド: パフォーマンス*」の『DMS 表スペース・アドレス・マップの図』
- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『リカバリー履歴ファイルについて』

関連資料:

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

データベース・オブジェクトのスペース所要量

データベース・オブジェクトのサイズは、正確に見積もることができません。サイズの見積もりを難しくする原因は、ディスクのフラグメント化によって発生するオーバーヘッド、フリー・スペース、および可変長列の使用などです。これは、列タイプや行の長さが広い範囲で異なる可能性があるためです。まずデータベースのサイズを見積もってから、テスト・データベースを作成し、それに標準的なデータを入れてみてください。

コントロール・センターからは、さまざまなデータベース・オブジェクトのサイズ所要量の決定に役立つ次のようなユーティリティーにアクセスできます。

- オブジェクトを 1 つ選択して、 Estimate Size ユーティリティーを使用することができます。このユーティリティーは、表などの既存オブジェクトの現行サイズを表示します。その後、オブジェクトを変更すると、オブジェクトの新しい見積もり値が算出されます。このユーティリティーは、今後の増加を考慮に入れてストレージ所要量を概算するのに役立ちます。このユーティリティーは、オブジェクト・サイズの見積もり値を 1 つ算出するだけではありません。オブジェクトのサイズの可能な範囲、つまり現行値に基づいた最小値と可能最大値を算出します。
- 「関連項目表示 (Show Related)」ウィンドウを使用すると、関連するさまざまなオブジェクトを判別できます。
- インスタンスの任意のデータベース・オブジェクトを選択して、「DDL の生成 (Generate DDL)」を要求することができます。この機能では **db2look** ユーティリティーを使用して、データベースのデータ定義ステートメントを生成します。

このいずれの場合も、「SQL の表示 (Show SQL)」ボタンまたは「コマンドの表示 (Show Command)」ボタンのどちらかが使用できます。また、結果として生成される SQL ステートメントまたはコマンドをスクリプト・ファイルに保管し、後で使用することもできます。これらのすべてのユーティリティーにはオンライン・ヘルプがあります。

物理データベース要件の計画の際には、これらのユーティリティーを念頭に置いてください。

データベースのサイズ見積もりを行う時、以下の要素を考慮する必要があります。

- システム・カタログ表
- ユーザー表データ
- ロング・フィールドのデータ
- ラージ・オブジェクト (LOB) データ
- 索引スペース
- ログ・ファイル・スペース
- 一時ワークスペース

以下に関するスペース所要量については、説明を省略します。

- ローカル・データベース・ディレクトリーのファイル
- システム・データベース・ディレクトリーのファイル
- オペレーティング・システムに必要なファイル管理オーバーヘッド。これには次のものが含まれます。
  - ファイル・ブロック・サイズ
  - ディレクトリー制御スペース

関連概念:

- 83 ページの『システム・カタログ表のスペース所要量』
- 83 ページの『ユーザー表データのスペース所要量』
- 85 ページの『ロング・フィールドのデータのスペース所要量』

- 86 ページの『ラージ・オブジェクト・データのスペース所要量』
- 87 ページの『索引のスペース所要量』
- 89 ページの『ログ・ファイルのスペース所要量』
- 91 ページの『一時表のスペース所要量』

関連資料:

- 「*コマンド・リファレンス*」の『db2look - DB2 統計および DDL 抽出ツール・コマンド』

システム・カタログ表のスペース所要量

データベースが作成されると、システム・カタログ表も作成されます。システム表は、データベース・オブジェクトと特権がデータベースに追加されるたびに増加します。最初の時点では、約 3.5 MB のディスク・スペースが使用されます。

カタログ表に割り当てられるスペースの量は、表スペースのタイプとカタログ表が含まれる表スペースのエクステント・サイズによって異なります。たとえば、エクステント・サイズが 32 の DMS 表スペースを使用した場合、最初の時点ではカタログ表に 20 MB のスペースが割り振られます。

注: 複数パーティションを持つデータベースの場合、カタログ表は CREATE DATABASE の発行元のパーティション上にのみ存在します。カタログ表のディスク・スペースは、そのパーティションのためだけに必要です。

関連概念:

- 81 ページの『データベース・オブジェクトのスペース所要量』
- 「*管理ガイド: インプリメンテーション*」の『システム・カタログ表の定義』

ユーザー表データのスペース所要量

デフォルトでは、表データは 4 KB のページに格納されます。各ページ (ページ・サイズは関係ない) には、68 バイトからなる データベース・マネージャー 用のオーバーヘッドが含まれます。そのため、ユーザー・データ (つまり行) を入れるのに 4028 バイトが残されています。ただし 4 KB のページ上のいずれの行も、長さ 4005 バイトを超えてはなりません。 1 つの行が複数のページにまたがることはありません。ページ・サイズが 4 KB の場合、使用できる列の数は最大で 500 です。

表データ・ページには、 LONG VARCHAR、 LONG VARGRAPHIC、 BLOB、CLOB、および DBCLOB データ・タイプとして定義された列のデータは含まれません。ただし、それらの列の記述子は、表データ・ページの行に含まれます。

通常、行は先頭一致順で正規の表に挿入されます。ファイルの中から、新しい行を保持できる最初の使用可能なスペースが (フリー・スペース・マップを使って) 検索されます。行が更新されると、それを含むページ上に十分なスペースがある限り、行は元の場所のままで更新されます。この場合には、オリジナルの行ロケーションにレコードが 1 つ作成され、更新された行の表ファイルの中の新しいロケーションを指し示します。

ALTER TABLE APPEND ON ステートメントが呼び出されると、データは常に追加され、データ・ページのフリー・スペースについての情報は保持されません。

表にクラスター索引が定義されているなら、 DB2® Universal Database (DB2 UDB) は、そのクラスター索引のキー順に従ってデータを物理的にクラスター化することを試みます。その表に行が挿入されると、DB2 UDB は、まずクラスター索引の中からそのキー値を検索します。キー値が見つかったなら、 DB2 UDB はそのキーの指し示すデータ・ページにレコードを挿入することを試みます。キー値が見つからないなら、それより高い次のキー値が使用されて、そのキー値のレコードを含むページにレコードが挿入されます。表の「ターゲット」ページに十分なスペースがない場合には、その近くにあるページからスペースを検索するため、フリー・スペース・マップが使用されます。時間が経過するにつれて、データ・ページのスペースが完全に使用されるようになると、レコードは表の中の「ターゲット」ページからますます遠い位置に入れられるようになります。やがて表データはクラスター化されていないと見なされるようになったなら、表の再編成を使用することによりクラスター化されたデータの並びが復元できます。

表がマルチディメンション・クラスタリング (MDC) 表の場合、DB2 UDB は、レコードが 1 つ以上の定義済みディメンションまたはクラスター索引に従って、常に物理的にクラスター化されることを保証します。 MDC 表が特定のディメンションで定義されているなら、各ディメンションごとにブロック索引が作成され、セル (ディメンション値のユニークな組み合わせ) をブロックにマップする複合ブロック索引が作成されます。この複合ブロック索引は、特定のレコードがどのセルに属するか、また表の中でそのセルに属するレコードを含むブロックまたはエクステントは正確に言ってどれであるかを決定するために使用されます。その結果、レコード挿入時に DB2 UDB は、ディメンション値が同じであるレコードを含むブロックのリストを複合ブロック索引から検索し、スペース検索の対象をそれらのブロックだけに限定します。セルがまだ存在していない場合、またはそのセルの既存のブロック中に十分なスペースがない場合には、別のブロックがそのセルに割り当てられ、そこにレコードが挿入されます。その場合も、ブロック内で使用可能なスペースをすばやく見つけるために、フリー・スペース・マップが使用されます。

4 KB ページ数を見積もるには、データベース内のそれぞれのユーザー表ごとに、以下のように計算します。

```
ROUND DOWN(4028/(average row size + 10)) = records_per_page
```

上の結果を以下の式に代入します。

```
(number_of_records/records_per_page) * 1.1 = number_of_pages
```

ここで平均行サイズは平均列サイズの合計です。また 1.1 はオーバーヘッドに起因する値です。

注: この公式はあくまでも見積もりの算出です。フラグメント化やオーバーフロー・レコードのためにレコード長にばらつきがある場合、この見積もりの精度は低下します。

ページ・サイズが 8 KB、16 KB、または 32 KB のバッファー・プールまたは表スペースを作成するためのオプションもあります。特定サイズの表スペース内に作成される表にはすべて、一致するページ・サイズが指定されます。単一の表または索引オブジェクトのサイズは、 (32 KB のページ・サイズを想定すると) 最大 512

GB まで可能です。ページ・サイズが 8 KB、16 KB、または 32 KB の場合、使用できる列は最大で 1012 です。 4 KB ページ・サイズの場合、列の最大数は 500 です。行の最大長は、ページ・サイズに応じて以下のように異なります。

- ページ・サイズが 4 KB の場合、行の最大長は 4005 バイトです。
- ページ・サイズが 8 KB の場合、行の最大長は 8101 バイトです。
- ページ・サイズが 16 KB の場合、行の最大長は 16 293 バイトです。
- ページ・サイズが 32 KB の場合、行の最大長は 32 677 バイトです。

ページ・サイズを大きくすると、索引のレベルの数を減少させることができます。行のランダム読み取りおよび書き込みを行う OLTP (オンライン・トランザクション処理) アプリケーションを使用する場合は、不必要な行に使用するバッファー・スペースが少なくなるので、ページ・サイズは小さい方が望ましいです。一度に多くの連続した行にアクセスする DSS (意思決定支援システム) アプリケーションを使用する場合は、指定された数の行を読み取るのに必要な入出力要求の数が減るので、ページ・サイズは大きい方が望ましいです。ただし、ページ・サイズを 255 で割った数字よりも行サイズが小さいときは例外です。このような場合は、各ページに無駄なスペースが生じます。 (1 ページあたりの行数は、最大で 255 です。) 無駄なスペースを少なくするために、ページ・サイズは小さい方が望ましいでしょう。

バックアップを異なるページ・サイズにリストアすることはできません。

756 列以上を表す IXF データ・ファイルをインポートすることはできません。

宣言済み一時表は、独自の「ユーザー一時」表スペース・タイプの中にのみ作成されます。デフォルトのユーザー一時表スペースはありません。一時表には LONG データを入れることはできません。これらの表は、アプリケーションがデータベースから切断すると暗黙的にドロップされます。これらのスペース所要量を見積もる際には、この点を考慮に入れる必要があります。

関連概念:

- 81 ページの『データベース・オブジェクトのスペース所要量』

ロング・フィールドのデータのスペース所要量

ロング・フィールド・データは、他のデータ・タイプのストレージ・スペースとは異なる、別個の表オブジェクトに格納します。

データが格納される 32 KB 域は、「2 の累乗」× 512 バイトのサイズのセグメントに分割されます。 (このため、これらのセグメントは、512 バイト、1024 バイト、2048 バイトというように、最高 32 768 バイトまでが可能です。)

ロング・フィールド・データ・タイプ (LONG VARCHAR または LONG VARGRAPHIC) は、フリー・スペースを容易にレクラメーション処理できるような方法で保管されます。割り振りとフリー・スペースに関する情報は 4 KB の割り振りページに格納されますが、オブジェクト中はさほど頻繁には見られません。

オブジェクト内の未使用スペースの量は、ロング・フィールド・データのサイズと、そのサイズがデータのすべてのオカレンスを通じてある程度一定しているかど

うかによって決まります。 255 バイトを超えるデータ入力項目の場合、未使用のスペースは、ロング・フィールド・データのサイズの 50% に達することもあります。

文字データの長さがページ・サイズより短く、データの残りの部分と一緒にレコードの中に収まる場合には、 CHAR、GRAPHIC、VARCHAR、または VARGRAPHIC の各データ・タイプを、 LONG VARCHAR または LONG VARGRAPHIC の代わりに使用する必要があります。

関連概念:

- 81 ページの『データベース・オブジェクトのスペース所要量』

ラージ・オブジェクト・データのスペース所要量

ラージ・オブジェクト (LOB) データは、他のデータ・タイプのストレージ・スペースとは構造が異なる、 2 つの別個の表オブジェクトに格納します。

LOB データに必要なスペースを見積もるには、これらのデータ・タイプで定義されたデータを格納するための、次のような 2 つの表オブジェクトについて考慮する必要があります。

- **LOB データ・オブジェクト**

  データが格納される 64 MB 域は、「2 の累乗」× 1024 バイトのサイズのセグメントに分割されます。 (つまり、このセグメントの大きさは、1024 バイト、2048 バイト、 4096 バイトというようにして、最高 64 MB までとなります。)

  LOB データに使用するディスク・スペースの量を少なくするために、 CREATE TABLE および ALTER TABLE ステートメントの *lob-options* 文節で COMPACT オプションを指定することができます。 COMPACT オプションを指定すると、LOB データは小さなセグメントに分割され、必要なディスク・スペースの量を最小限にとどめることができます。これはデータ圧縮を伴わず、単に最も近い 1 KB 境界になるよう、最小限のスペースを使用しているだけです。 LOB 値に付加する場合、 COMPACT オプションを指定するとパフォーマンスが低下する可能性があります。

  LOB データ・オブジェクトでのフリー・スペース量は、更新および削除の量と挿入する LOB 値のサイズによって変わります。

- **LOB 割り振りオブジェクト**

  割り振りとフリー・スペースに関する情報は、実際のデータとは別の 4 KB 割り振りページに格納されます。使用される 4 KB ページの数は、ラージ・オブジェクト・データに割り振られるデータの量 (未使用スペース含む) に依存しています。オーバーヘッドは次のように計算されます。 64 GB ごとに 4 KB ページ 1 個 + 8 MB ごとに 4 KB ページ 1 個。

文字データの長さがページ・サイズより短く、データの残りの部分と一緒にレコードの中に収まる場合には、 CHAR、GRAPHIC、VARCHAR、または VARGRAPHIC の各データ・タイプを、 BLOB、CLOB、または DBCLOB の代わりに使用する必要があります。

関連概念:

- 81 ページの『データベース・オブジェクトのスペース所要量』

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『ラージ・オブジェクト (LOB)』

索引のスペース所要量

各索引に必要なスペースは、次のようにして見積もることができます。

```
(average index key size + 9) * number of rows * 2
```

ここで、

- 「average index key size (平均索引キー・サイズ)」は、索引キーの中の各列のバイト・カウントです。 (VARCHAR および VARGRAPHIC 列の平均の列サイズを見積もるには、現行データ・サイズの平均に 2 バイトを加えます。宣言されている最大サイズは使用しません。)
- 「2」倍しているのは、ノンリーフ・ページやフリー・スペースなどのオーバーヘッドのためです。

注:

1. NULL 値が可能な列の場合は、NULL 値標識のために 1 バイトをさらに追加してください。
2. マルチディメンション・クラスタリング (MDC) 表に対して内部的に作成されたブロック索引の場合、 “number of rows” (行数) を “number of blocks” で置き換えることになります。

索引の作成時には一時スペースが必要になります。索引作成時に必要な一時スペースの最大量は、次のようにして見積もることができます。

```
(average index key size + 9) * number of rows * 3.2
```

ここで 3.2 は索引オーバーヘッドおよび索引の作成時のソートに必要なスペースに起因する値です。

注: 非ユニーク索引の場合、重複キー項目を保管するために、5 バイトだけが必要です。上記に示した見積もりは、重複がないものと想定しています。 1 つの索引を保管するために必要なスペースは、上記の公式では余分に見積もってしまう場合があります。

リーフ・ページの数を見積もるために、以下の 2 つの公式を使用できます (2 番目の方がより正確な見積もりが可能です)。これらの見積もりの精度は、平均値がどの程度うまく実際のデータを反映しているかに大きく依存しています。

注: SMS 表スペースの場合、必要な最小スペースは 12 KB です。 DMS 表スペースの場合、最小は 1 エクステントです。

- リーフ・ページ当たりのキーの数の平均値は、おおよそ以下のように見積もることができます。

```
(.9 * (U - (M*2))) * (D + 1)
----------------------------
      K + 7 + (5 * D)
```

ここで、

- U (ページでの使用可能なスペース) は、ページ・サイズから 100 を引いた数値とほぼ等しい。ページ・サイズが 4096 の場合、U は 3996。
- M = U / (9 + *minimumKeySize*)
- D = キー値当たりの重複の平均数
- K = *averageKeySize*

minimumKeySize と *averageKeysize* は、それぞれ NULL 可能なキー部、および各可変長キー部のために余分に 2 バイトが必要であることに注意してください。

組み込み列がある場合は、それらを *minimumKeySize* と *averageKeySize* 用に計算に入れる必要があります。

索引作成中に、デフォルトの 10 % 以外の空きパーセントが指定されている場合、 .9 は、(100 - pctfree)/100 の値で置き換えることができます。

- リーフ・ページ当たりのキーの数の平均値をより正確に見積もるには、以下のようにします。

```
L = number of leaf pages = X / (avg number of keys on leaf page)
```

ここで X は表の中の行の総数です。

次のようにして索引の元のサイズを見積もることができます。

```
(L + 2L/(average number of keys on leaf page)) * pagesize
```

DMS 表スペースの場合、1 つの表上のすべての索引について合計サイズを算出し、索引が存在する表スペースのエクステント・サイズの倍数に切り上げます。

挿入 (INSERT) /更新 (UPDATE) 活動による索引の増加のための追加スペースを設ける必要がありますが、これはページ分割という結果になることがあります。

元の索引サイズのより正確な見積もりと、索引の中のレベルの数の見積もりを出すには、以下の計算式を使用します。 (これは、索引定義で組み込み列が使用されている場合は特に注意を引くものとなります。) ノンリーフ・ページ当たりのキーの数の平均値は、おおよそ以下のようになります。

```
(.9 * (U - (M*2))) * (D + 1)
----------------------------
      K + 13 + (9 * D)
```

ここで、

- U (ページでの使用可能なスペース) は、ページ・サイズから 100 を引いた数値とほぼ等しい。ページ・サイズが 4096 の場合、U は 3996。
- D はノンリーフ・ページ上のキー値あたりの平均重複数 (この数はリーフ・ページ上での数よりもずっと小さいので、この値を 0 に設定して計算を単純化することもできます)。
- M = U / (9 + ノンリーフ・ページの *minimumKeySize*)
- K = ノンリーフ・ページの *averageKeySize*

ノンリーフ・ページの *minimumKeySize* および *averageKeySize* は、組み込み列が存在する場合を除いて、リーフ・ページのものと同じです。組み込み列は、ノンリーフ・ページ上には保管されません。

`(100 - pctfree)/100` の値が `.9` より大きくなければ、この値を `.9` で置き換えることはできません。なぜなら、索引作成中に最大 10 % のフリー・スペースがノンリーフ・ページに残されるからです。

ノンリーフ・ページの数は、次のようにして見積もることができます。

```
if L > 1 then {P++; Z++}
While (Y > 1)
{
   P = P + Y
   Y = Y / N
  Z++
}
```

ここで、

– P はページの数 (初期値は 0)。
– L はリーフ・ページの数。
– N は各ノンリーフ・ページのキーの数。
– Y = L / N
– Z は索引ツリーのレベルの数 (初期値は 1)。

ページの合計数は、

```
T = (L + P + 2) * 1.0002
```

0.02 % を追加するのは、スペース・マップ・ページを含むオーバーヘッドのためです。

索引を作成するために必要なスペースの量は次のように見積もります。

```
T * pagesize
```

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『索引』
- 81 ページの『データベース・オブジェクトのスペース所要量』
- 「*管理ガイド: パフォーマンス*」の『索引の終結処理および保守』

ログ・ファイルのスペース所要量

ログ制御ファイルの場合、32 KB のスペースが必要です。

また、最低でも、アクティブ・ログ構成のために十分なスペースが必要です。以下のように計算することができます。

```
(logprimary + logsecond) * (logfilsiz + 2 ) * 4096
```

ここで、

- *logprimary* は、データベースの構成ファイルに定義されている 1 次ログ・ファイルの数です。

- *logsecond* は、データベース構成ファイル内に定義されている 2 次ログ・ファイルの数です。この計算式で、*logsecond* を *-1* に設定することはできません。(*logsecond* を -1 に設定すると、無限のアクティブ・ログ・スペースを要求していることになります。)
- *logfilsiz* は、データベースの構成ファイルに定義されている各ログ・ファイルのページ数です。
- 2 は各ログ・ファイルに必要なヘッダー・ページの数です。
- 4096 は 1 ページのバイト数です。

データベースが循環ロギングのために使用される場合、この公式の結果は、十分なディスク・スペース量を提供します。

データベースがロールフォワード・リカバリーのために使用される場合、次のような特別のログ・スペース所要量について考慮する必要があります。

- *logretain* 構成パラメーターを使用可能にすると、ログ・ファイルがログ・パス・ディレクトリーにアーカイブされます。オンライン・ディスク・スペースは、ログ・ファイルを別のロケーションに移動しない限り、いつかいっぱいになります。
- *userexit* 構成パラメーターを使用可能にすると、 userexit プログラムによって、アーカイブ・ログ・ファイルが別のロケーションに移動されます。次のもののために、さらに余分のログ・スペースが必要です。
  - userexit プログラムによって移動される前のオンライン・アーカイブ・ログ。
  - 将来の利用のために初期化されている新しいログ・ファイル。

データベースが無限ロギングのために使用される場合 (つまり、 *logsecond* を *-1* に設定する場合)、 *userexit* 構成パラメーターが使用されなければなりません。したがって、同じディスク・スペースに関する考慮事項があります。 DB2® Universal Database (DB2 UDB) は、少なくとも、ログ・パス内の *logprimary* によって指定されたアクティブ・ログ・ファイルの数を保持します。したがって、上記の公式内の *logsecond* に -1 の値を使用しないでください。ログ・ファイルのアーカイブによって生じる遅延を許可するために、余分のディスク・スペースを提供するようにしてください。

ログ・パスをミラーリングする場合、見積もりログ・ファイルのスペース所要量を 2 倍にする必要があります。

関連概念:

- 81 ページの『データベース・オブジェクトのスペース所要量』
- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『リカバリー・ログについて』
- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『ログのミラーリング』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『logfilsiz - 「ログ・ファイルのサイズ」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『logprimary - 「1 次ログ・ファイル数」構成パラメーター』

- 「*管理ガイド: パフォーマンス*」の『logsecond - 「2 次ログ・ファイル数」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『mirrorlogpath - 「ミラー・ログ・パス」構成パラメーター』

一時表のスペース所要量

一部の SQL ステートメントには、処理のための一時表が必要です (メモリー中で行えないソートのための作業ファイルなど)。それらについては、それらを使用する時に記憶のためのディスク・スペースが必要になります。これらの一時表にはディスク・スペースが必要です。必要なスペースの量は、照会のサイズ、数、および性質、また戻される表のサイズに応じて異なります。実際の作業環境はそれぞれに異なっているため、一時表のスペース要件を見積もることは困難です。たとえば、さまざまなシステム一時表の存在期間が長いために、システム一時表スペースに割り振られるスペースの量が実際より多いように思える場合があります。このことは、DB2_SMS_TRUNC_TMPTABLE_THRESH を使用した場合に発生することがあります。

データベース・システム・モニターおよび表スペース照会 API を使って、通常の操作で使用されるワークスペース量を追跡することができます。

関連概念:

- 81 ページの『データベース・オブジェクトのスペース所要量』

関連資料:

- 「*管理 API リファレンス*」の『sqlbmtsq - 表スペース照会』

データベース・パーティション・グループ

データベース・パーティション・グループは、 1 つ以上のデータベース・パーティションのセットです。データベースに対して表を作成したい場合、まず、表スペースが保管される予定のデータベース・パーティション・グループを作成した後、表が保管される予定の表スペースを作成します。

1 つのデータベースの中に、 1 つ以上のデータベース・パーティションのサブセットを名前付きで定義することができます。定義する各サブセットをデータベース・パーティション・グループ と呼びます。 1 つ以上のデータベース・パーティションが含まれる各サブセットを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 と呼びます。複数パーティションのデータベース・パーティション・グループは、同じインスタンスに属するデータベース・パーティションでのみ定義することができます。

92 ページの図 25 は、5 つのパーティションを持つデータベースの例を示したものであり、その中は以下のようになっています。

- あるデータベース・パーティション・グループは、 1 つのデータベース・パーティションを除き、すべてにまたがっています (データベース・パーティション・グループ 1)。

- あるデータベース・パーティション・グループには、 1 つのデータベース・パーティションが含まれます (データベース・パーティション・グループ 2)。
- あるデータベース・パーティション・グループには、 2 つのデータベース・パーティションが含まれます。(データベース・パーティション・グループ 3)。
- データベース・パーティション・グループ 2 に含まれているデータベース・パーティションは、データベース・パーティション・グループ 1 によって共有 (およびオーバーラップ) されます。
- データベース・パーティション・グループ 3 には、 1 つのデータベース・パーティションがありますが、これはデータベース・パーティション・グループ 1 によって共有 (およびオーバーラップ) されています。

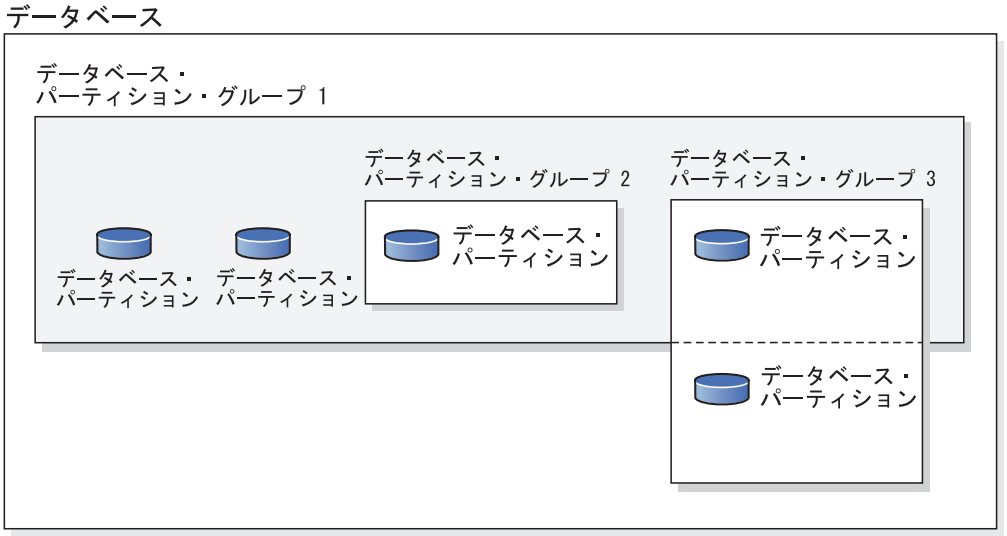

図 25. データベース内のデータベース・パーティション・グループ

CREATE DATABASE PARTITION GROUP ステートメントを使用して、新しいデータベース・パーティション・グループを作成します。これは、 ALTER DATABASE PARTITION GROUP ステートメントを使用して変更できます。データはデータベース・パーティション内のすべてのパーティションに分割されており、データベース・パーティション・グループ内では、 1 つまたは複数のデータベース・パーティションを追加またはドロップできます。複数パーティションのデータベース・パーティション・グループを使用している場合、データベース・パーティション・グループ設計に関するいくつかの考慮事項を検討する必要があります。

データベース・システム構成の一部である各データベース・パーティションは、*db2nodes.cfg* と呼ばれる**パーティション構成ファイル**の中にあらかじめ定義されていなければなりません。 1 つのデータベース・パーティションには、最小で 1 つのデータベース・パーティションを、最大でそのデータベース・システムに定義されたすべてのデータベース・パーティションを含めることができます。

データベース・パーティション・グループが作成または修正されるときには、パーティション・マップ がそれに関連付けられます。パーティション・マップは、パーティション・キー およびハッシュ・アルゴリズムとともにデータベース・マネージャーによって使用され、データベース・パーティション・グループ内のどのデータベース・パーティションが特定のデータの行を保管するかが決められます。

非パーティション・データベースでは、パーティション・キーおよびパーティション・マップは必要ありません。データベース・パーティションとはデータベースの一部分であり、ユーザー・データ、索引、構成ファイル、およびトランザクション・ログで構成されます。データベースが作成されたときに作成されたデフォルトのデータベース・パーティション・グループは、データベース・マネージャーによって使用されます。 IBMCATGROUP は、システム・カタログが入っている表スペースのデフォルトのデータベース・パーティション・グループです。 IBMTEMPGROUP は、システム一時表スペース用のデフォルトのデータベース・パーティション・グループです。 IBMDEFAULTGROUP は、そこに書き込むことのできる、ユーザー定義の表が入る表スペース用のデフォルトのデータベース・パーティション・グループです。宣言済み一時表のためのユーザー一時表スペースは、IBMDEFAULTGROUP または任意のユーザー作成のデータベース・パーティション・グループの中に作成できますが、 IBMTEMPGROUP の中には作成できません。

関連概念:

- 93 ページの『データベース・パーティション・グループの設計』
- 94 ページの『パーティション・マップ』
- 96 ページの『パーティション・キー』

関連資料:

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

データベース・パーティション・グループの設計

パーティション・データベースを使用していない場合には、データベース・パーティション・グループについての設計上の考慮事項はありません。

DB2® Design Advisor は、データベース・パーティション・グループを推奨するために使用できるツールです。 DB2 Design Advisor は、コントロール・センターから、あるいはコマンド行プロセッサーから db2advis を使用して利用できます。

複数パーティションのデータベース・パーティション・グループを使用している場合は、以下の設計のポイントを考慮してください。

- 複数パーティションのデータベース・パーティション・グループでは、パーティション・キーのスーパーセットである場合にのみ、ユニーク索引を作成することができます。
- データベース内のデータベース・パーティションの数によっては、 1 つ以上の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、および 1 つ以上の複数パーティションのデータベース・パーティション・グループを作成できます。
- 各データベース・パーティションにはユニークなパーティション番号を割り当てる必要があります。同じデータベース・パーティションが 1 つ以上のデータベース・パーティション・グループに存在することもできます。

- システム・カタログ表を含んでいるデータベース・パーティションを速やかにリカバリーさせるためには、同じデータベース・パーティションにユーザー表を入れないようにしてください。こうするには、 IBMCATGROUP データベース・パーティション・グループのデータベース・パーティションを含まないデータベース・パーティション・グループの中に、ユーザー表を入れます。

小さな表は、大きな表とのコロケーション を利用したい場合を除き、単一パーティションのデータベース・パーティション・グループの中に置いてください。コロケーションとは、同じデータベース・パーティションにある、関連データが入った複数の異なる表から行を配置することです。配置された表を使用して、DB2 Universal Database™ (DB2 UDB) は結合ストラテジーをより効率的に利用することができます。配置された表は、 1 つの単一パーティションのデータベース・パーティション・グループの中に存在することができます。複数の表が 1 つの複数パーティションのデータベース・パーティション・グループの中に存在し、パーティション・キーの中に同数の列を持ち、対応する列のデータ・タイプがパーティション互換である場合、それらの表は配置されていると見なされます。同じパーティション・キー値を持つ配置された表の中の行は、同じデータベース・パーティションに置かれます。それぞれの表が同じデータベース・パーティション・グループの中の別個の表スペースの中に入っていても、配置されていると見なされます。

中間サイズの表を、あまりに多くのデータベース・パーティションにわたって拡張することは避けるべきです。たとえば、100 MB の表の場合、32 パーティションのデータベース・パーティション・グループ上よりも、 16 パーティションのデータベース・パーティション・グループ上のほうがパフォーマンスが良くなります。

データベース・パーティション・グループを使用して、オンライン・トランザクション処理 (OLTP) の表を意思決定支援 (DSS) の表と分離して、 OLTP トランザクションのパフォーマンスが影響を受けて低下しないようにすることができます。

関連概念:

- 91 ページの『データベース・パーティション・グループ』
- 94 ページの『パーティション・マップ』
- 96 ページの『パーティション・キー』
- 98 ページの『表のコロケーション』
- 99 ページの『パーティションの互換性』
- 100 ページの『複製されたマテリアライズ照会表』

関連資料:

- 「コマンド・リファレンス」の『設計アドバイザー・コマンド』

パーティション・マップ

パーティション・データベース環境では、どのデータベース・パーティションに表のどの行が保管されているかをデータベース・マネージャーが何らかの方法で認識する必要があります。データベース・マネージャーは必要なデータの場所を認識する必要があり、データを見付けるためにパーティション・マップ というマップを使用します。

パーティション・マップは内部で生成された配列であり、複数パーティションのデータベース・パーティション・グループの場合は 4 096 項目が、単一パーティションのデータベース・パーティション・グループの場合は単一の項目が入っています。単一パーティションのデータベース・パーティション・グループの場合、パーティション・マップの項目は 1 つのみで、そこには、データベース表のすべての行が保管されているデータベース・パーティションのパーティション番号が入っています。複数パーティションのデータベース・パーティション・グループの場合、データベース・パーティション・グループのパーティション番号は、ラウンドロビン方式で指定されます。都市の地図が格子状のセクションで構成されているように、データベース・マネージャーは、パーティション・キー を使用して、データが保管されているロケーション (データベース・パーティション) を判別します。

たとえば、 4 つのデータベース・パーティション (0 から 3 までの番号が付けられている) 上に作成されたデータベースがあるとします。このデータベースの IBMDEFAULTGROUP データベース・パーティション・グループのパーティション・マップは、次のようになります。

```
0 1 2 3 0 1 2 ...
```

データベース・パーティションの 1 および 2 を使用してデータベース・パーティション・グループがデータベース内に作成されている場合、そのデータベース・パーティション・グループのパーティション・マップは、以下のようになります。

```
1 2 1 2 1 2 1 ...
```

データベースにロードされる表のパーティション・キーが 1 と 500 000 の間の可能値を持つ整数である場合、パーティション・キーは、0 と 4 095 の間のパーティション番号になるようハッシュが行われます。この番号は、その行のデータベース・パーティションを選択するためのパーティション・マップの索引として使用されます。

96 ページの図 26 は、パーティション・キー値 (c1, c2, c3) を持つ行が、パーティション 2 にマップされ、次にパーティション 2 がデータベース・パーティション n5 を参照する方法を示しています。

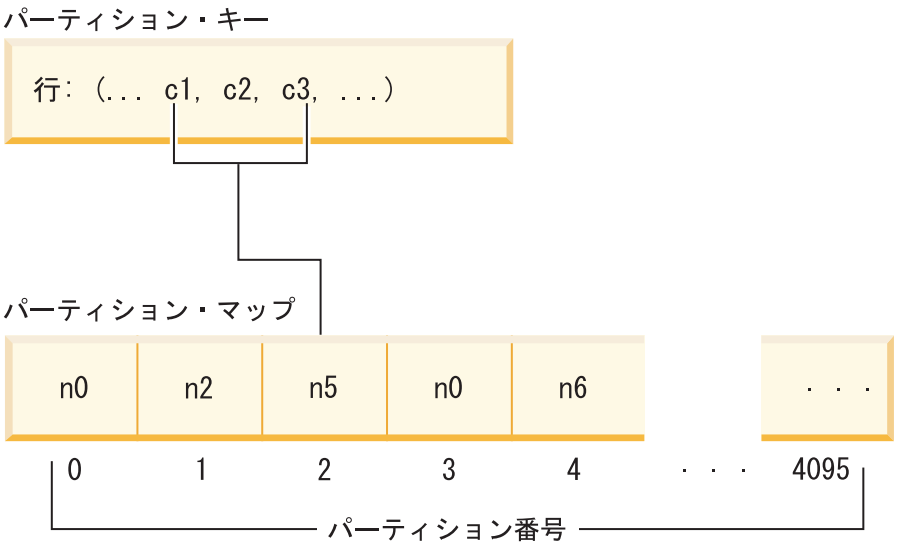

図 26. パーティション・マップを使用したデータ分散

パーティション・マップは、パーティション・データベースのどこにデータが保管されるかを制御するための柔軟性のある手段です。データベース内のデータベース・パーティションにわたるデータ分散を将来変更する必要がある場合、データ再分散ユーティリティーを使用することができます。このユーティリティーによって、データ分散のバランスをとり直したり、データ分散にスキューを導入したりすることができます。

表パーティション情報入手 (**sqlugtpi**) API を使用して、見ることができるパーティション・マップのコピーを入手することができます。

関連概念:

- 91 ページの『データベース・パーティション・グループ』
- 93 ページの『データベース・パーティション・グループの設計』
- 96 ページの『パーティション・キー』

関連資料:

- 「*管理 API リファレンス*」の『sqlugtpi - 表のパーティション情報の入手』

パーティション・キー

パーティション・キー とは、特定のデータの行が保管されるパーティションを判別するために使用される 1 つの列 (または列のグループ) のことです。パーティション・キーは、CREATE TABLE ステートメントを使用して表の上に定義されます。データベース・パーティション・グループ内の複数のデータベース・パーティションにわたって分割された表スペース内の表に対してパーティション・キーが定義されていない場合、デフォルトによって、パーティション・キーが主キーの最初の列から作成されます。主キーが指定されていない場合は、デフォルトのパーティション・キーは、その表に定義された最初のロング・フィールド列以外の列になります。 (*長形式* には、すべてのロング・データ・タイプとすべてのラージ・オブジェクト (LOB) データ・タイプが含まれます。) 単一パーティション・データベース・パーティション・グループに関連した表スペースの中に表を作成している場合、パ

ーティション・キーが必要であれば、パーティション・キーを明示して定義しなければなりません。デフォルトでは、パーティション・キーは作成されません。

デフォルトのパーティション・キーの要件を満たす列がない場合、表はパーティション・キーなしで作成されます。パーティション・キーのない表は、単一パーティション・データベース・パーティション・グループでのみ使用できます。後で、ALTER TABLE ステートメントを使用してパーティション・キーを追加またはドロップできます。パーティション・キーの変更は、単一パーティション・データベース・パーティション・グループに関連した表スペースにある表に対してのみ行うことができます。

適切なパーティション・キーを選択することが重要です。以下の点を考慮してください。

- 表がアクセスされる方法
- 照会のワークロードの性質
- データベース・システムによって採用されている結合ストラテジー

コロケーションが主な考慮事項ではない場合、表に対する適切なパーティション・キーは、データベース・パーティション・グループ内のすべてのデータベース・パーティションに均等にデータが分散するようなパーティション・キーです。データベース・パーティション・グループに関連する表スペースの中のそれぞれの表に対するパーティション・キーによって、その表が配置されているかどうかが判別されます。表は、以下の場合に配置されていると考えられます。

- 表が同データベース・パーティション・グループ内にある表スペースに置かれている。
- それぞれの表のパーティション・キーが同じ数の列を持っている。
- 対応する列のデータ・タイプがパーティション互換である。

これらの特性により、同じパーティション・キー値を持つ配置された表の各行は、確実に同じパーティションに配置されるようになります。

パーティション・キーが不適切であると、データの分散が不均一になる可能性があります。不均一に分配されたデータを持つ列、および異なる値の数が少ない列は、パーティション・キーとして選択するべきではありません。異なる値の数は、データベース・パーティション・グループ内のすべてのデータベース・パーティションにわたって行を均等に分散するのに十分な大きさでなければなりません。パーティション化ハッシュ・アルゴリズムを適用するためのコストは、パーティション・キーのサイズに比例します。パーティション・キーは 16 列より多くできず、列が少ないほどパフォーマンスは良くなります。不必要な列は、パーティション・キーの中に含めるべきではありません。

パーティション・キーを定義する場合には、以下の点を考慮する必要があります。

- ロング・データ・タイプ (LONG VARCHAR、 LONG VARGRAPHIC、BLOB、CLOB、または DBCLOB) のみを含む複数パーティションの表を作成することはできません。
- パーティション・キーの定義は変更できません。
- パーティション・キーには、最も頻繁に結合される列を含める必要があります。

- パーティション・キーは、GROUP BY 文節に頻繁に関与している列で構成する必要があります。
- どのユニーク・キーまたは主キーにも、すべてのパーティション・キー列が含まれていなければなりません。
- オンライン・トランザクション処理 (OLTP) 環境では、パーティション・キーの中のすべての列が、定数またはホスト変数を持つ等号 (=) 述部を使用することによって、トランザクションに関与する必要があります。たとえば、以下のようなトランザクションでよく使用される従業員番号 *emp_no* があるとします。

  ```
  UPDATE emp_table SET ... WHERE
  emp_no = host-variable
  ```

  この場合、 EMP_NO 列を EMP_TABLE の適切な単一列パーティション・キーとして使用できるでしょう。

ハッシュ・パーティション化 とは、パーティション表内の各行の配置を決定する方式です。この方式は、以下のようなしくみです。

1. ハッシュ・アルゴリズムが、パーティション・キーの値に適用され、ゼロと 4095 の間のパーティション番号を生成します。
2. データベース・パーティション・グループが作成されるときに、パーティション・マップが作成されます。パーティション番号のそれぞれは、パーティション・マップを充てんするために、ラウンドロビン方式で順番に繰り返されます。
3. パーティション番号は、パーティション・マップへの索引として使用されます。パーティション・マップの中のそのロケーションにある番号は、その行が保管されているデータベース・パーティションの番号になります。

関連概念:

- 91 ページの『データベース・パーティション・グループ』
- 93 ページの『データベース・パーティション・グループの設計』
- 94 ページの『パーティション・マップ』
- 「*管理ガイド: パフォーマンス*」の『設計アドバイザー』

関連資料:

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

表のコロケーション

ある種の照会の応答で、特定の複数の表のデータが頻繁に使われる場合があります。このような場合、これらの表からの関連データをできるだけ近接して配置する必要があります。データベースが物理的に 2 つ以上のデータベース・パーティションに分割されている環境では、分割された表の関連する部分を、何らかの方法でできるだけ近接するように維持する必要があります。これを行うための機能を*表のコロケーション* と呼びます。

表は、同じデータベース・パーティション・グループ内に保管される場合、およびそれらのパーティション・キーが互換性がある場合に配置されます。両方の表を同じデータベース・パーティション・グループに置くことによって、共通のパーティション・マップにすることができます。これらの表は異なる表スペースに入れることは可能ですが、その表スペースは同じデータベース・パーティション・グループ

に関連付けられていなければなりません。各パーティション・キーの中の対応する列のデータ・タイプは、*パーティション互換* でなければなりません。

DB2® Universal Database (DB2 UDB) には、結合または副照会で複数の表にアクセスするときに、結合するべきデータが同じデータベース・パーティションに配置されていることを認識する機能があります。同じデータベース・パーティション内に配置されている場合、 DB2 では、データをデータベース・パーティション間で移動する代わりに、そのデータが保管されているデータベース・パーティションで結合または副照会を実行できます。データベース・パーティションで結合または副照会を実行するこの機能によって、大きなパフォーマンス上の利点が得られます。

関連概念:

- 91 ページの『データベース・パーティション・グループ』
- 93 ページの『データベース・パーティション・グループの設計』
- 96 ページの『パーティション・キー』
- 99 ページの『パーティションの互換性』

パーティションの互換性

パーティション・キーの対応する列の基本データ・タイプを比較して、*パーティション互換* として宣言することができます。パーティション互換データ・タイプは、同じ値を持つ 2 つの変数 (それぞれのタイプに 1 つの変数) が、同じパーティション化アルゴリズムによって同じパーティション番号にマップされるというプロパティーを持っています。

パーティションの互換性は、以下の特性を持ちます。

- ある基本データ・タイプは、同じ基本データ・タイプの別のものと互換性があります。
- 内部形式は、DATE、TIME、および TIMESTAMP データ・タイプに使用されます。これらはお互いに互換性はなく、どれも CHAR との互換性はありません。
- パーティションの互換性は、 NOT NULL または FOR BIT DATA 定義を指定した列による影響を受けません。
- 互換データ・タイプの NULL 値は、同一のものとして扱われます (非互換データ・タイプの NULL 値はそのように扱われません)。
- 「ユーザー定義タイプ」という基本データ・タイプは、パーティションの互換性を分析するために使用されます。
- パーティション・キーの中の同じ値の 10 進数は、その位取りおよび精度が異なっていても、同一のものとして扱われます。
- 文字ストリング (CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC) の中の後書きブランクは、ハッシュ・アルゴリズムによって無視されます。
- BIGINT、SMALLINT と INTEGER は、互換データ・タイプです。
- REAL と FLOAT は、互換データ・タイプです。
- 異なる長さの CHAR と VARCHAR は、互換データ・タイプです。
- GRAPHIC と VARGRAPHIC は、互換データ・タイプです。

- LONG VARCHAR、LONG VARGRAPHIC、CLOB、DBCLOB、および BLOB データ・タイプはパーティション・キーとしてサポートされないため、パーティションの互換性はこれらには適用されません。

関連概念:

- 91 ページの『データベース・パーティション・グループ』
- 93 ページの『データベース・パーティション・グループの設計』
- 96 ページの『パーティション・キー』

複製されたマテリアライズ照会表

マテリアライズ照会表 は、表の中のデータの判別にも使われる照会によって定義される表です。マテリアライズ照会表を使って、照会のパフォーマンスを向上させることができます。照会の一部はマテリアライズ照会表を使って解決できると DB2® Universal Database (DB2 UDB) が判断した場合、データベース・マネージャーは、その照会がマテリアライズ照会表を使用するように書き換えます。

パーティション・データベース環境では、マテリアライズ照会表を複製することができます。照会のパフォーマンスを向上させるために、*複製されたマテリアライズ照会表* を使用できます。複製されたマテリアライズ照会表には単一パーティション・データベース・パーティション・グループで作成された表に基づくものもありますが、これを別のデータベース・パーティション・グループ内のすべてのデータベース・パーティションにわたって複製することもできます。マテリアライズ照会表を作成するには、 REPLICATED キーワードを指定して CREATE TABLE ステートメントを呼び出します。

複製されたマテリアライズ照会表を使用することによって、一般的には配置されない表の間でのコロケーションを実現できます。複製されたマテリアライズ照会表は、 1 つの大きいファクト表と小さいディメンション表のある結合について特に役立ちます。必要とされる余分のストレージを最小限にし、すべてのレプリカを更新しなければならないことによる影響を最小限にとどめるには、複製する表は小さく、頻繁に更新されるものでなければなりません。

注: また、頻繁に更新されない大きい表を複製することも考慮する必要があります。この場合、一回限りの複製に多大なコストがかかりますが、コロケーションによって得られるパフォーマンス向上によってコストは相殺されます。

複製表の定義に使われる副選択文節で適切な述部を指定することによって、選択した列、選択した行、またはその両方を複製できます。

関連概念:

- 93 ページの『データベース・パーティション・グループの設計』
- 「*管理ガイド: パフォーマンス*」の『設計アドバイザー』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『マテリアライズ照会表の作成』

関連資料:

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

表スペースの設計

表スペースは、表、索引、ラージ・オブジェクト、およびロング・データを含む、ストレージ構造です。表スペースは、データベース・パーティション・グループの中に常駐します。表スペースによって、データベースと表データのロケーションをコンテナーに直接割り当てることができます。 (コンテナーとしては、ディレクトリー名、装置名、ファイル名があります。) これによってパフォーマンスが改善され、構成の柔軟性が高くなります。

表スペースはデータベース・パーティション・グループの中にあるので、表の保持のために選択された表スペースは、その表のデータがデータベース・パーティション・グループ内の複数のデータベース・パーティションにわたって分配される方法を定義します。 1 つの表スペースが複数のコンテナーにわたる場合もあります。 (1 つまたは複数の表スペースからの) 複数のコンテナーを、同じ物理ディスク (またはドライブ) 上に作成することも可能です。パフォーマンスを向上させるためには、各コンテナーごとに異なるディスクを使用する必要があります。 図 27 は、データベース内の表と表スペース、またそのデータベースに関連するコンテナーのリレーションシップについて示しています。

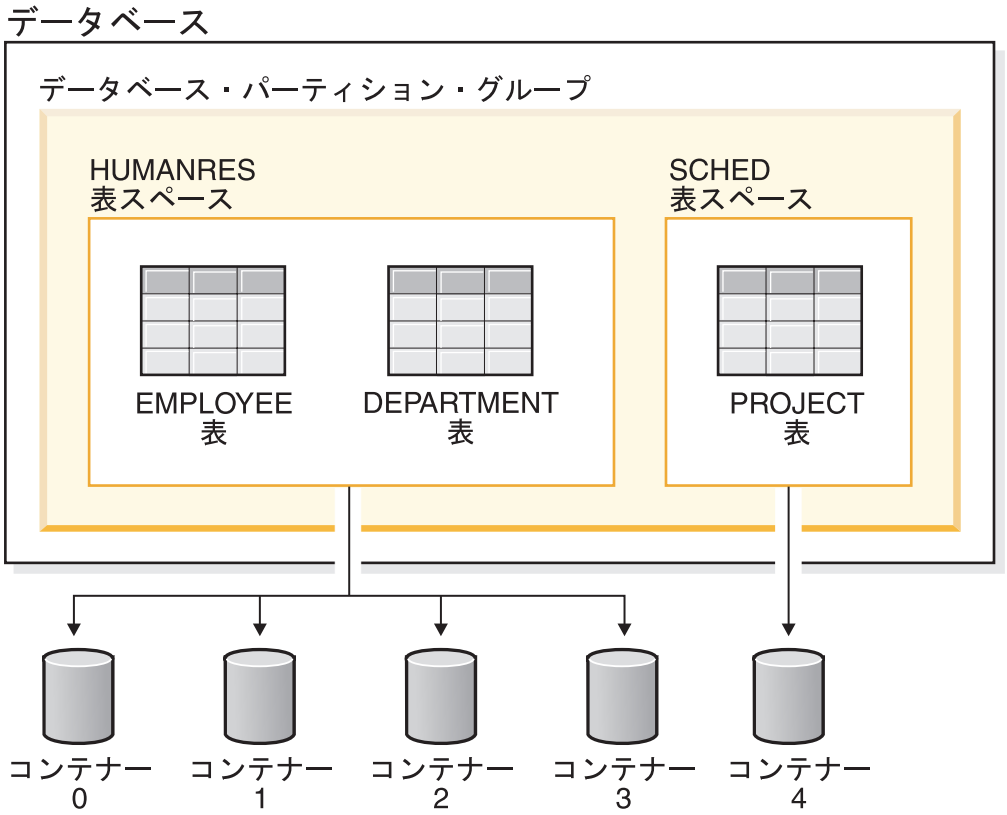

図 27. データベース内の表スペースと表

EMPLOYEE 表と DEPARTMENT 表は、コンテナー 0、1、2、および 3 にわたる HUMANRES 表スペースの中に入っています。 PROJECT 表は、コンテナー 4 の SCHED 表スペースに入っています。この例では、それぞれのコンテナーは別々のディスクの中に存在しています。

データベース・マネージャーは、コンテナー間でデータ・ロードの平衡を取ろうとします。結果として、データを格納するのにすべてのコンテナーが使われます。別のコンテナーを使用する前に、データベース・マネージャーがコンテナーに書き込むページ数は、エクステント・サイズ と呼ばれます。データベース・マネージャーは、毎回最初のコンテナーから表データを格納し始めるとは限りません。

図 28 は、エクステント・サイズが 4 KB ページ 2 つ分の HUMANRES 表スペースを表しています。それぞれのページには、割り振りエクステントが小さく設定されているコンテナーが 4 つずつあります。 DEPARTMENT 表と EMPLOYEE 表は、どちらも 7 ページあり、 4 つのコンテナーすべてにわたっています。

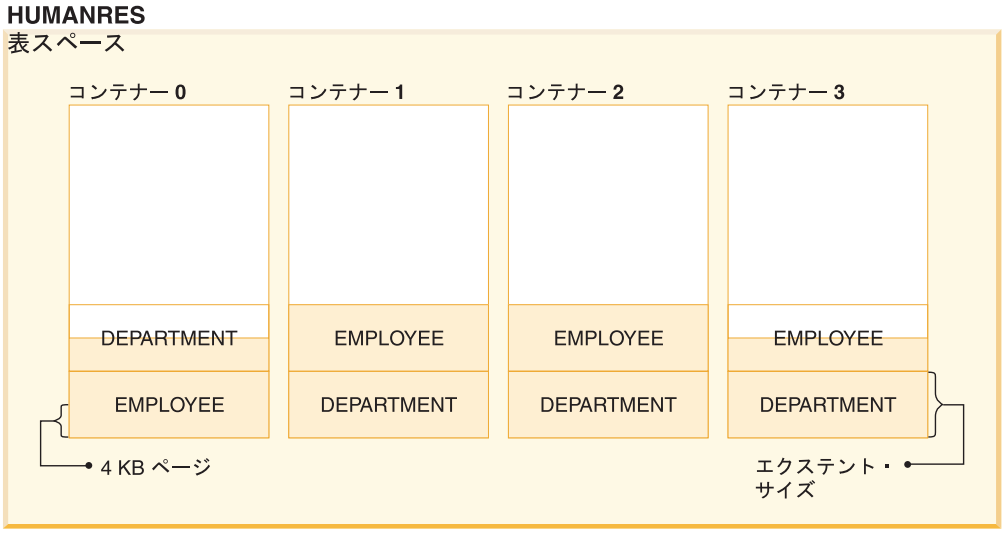

図 28. 表スペースの中のコンテナーとエクステント

1 つのデータベースには、少なくとも以下の 3 つの表スペースが含まれている必要があります。

- 1 つのカタログ表スペース。これには、データベースのすべてのシステム・カタログ表が入ります。この表スペースは SYSCATSPACE と呼ばれ、省略できません。 IBMCATGROUP は、この表スペースに対するデフォルト・データベース・パーティション・グループです。
- 1 つまたは複数のユーザー表スペース。これには、ユーザー定義のすべての表が入ります。デフォルトには、1 つの表スペース USERSPACE1 が作成されます。IBMDEFAULTGROUP は、この表スペースに対するデフォルト・データベース・パーティション・グループです。

  表スペース名は、表の作成時に指定してください。指定しない場合、望みどおりの結果が得られないかもしれません。

  表のページ・サイズは、行サイズまたは列数のいずれかによって決定されます。行の許容最大長は、表が作成された表スペースのページ・サイズに依存しています。ページ・サイズに指定できる値は 4 KB (デフォルト)、8 KB、16 KB、および 32 KB です。基本表には 1 つのページ・サイズを持つ表スペース、 LONG または LOB データにはページ・サイズの異なる別の表スペースを使用できます。 (SMS は複数の表スペースにまたがる表をサポートしませんが、 DMS はそ

のような表をサポートすることに留意してください。) 列の数または行のサイズが表スペースのページ・サイズの限界を超えると、エラーが戻されます (SQLSTATE 42997)。

- 1 つまたは複数の*一時表スペース*。一時表が入れられます。一時表スペースには、システム*一時表スペース*とユーザー*一時表スペース* があります。各データベースには、少なくとも 1 つのシステム一時表スペースが必要です。デフォルトには、データベースの作成時に TEMPSPACE1 という 1 つのシステム一時表スペースが作成されます。 IBMTEMPGROUP は、この表スペースに対するデフォルト・データベース・パーティション・グループです。ユーザー一時表スペースは、デフォルトにはデータベース作成の時点で*作成されません*。

  データベースが複数の一時表スペースを使用していて、新しい一時オブジェクトが必要になったときは、オプティマイザーが、このオブジェクトに適したページ・サイズを選択します。そして、対応するページ・サイズを持つ一時表スペースにそのオブジェクトが割り振られます。同じページ・サイズの一時表スペースが複数存在する場合は、ラウンドロビン式に表スペースが選択されます。ほとんどの環境では、1 ページ・サイズの複数の一時表スペースを持つことは推奨されていません。

  デフォルトの 4 KB よりも大きいページ・サイズで定義された表スペース内の表に対して照会が実行される (たとえば、 1012 列に対する ORDER BY) と、照会が失敗する場合があります。これは、より大きいページ・サイズで定義された一時表スペースが存在しない場合に起こります。場合によっては、さらに大きいページ・サイズ (8 KB、16 KB、または 32 KB) の一時表スペースを作成する必要があります。データ操作言語 (DML) ステートメントは、ユーザー表スペース内の最大ページ・サイズと同じページ・サイズの一時表スペースがなければ失敗する可能性があります。

  単一の SMS 一時表スペースの定義では、大半のユーザー表スペースで使用されているページ・サイズと等しいページ・サイズを指定してください。そうすれば、一般的な環境と処理に適したサイズが得られます。

パーティション・データベース環境では、カタログ・ノードは 3 つのデフォルト表スペースすべてを含み、その他のデータベース・パーティションはそれぞれ TEMPSPACE1 と USERSPACE1 だけを含みます。

表スペースには、次に示す 2 つの種類があります。 1 つのデータベースで両方を使用できます。

- システム管理スペース。オペレーティング・システムのファイル・マネージャーがストレージ・スペースを制御します。
- データベース管理スペース。データベース・マネージャーがストレージ・スペースを制御します。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『表スペースおよびその他のストレージ構造』
- 104 ページの『システム管理スペース』
- 106 ページの『データベース管理スペース』
- 126 ページの『SMS 表スペースと DMS 表スペースの比較』

- 127 ページの『表スペースのディスク入出力』
- 129 ページの『表スペースの設計における処理に関する考慮事項』
- 131 ページの『エクステント・サイズ』
- 132 ページの『表スペースとバッファー・プールとの間のリレーションシップ』
- 133 ページの『表スペースとデータベース・パーティション・グループとの間のリレーションシップ』
- 147 ページの『一時表スペースの設計』
- 149 ページの『カタログ表スペースの設計』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『表スペースの作成』
- 150 ページの『データが RAID 装置にある場合の表スペース・パフォーマンスの最適化』

関連資料:

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

システム管理スペース

SMS (システム管理スペース) 表スペースでは、オペレーティング・システムのファイル・システム・マネージャーが、表の保管されるスペースの割り振りと管理を行います。ストレージ・モデルは、通常、ファイル・システム・スペースに保管された表オブジェクトを表す多くのファイルからなります。ユーザーがファイルのロケーションを決定し、DB2® Universal Database (DB2 UDB) がそれらの名前を制御し、そしてファイル・システムがそれらを管理する責任を持ちます。データベース・マネージャーは、各ファイルに書き込まれるデータの量を制御することにより、それぞれの表スペース・コンテナーにデータを均等に分配します。デフォルトでは、データベース作成時間に作成された初期表スペースは SMS です。

各表には、それと関連した少なくとも 1 つの SMS 物理ファイルがあります。

挿入のパフォーマンスを向上させる必要がある場合は、複数ページのファイル割り振りを検討する必要があります。これによって、システムは、一度に (1 ページずつではなく) 1 エクステントずつファイルの割り振りまたは拡張を行うことができます。パフォーマンスのために、 SMS 表スペース内にマルチディメンション・クラスタリング (MDC) 表を保管する場合、複数ページ・ファイル割り振りを使用可能にする必要があります。バージョン 8.2 以降でデータベース (パーティション・データベースを含む) を作成する場合、複数ページ・ファイル割り振りはデフォルトで有効になっています。しかし、DB2_NO_MPFA_FOR_NEW_DB レジストリー変数がオンになっている場合には、新しいデータベースの作成時に複数ページ・ファイル割り振りがデフォルトでないことがあります。データベースに対して複数ページ・ファイル割り振りを有効にするには、 `db2empfa` ユーティリティーを使用します。パーティション・データベース環境では、このユーティリティーを各データベース・パーティションで実行する必要があります。複数ページのファイル割り振りがいったん使用可能になると、それを使用不能にすることはできません。

SMS 表スペースは、 CREATE DATABASE コマンドまたは CREATE TABLESPACE ステートメントの MANAGED BY SYSTEM オプションを使用して定義されます。 SMS 表スペースを定義するときは、かぎとなる以下の 2 つの要素を考慮に入れる必要があります。

- 表スペース用のコンテナー

  表スペースで使用するコンテナーの数を指定しなければなりません。 SMS 表スペースが作成された後ではコンテナーを追加または削除することができないため、使用したいすべてのコンテナーを確認しておくことがきわめて重要です。パーティション・データベース環境では、 SMS 表スペースのデータベース・パーティション・グループに新しいパーティションが追加されたときに、 ALTER TABLESPACE ステートメントを使用して、新しいパーティションにコンテナーを追加することができます。

  SMS 表スペースの各コンテナーは、絶対または相対ディレクトリー名を識別します。これらのディレクトリーは、それぞれ別個のファイル・システム (または物理ディスク) に置くことができます。表スペースの最大サイズは次のように見積もることができます。

  ```
  number of containers * (maximum file system size
     supported by the operating system)
  ```

  この公式は、各コンテナーごとに別個のファイル・システムがマップされており、各ファイル・システムには使用できるスペースの上限があることを前提とします。現実にはそのようなことはあまりなく、多くの場合、表スペースの最大サイズはもっと小さくなります。データベース・オブジェクトのサイズにおける SQL 制限もあります。これは、表スペースの最大サイズに影響を与える場合があります。

  注: コンテナーを定義するときには注意が必要です。コンテナーにファイルやディレクトリーがすでに存在する場合は、エラー (SQL0298N) が戻されます。

- 表スペースのエクステント・サイズ

  エクステント・サイズの指定は、表スペースの作成時にしか行えません。後から変更することはできませんので、適切なエクステント・サイズを選択してください。

  表スペースを作成するときにエクステント・サイズを指定しなければ、データベース・マネージャーは、 *dft_extent_sz* データベース構成パラメーターで定義されているデフォルト・エクステント・サイズを使って表スペースを作成します。この構成パラメーターは、データベースの作成時に指定した情報をもとに初期設定されます。 CREATE DATABASE コマンドで *dft_extent_sz* パラメーターを指定しなければ、デフォルト・エクステント・サイズは 32 に設定されます。

コンテナーの数および表スペースのエクステント・サイズに適切な値を選択するには、以下のことを理解しておく必要があります。

- 使用しているオペレーティング・システムでの、論理ファイル・システムのサイズ制限。

たとえば、一部のオペレーティング・システムでは、2 GB の制限があります。そのため、64 GB の表オブジェクトを希望する場合は、このタイプのシステムでは少なくとも 32 のコンテナーが必要です。

表スペースを作成するときに、コンテナーが別々のファイル・システムに存在するよう指定すれば、データベースに格納できるデータ量を増やすことができます。

- データベース・マネージャーが、表スペースと関連したデータ・ファイルおよびコンテナーを管理する方法。

  最初の表データ・ファイル (SQL00001.DAT) は、その表スペースに指定された最初のコンテナーの中に作成され、このファイルは、エクステント・サイズまで拡張することが許されます。そのサイズまで達したら、データベース・マネージャーは次のコンテナーの SQL00001.DAT にデータを書き込みます。このプロセスは、すべてのコンテナーに SQL00001.DAT ファイルが入るまで続きます。その後、データベース・マネージャーは最初のコンテナーに戻ります。このプロセス (ストライピング という) は、コンテナーが満杯になるか (SQL0289N)、またはオペレーティング・システムがそれ以上スペースを割り振れなくなる (ディスク満杯エラー) まで、コンテナー・ディレクトリーにわたって続行されます。また、ストライピングは索引 (SQL*nnnnn*.INX)、ロング・フィールド (SQL*nnnnn*.LF)、および LOB (SQL*nnnnn*.LB および SQL*nnnnn*.LBA) ファイルにも使用されます。

  注: SMS 表スペースは、そのコンテナーのうちのいずれか 1 つが満杯になると、ただちに満杯になります。したがって、各コンテナーに同じ量のスペースを割り当てることが重要です。

  複数のコンテナーにわたってデータをより均等に分散させるのに役立てるため、データベース・マネージャーは、表の ID (上記の例では SQL00001.DAT) とコンテナー数を考慮することにより、最初に使用するコンテナーを判別しています。コンテナーは 0 から順番に番号が付けられます。

関連概念:

- 101 ページの『表スペースの設計』
- 106 ページの『データベース管理スペース』
- 126 ページの『SMS 表スペースと DMS 表スペースの比較』

関連資料:

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

データベース管理スペース

DMS (データベース管理スペース) 表スペースでは、データベース・マネージャーがストレージ・スペースを制御します。ストレージ・モデルは、DB2® Universal Database (DB2 UDB) によってスペースが管理される、限定された数の装置またはファイルからなります。データベース管理者は、どの装置およびファイルを使用するかを決定し、 DB2 UDB がそれらの装置およびファイル上のスペースを管理します。この表スペースは基本的に、データベース・マネージャーの必要を最もよく満たすように設計された特別な目的のファイル・システムを実現したものです。

ユーザー定義の表およびデータが入っている DMS 表スペースは、以下のものとして定義することができます。

- 通常表データと索引データを格納する *REGULAR* 表スペース。
- ロング・フィールド、LOB データ、または索引データを格納する *LARGE* 表スペース。

DMS 表スペースおよびコンテナーを設計するときは、以下の要素を考慮してください。

- データベース・マネージャーは、ストライピングを使用して、すべてのコンテナーにわたって均等にデータを分散させるようにしています。
- REGULAR 表スペースの最大サイズは、4 KB ページの場合は 64 GB、 8 KB ページの場合は 128 GB、16 KB ページの場合は 256 GB、32 KB ページの場合は 512 GB です。 LARGE 表スペースの最大サイズは 2 TB です。
- SMS 表スペースとは異なり、 DMS 表スペースを構成するコンテナーのサイズはすべて同じにする必要はありません。しかし、サイズが異なると、コンテナー間のストライピングが不均一になり、最適なパフォーマンスが得られるとは限らないため、通常は推奨されていません。いずれかのコンテナーが満杯である場合、DMS 表スペースは、他のコンテナーからの使用可能なフリー・スペースを使用します。
- スペースは事前に割り振られるため、表スペースを作成する前に、スペースが利用可能になっていなければなりません。装置コンテナーを使用する場合は、コンテナーを定義するのに十分なスペースの装置が存在していなければなりません。それぞれの装置には、コンテナーを 1 つだけ定義することができます。スペースを無駄にしないために、装置のサイズとコンテナーのサイズが等しくなるようにしてください。たとえば、ある装置に 5 000 ページが割り振られているのに、装置コンテナーに 3 000 ページしか割り振らないと、その装置の 2 000 ページは使用できなくなります。
- デフォルトでは、すべてのコンテナー内の 1 つのエクステントがオーバーヘッドのために予約されます。フル・エクステントしか使用されないため、最適なスペース管理を実現するには、コンテナーを割り振るときに以下の公式を使って適切なサイズを決定してください。

  ```
  extent_size * (n + 1)
  ```

  extent_size は表スペース内の各エクステントのサイズ、 *n* はコンテナーに格納するエクステントの数を表します。
- DMS 表スペースの最小サイズは、5 つのエクステントです。 5 つのエクステントよりも小さい表スペースを作成しようとすると、エラー (SQL1422N) が生じます。
  - 表スペース内のエクステント 3 つはオーバーヘッドのために確保されています。
  - いずれのユーザー表データを格納するにも、最低で 2 つのエクステントが必要です。 (これらのエクステントは 1 つの表の正規データを格納するためのものです。専用のエクステントを必要とする、索引、ロング・フィールド、またはラージ・オブジェクトのためではありません。)
- 装置コンテナーは、物理ボリュームではなく、論理ボリュームに設定された名前を使用します。

- DMS 表スペースでは、装置の代わりにファイルを使用することができます。ファイルと装置の間に操作上の違いはありません。しかし、ファイル・システムに関連するランタイム・オーバーヘッドのために、ファイルの方が効率が悪くなる可能性があります。以下の場合には、ファイルの方が便利です。
  - 装置が直接サポートされていない
  - 装置が使用可能でない
  - 最大パフォーマンスは必要ない
  - 自分で装置をセットアップしたくない
- 実際のワークロードに LOB または LONG VARCHAR データが含まれる場合、ファイル・システムのキャッシュによってパフォーマンスが改善されることがあります。 LOB および LONG VARCHAR は DB2 UDB のバッファー・プールには入れられないことに注意してください。
- オペレーティング・システムによっては、 2 GB より大きいサイズの物理装置を持つことができるものがあります。物理装置を複数の LU にパーティション化して、オペレーティング・システムによって許されるサイズより大きなコンテナーがないようにする必要があります。

関連概念:

- 101 ページの『表スペースの設計』
- 104 ページの『システム管理スペース』
- 126 ページの『SMS 表スペースと DMS 表スペースの比較』
- 108 ページの『表スペース・マップ』
- 112 ページの『DMS 表スペースでコンテナーを追加/拡張する方法』

表スペース・マップ

表スペース・マップは DMS 表スペースを表す DB2® Universal Database (DB2 UDB) の内部表記であり、表スペースにおける論理ページ・ロケーションから物理ページ・ロケーションへの変換を記述しています。以下では、表スペース・マップが役立つ理由と、表スペース・マップの情報がどこから得られるかを説明します。

DB2 UDB データベースでは、 DMS 表スペース内の各ページに 0 から N-1 までの番号が論理的に付けられています (N は、表スペース内の使用できるページ数)。

DMS 表スペース内のページは、エクステント・サイズに基づいて各エクステントにグループ分けされ、表スペース管理の観点から言うと、すべてのオブジェクト割り振りはエクステント・ベースで行われます。つまり、表があるエクステントの半分のページしか使っていなくても、そのエクステント全体が使用中で、そのオブジェクトに所有されていると見なされます。デフォルトでは、コンテナー・タグを保持するために 1 つのエクステントが使用され、このエクステント内のすべてのページはデータを保持できません。ただし、レジストリー変数 DB2_USE_PAGE_CONTAINER_TAG をオンにすれば、コンテナー・タグに使用されるのは 1 ページだけになります。

コンテナー内のスペースはエクステント単位で割り振られるため、完全なエクステントを形成しないページは使用されません。たとえば、205 ページからなるコンテ

ナーがあり、エクステント・サイズが 10 である場合、 1 つのエクステントがタグ用に使用され、 19 個のエクステントにデータが保持されます (残る 5 ページはむだになります)。

DMS 表スペースにただ 1 つのコンテナーが含まれている場合、論理ページ番号からディスク上の物理ロケーションへの変換プロセスは単純で、ページ 0、1、2 はディスク上に同じ順番で格納されます。

複数のコンテナーが存在し、各コンテナーのサイズが同じである場合にも、変換プロセスはかなり単純です。表スペースの最初のエクステント (ページ 0 から [エクステント・サイズ - 1] ページまでを含む) は最初のコンテナーに格納され、 2 番目のエクステントは 2 番目のコンテナーに格納されます (以下同様)。最後のコンテナーに来るとラウンドロビン方式で処理が繰り返され、最初のコンテナーから再び開始されます。

さまざまなサイズのコンテナーが含まれる表スペースの場合、より大きなコンテナーの余分のスペースが利用されないため、単純なラウンドロビン方式は使用できません。このような場合、表スペース・マップが役立ちます。表スペース・マップは、表スペース内でエクステントがどのように配置されるかを管理し、物理コンテナー内のすべてのエクステントが利用されるようにします。

注: 以下の例では、コンテナー・サイズを決定する上でコンテナー・タグのサイズは考慮されていません。また、目的は単に例を示すことなので、かなり小さなコンテナー・サイズを使っていますが、そのサイズをお勧めするという意味ではありませんのでご注意ください。さらに、 1 つの表スペース内でいろいろなサイズのコンテナーを使っていますが、実際には、同じサイズのコンテナーを使用することをお勧めします。

例 1:

表スペースに 3 つのコンテナーがあり、各コンテナーには 80 個の使用可能なページが含まれ、表スペースのエクステント・サイズは 20 です。したがって各コンテナーには 4 つのエクステント (80/20) が含まれ、合計で 12 個のエクステントが存在します。これらのエクステントは、 110 ページの図 29 のようにディスクに配置されます。

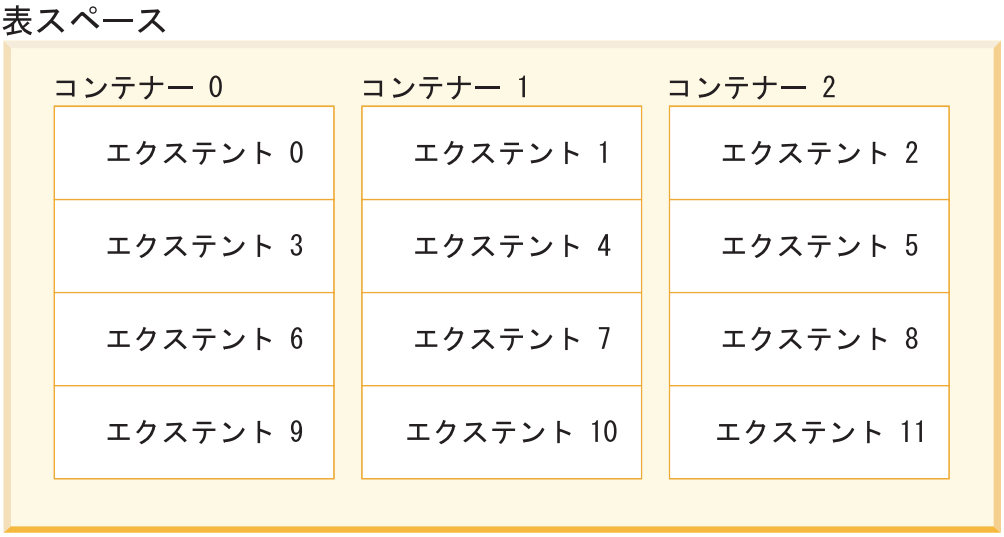

図 29. 3 つのコンテナー、12 のエクステントが含まれる表スペース

表スペース・マップを表示するには、スナップショット・モニターを使って表スペースのスナップショットを取ってください。例 1 では 3 つのコンテナーのサイズが等しく、表スペース・マップは以下のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0      11      239        0        3    0   3 (0, 1, 2)
```

範囲 (range) とはマップの一部であり、ストライプ内の隣接する範囲の中にすべて同じコンテナー・セットが含まれます。例 1 では、すべてのストライプ (0 から 3 まで) に 3 つのコンテナー (0、1、2) からなる同じセットが含まれ、これが 1 つの範囲と見なされます。

表スペース・マップの見出しは、Range Number (範囲番号)、 Stripe Set (ストライプ・セット)、Stripe Offset (ストライプ・オフセット)、 Maximum extent number addressed by the range (範囲でアドレス指定される最大エクステント番号)、Maximum page number addressed by the range (範囲でアドレス指定される最大ページ番号)、 Start Stripe (開始ストライプ)、End Stripe (終了ストライプ)、 Range adjustment (範囲調整)、および Container list (コンテナー・リスト) です。これらについては、例 2 で詳しく説明されます。

この表スペースは 111 ページの図 30 のように表すこともできます。この図では、各垂直線がコンテナー、各水平線がストライプ、各セル番号がエクステントです。

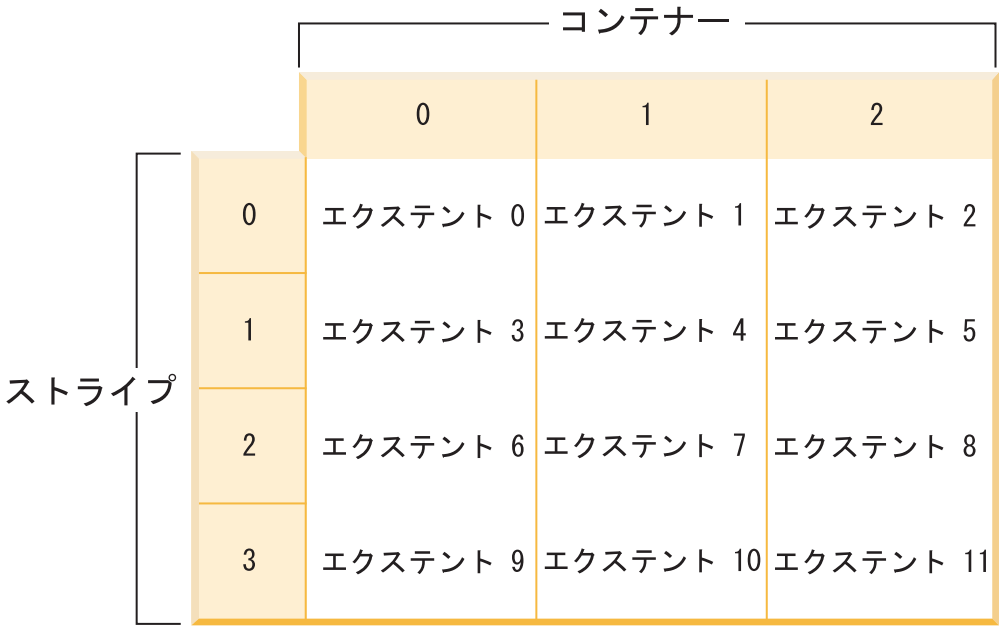

図 30. 3 つのコンテナー、 12 のエクステントが含まれる表スペース (ストライプを強調した場合)

例 2:

表スペースには 2 つのコンテナーがあり、最初のコンテナーのサイズは 100 ページ、2 番目のサイズは 50 ページ、そしてエクステント・サイズは 25 です。つまり最初のコンテナーには 4 つのエクステント、 2 番目のコンテナーには 2 つのエクステントが含まれます。この表スペースは、図 31 のように表すことができます。

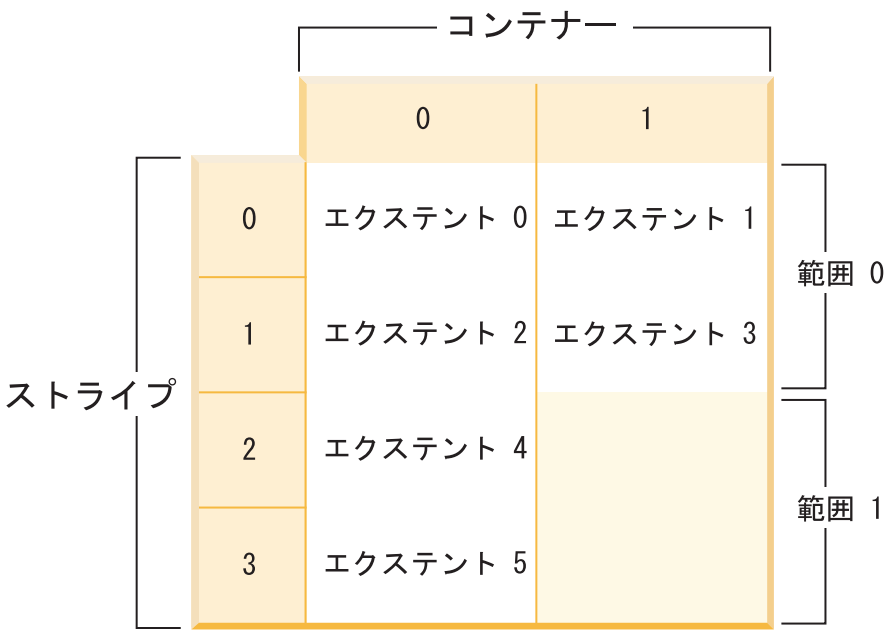

図 31. 2 つのコンテナーが含まれる表スペース (範囲を強調した図)

ストライプ 0 および 1 には両方のコンテナー (0、1) が含まれますが、ストライプ 2 および 3 には最初のコンテナー (0) しか含まれません。この 2 つのストライプ・セットが、それぞれ範囲になります。表スペース・マップは、以下の表スペース・スナップショットのようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       3       99        0        1    0   2 (0, 1)
   [1]       [0]       0       5      149        2        3    0   1 (0)
```

最初の範囲には 4 つのエクステントが存在するので、この範囲でアドレス指定される最大エクステント番号 (Max Extent) は 3 です。各エクステントには 25 ページが含まれるので、最初の範囲には全部で 100 ページが存在します。ページ番号もまた 0 から始まるので、この範囲でアドレス指定される最大ページ番号 (Max Page) は 99 になります。この範囲の最初のストライプ (開始ストライプ) は 0、最後のストライプ (終了ストライプ) は 1 です。この範囲には、2 つのコンテナー (0 と 1) があります。ストライプ・オフセットは、ストライプ・セットの最初のストライプです (この場合はただ 1 つのストライプ・セットしか存在しないので 0)。範囲調整 (Adj.) とは、表スペースの中でデータをリバランスするときに使われるオフセットです。 (表スペースでスペースが追加またはドロップされるとき、リバランスが行われる場合があります。) リバランスが行われない限り、この値は常に 0 です。

2 番目の範囲には 2 つのエクステントがあり、前の範囲内でアドレス指定される最大エクステント番号が 3 であるため、この範囲でアドレス指定される最大エクステント番号は 5 です。 2 番目の範囲には全部で 50 ページ (2 エクステント × 25 ページ) 存在し、前の範囲内でアドレス指定される最大ページ番号が 99 であるため、この範囲でアドレス指定される最大ページ番号は 149 です。この範囲はストライプ 2 で始まり、ストライプ 3 で終わります。

関連概念:

- 106 ページの『データベース管理スペース』
- 「システム・モニター *ガイドおよびリファレンス*」の『スナップショット・モニター』
- 112 ページの『DMS 表スペースでコンテナーを追加/拡張する方法』
- 123 ページの『DMS 表スペースでコンテナーをドロップ/縮小する方法』

関連資料:

- 「コマンド・リファレンス」の『GET SNAPSHOT コマンド』

DMS 表スペースでコンテナーを追加/拡張する方法

表スペースを作成すると、その表スペース・マップが作成され、すべての初期コンテナーがストライプ 0 で開始するようにラインアップします。したがって、個々の表スペース・コンテナーが満杯になるまで、データはすべての表スペース・コンテナー間で均等にストライピングされます。 (114 ページの例 1 を参照。)

ALTER TABLESPACE ステートメントを使用すれば、既存の表スペースにコンテナーを追加したり、コンテナーを拡張して記憶容量を増やしたりできます。

既存のコンテナーよりも小さいコンテナーを追加すると、データの分散が不均等になります。この結果、等しいサイズのコンテナーの場合よりも、データの事前取り出しなどの並列入出力の実行の効率が低下します。

表スペースに新しいコンテナーを追加したり、既存のコンテナーを拡張したりすると、表スペース・データのリバランスという処理が発生する場合があります。

リバランス

コンテナーの追加または拡張を実行したときのリバランスの処理では、表スペースのエクステントが 1 つのロケーションから別のロケーションに移動します。目的は、表スペース内のデータのストライピングをきちんと調整することです。

リバランスの処理中も、表スペースへのアクセスは制限されません。通常どおり、オブジェクトのドロップ、作成、データ挿入、照会ができます。しかし、リバランスの処理は、パフォーマンスに大きな影響を与えます。複数のコンテナーを追加する必要があって、しかもコンテナーのリバランスを実行するつもりであれば、 1 つの ALTER TABLESPACE ステートメントですべてのコンテナーを同時に追加するべきです。そうすれば、データベース・マネージャーがデータのリバランスを何度も行う必要はなくなります。

リバランスの処理では、表スペースの最高水準点が重要な役割を果たします。この場合の最高水準点とは、表スペース内で割り振られている最高ページのページ番号です。たとえば、 1000 ページの表スペースがあって、エクステントのサイズが 10 であれば、エクステントの数は 100 になります。その表スペース内で割り振られている最高エクステントが 42 番目のエクステントであれば、最高水準点は、42 * 10 = 420 ページということになります。これは、使用済みページの数と同じではありません。最高水準点よりも低い位置にあるエクステントの中には、再利用のために解放されるものもあるからです。

このリバランスが始まる前に、コンテナーの変更内容に基づいて、新しい表スペース・マップが作成されます。リバランスとは、現在のマップで決められているロケーションから、新しいマップで決められているロケーションにエクステントを移す処理です。リバランスの処理は、エクステント 0 から始まり、エクステントを 1 つずつ移していき、最後に最高水準点を含んでいるエクステントを移します。エクステントを移すたびに、現在のマップが新しいマップに合わせて 1 箇所ずつ変更されていきます。リバランスの処理が完了した時点で、現在のマップと新しいマップは、最高水準点を含んでいるストライプの位置までまったく同じになっているはずです。現在のマップは、新しいマップとまったく同じになり、リバランスの処理は完了します。現在のマップ内のエクステントのロケーションが新しいマップ内のロケーションと同じ場合は、エクステントの移動は行われず、 I/O も発生しません。

新しいコンテナーを追加した場合、新しいマップ内でのそのコンテナーの位置は、そのコンテナーのサイズと、そのストライプ・セット内の他のコンテナーのサイズによって決まります。追加するコンテナーが、ストライプ・セット内の最初のストライプから始まり、ストライプ・セット内の最後のストライプで終わる (または最後のストライプを超える) だけのサイズを持っていれば、そのコンテナーはそのような形で配置されます (115 ページの例 2 を参照)。そのコンテナーがそれだけのサイズを持っていなければ、マップ内では、そのコンテナーがストライプ・セット内

の最後のストライプで終わるように配置されます (118 ページの例 4を参照)。このような動作になっているのは、リバランスを必要とするデータの量を最小限に抑えるためです。

注: 以下の例では、コンテナー・サイズを決定する上でコンテナー・タグのサイズは考慮されていません。また、目的は単に例を示すことなので、かなり小さなコンテナー・サイズを使っていますが、そのサイズをお勧めするという意味ではありませんのでご注意ください。さらに、 1 つの表スペース内でいろいろなサイズのコンテナーを使っていますが、実際には、同じサイズのコンテナーを使用することをお勧めします。

例 1:

3 つのコンテナーを含んだ表スペースを作成するとします。エクステントのサイズは 10、 3 つのコンテナーのサイズはそれぞれ 60 ページ、40 ページ、 80 ページ (6 エクステント、4 エクステント、 8 エクステント) です。この場合にマップに基づいて作成される表スペースを図にすると、 115 ページの図 32 のようになります。

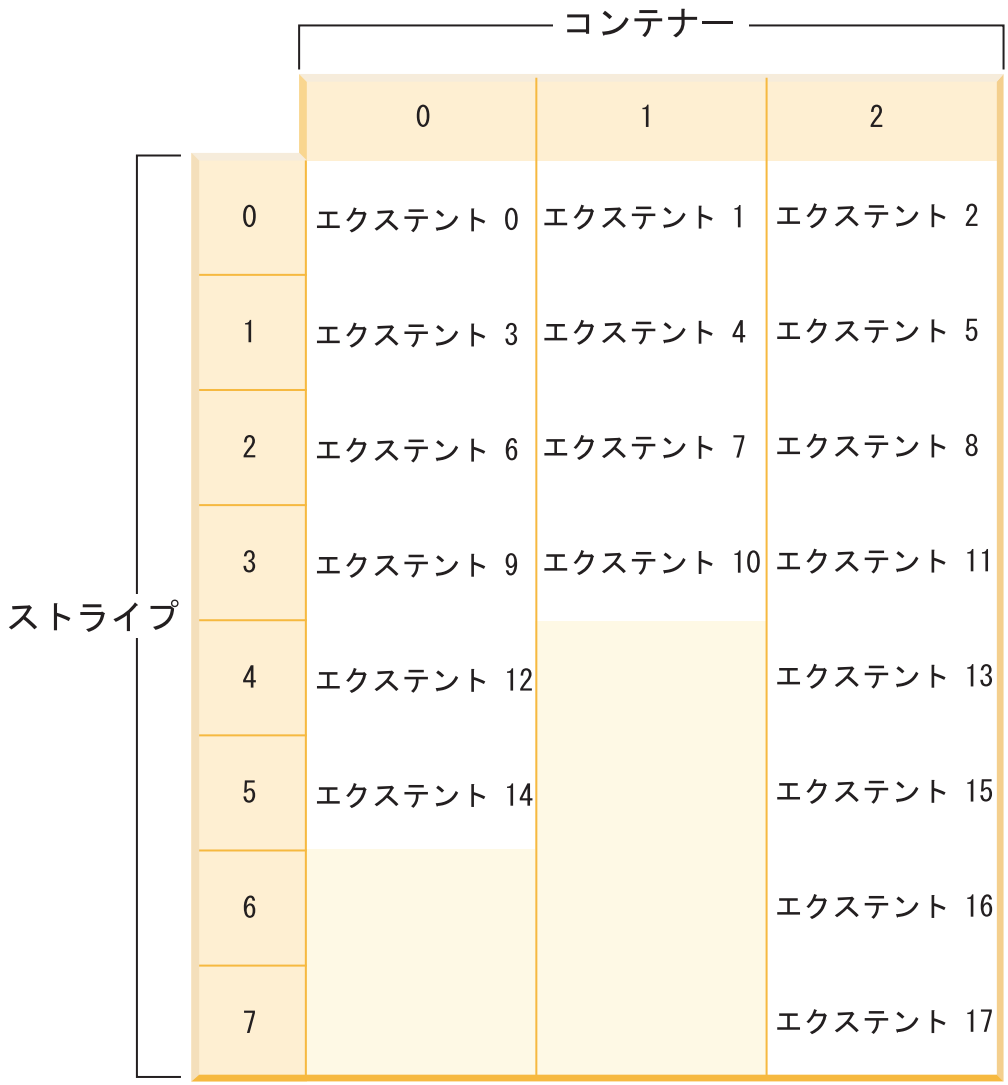

図 32. 3 つのコンテナー、18 のエクステントが含まれる表スペース

表スペースのスナップショットから分かるとおり、対応する表スペース・マップは、次のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0      11      119        0        3    0   3 (0, 1, 2)
   [1]       [0]       0      15      159        4        5    0   2 (0, 2)
   [2]       [0]       0      17      179        6        7    0   1 (2)
```

表スペース・マップの見出しは、Range Number (範囲番号)、 Stripe Set (ストライプ・セット)、Stripe Offset (ストライプ・オフセット)、 Maximum extent number addressed by the range (範囲でアドレス指定される最大エクステント番号)、Maximum page number addressed by the range (範囲でアドレス指定される最大ページ番号)、 Start Stripe (開始ストライプ)、End Stripe (終了ストライプ)、 Range adjustment (範囲調整)、および Container list (コンテナー・リスト) です。

例 2:

例 1 の表スペースに 80 ページのコンテナーを追加するとしましょう。この場合、追加するコンテナーは、最初のストライプ (ストライプ 0) から始まり、最後のストライプ (ストライプ 7) で終わるだけのサイズになっています。したがって、このコンテナーは、最初のストライプから始まる位置に配置されます。結果として生成される表スペースは、 図 33 の図のようになります。

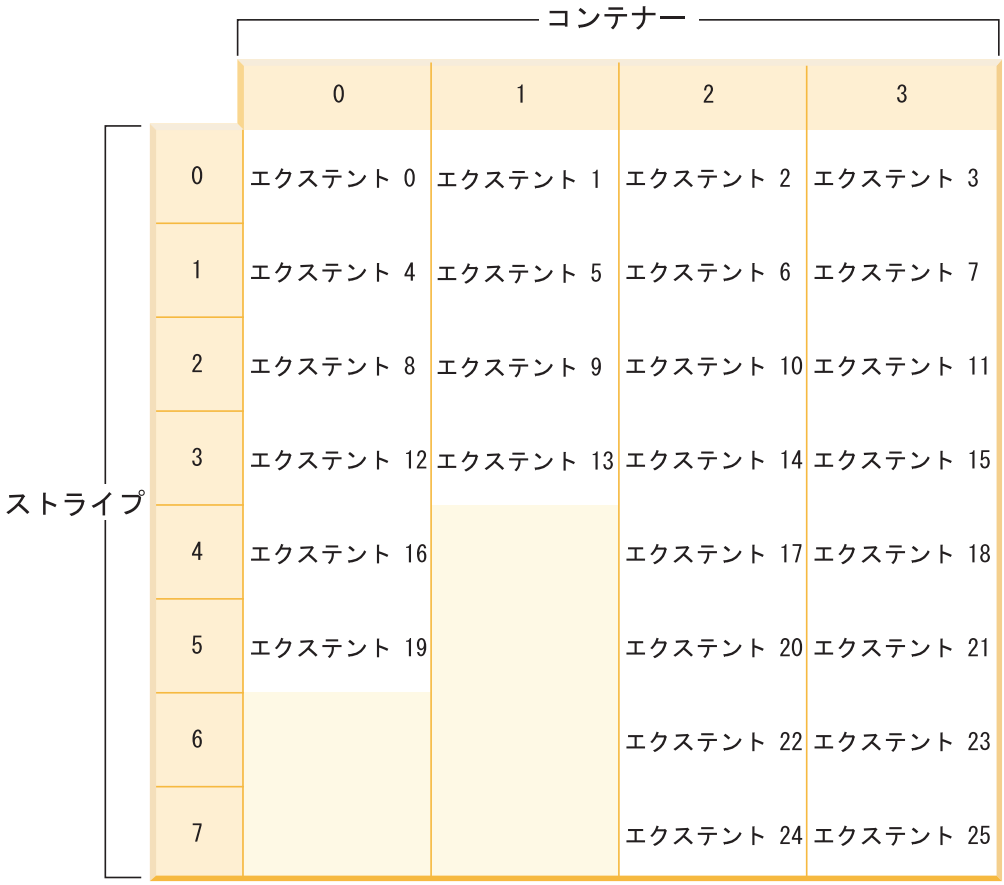

図 33. 4 つのコンテナー、26 のエクステントが含まれる表スペース

表スペースのスナップショットから分かるとおり、対応する表スペース・マップは、次のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0      15      159        0       3    0   4 (0, 1, 2, 3)
   [1]       [0]       0      21      219        4       5    0   3 (0, 2, 3)
   [2]       [0]       0      25      259        6       7    0   2 (2, 3)
```

ここで、最高水準点がエクステント 14 にあるとすれば、リバランスの処理は、エクステント 0 から始まり、エクステント 14 までのすべてのエクステントが移動することになります。両方のマップのエクステント 0 のロケーションは同じなので、このエクステントは、動かす必要がありません。エクステント 1 と 2 についても同じことが言えます。一方、エクステント 3 は動かす必要があります。したがって、古いロケーション (コンテナー 0 の 2 番目のエクステント) から読み取られて、新しいロケーション (コンテナー 3 の 1 番目のエクステント) に書き込まれま

す。このあと、エクステント 14 までのすべてのエクステントが移動します。エクステント 14 が移動した時点で、現在のマップが新しいマップと同じになって、リバランスの処理が終了します。

この場合、新しく追加するスペースをすべて最高水準点の後に入れるようにマップを変更すれば、リバランスの処理は不要になり、すべてのスペースがすぐに使用できる状態になります。一方、一部のスペースだけを最高水準点の後に入れるようにマップを変更すれば、最高水準点よりも上の位置にあるストライプ内のスペースだけがすぐに使用できる状態になります。残りのスペースは、リバランスの処理が完了するまで使用できません。

コンテナーを拡張する場合も、リバランスの動作は同じです。ストライプ・セット内の最後のストライプを超えてコンテナーを拡張する場合は、それに合わせてストライプ・セットが拡張され、その後のストライプ・セットもシフトアウトすることになります。したがって、コンテナーが後続のストライプ・セットに入り込むことはありません。

例 3:

例 1 の表スペースを再び取り上げましょう。コンテナー 1 を 40 ページから 80 ページに拡張すると、新しい表スペースは、 118 ページの図 34 のようになります。

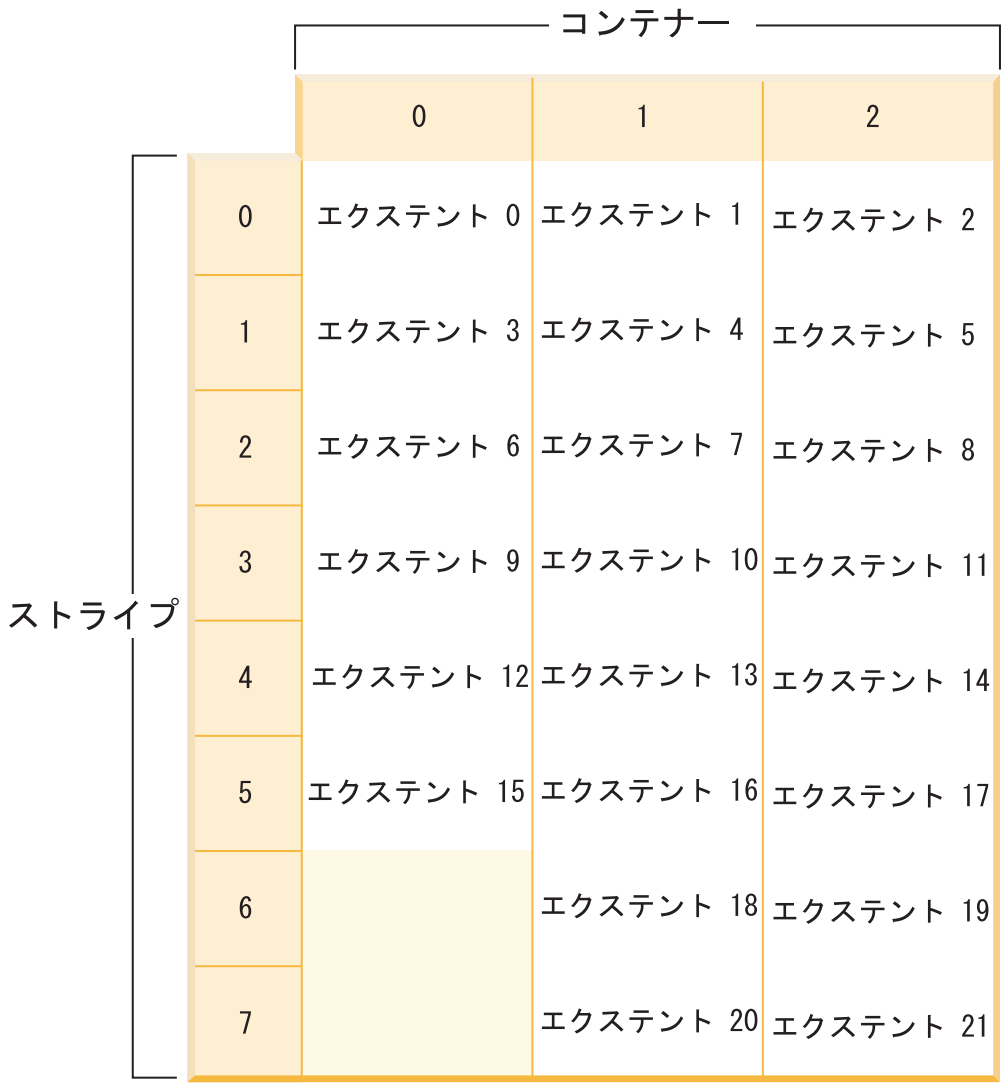

図 34. 3 つのコンテナー、22 のエクステントが含まれる表スペース

表スペースのスナップショットから分かるとおり、対応する表スペース・マップは、次のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0      17      179        0        5    0   3 (0, 1, 2)
   [1]       [0]       0      21      219        6        7    0   2 (1, 2)
```

例 4:

114 ページの例 1 の表スペースをさらに取り上げましょう。その表スペースに 50 ページ (5 エクステント) のコンテナーを追加すると、そのコンテナーは新しいマップに次のような要領で追加されます。この場合のコンテナーは、最初のストライプ (ストライプ 0) から始まり、最後のストライプ (ストライプ 7) で終わる (または最

後のストライプを超える) だけのサイズになっていないので、最後のストライプで終わるような位置に配置されます。 (図 35 を参照。)

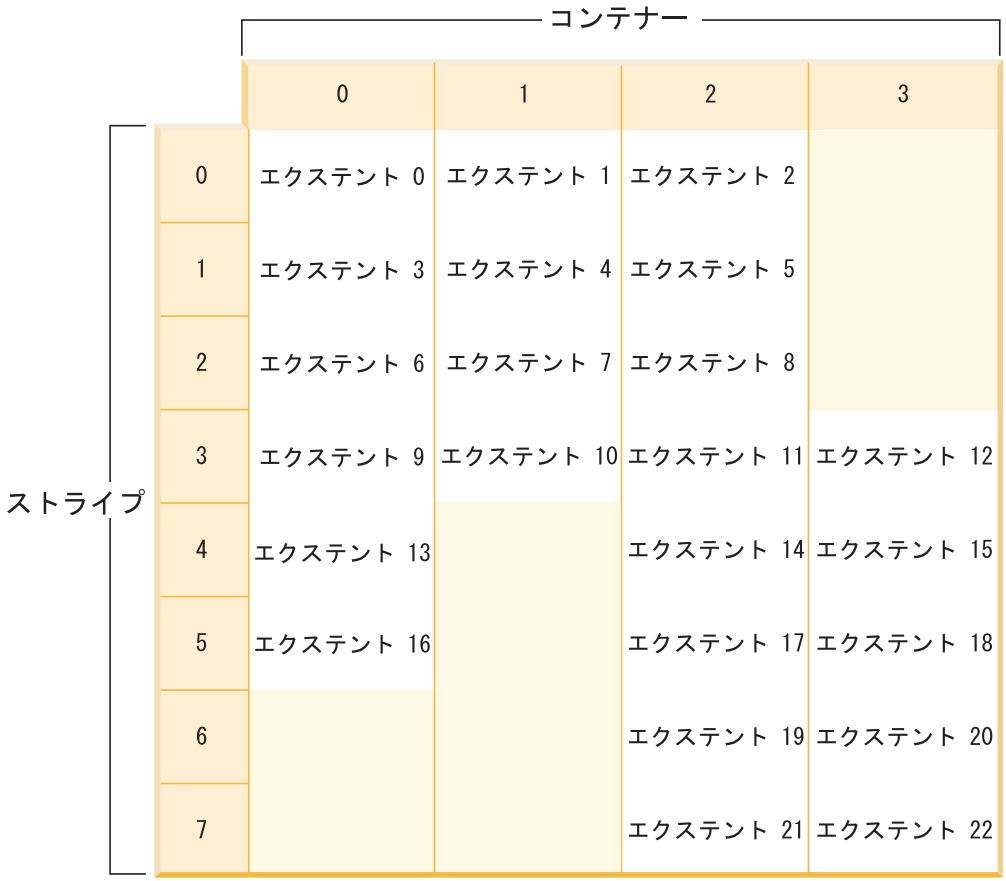

図 35. 4 つのコンテナー、23 のエクステントが含まれる表スペース

表スペースのスナップショットから分かるとおり、対応する表スペース・マップは、次のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       8       89        0        2    0   3 (0, 1, 2)
   [1]       [0]       0      12      129        3        3    0   4 (0, 1, 2, 3)
   [2]       [0]       0      18      189        4        5    0   3 (0, 2, 3)
   [3]       [0]       0      22      229        6        7    0   2 (2, 3)
```

コンテナーを拡張するには、 ALTER TABLESPACE ステートメントで EXTEND オプションか RESIZE オプションを使用します。コンテナーを追加して、データのリバランスの処理を実行するには、 ALTER TABLESPACE ステートメントで ADD オプションを使用します。複数のストライプ・セットがある表スペースにコンテナーを追加する場合は、どのストライプ・セットに追加するかを指定できます。そのためには、 ALTER TABLESPACE ステートメントで ADD TO STRIPE SET オプションを使用します。ストライプ・セットを指定しないと、デフォルトの動作として、現在のストライプ・セットにコンテナーが追加されます。現在のストライプ・

セットとは、最後に作成されたストライプ・セットであり、最後にスペースが追加されたストライプ・セットではありません。

ストライプ・セットを変更すると、そのストライプ・セットと後続のストライプ・セットに対して、リバランスの処理が実行される場合があります。

リバランスの処理の進行状況をモニターするには、表スペースのスナップショットを使用します。表スペースのスナップショットからは、リバランスの開始時刻、移動したエクステントの数、移動する必要があるエクステントの数など、リバランスの処理についての情報が得られます。

リバランスを回避する方法 (ストライプ・セットの使用)

コンテナーの追加または拡張を実行する場合でも、表スペースの最高水準点を超えた位置にスペースを追加するのであれば、リバランスの処理は実行されません。

コンテナーを追加する場合は、最高水準点の下の位置にスペースを追加するケースがほとんどです。したがって、コンテナーを追加したときには、リバランスの処理が必要になるケースが多いわけです。しかし、最高水準点よりも上の位置に新しいコンテナーを強制的に追加するための方法があります。この方法を使用すれば、表スペースのコンテンツのリバランスの処理を回避できます。このような方法の利点は、新しいコンテナーをすぐに使用できるということです。リバランスの処理を回避するためのオプションは、コンテナーを追加する場合にのみ使用できるもので、既存のコンテナーを拡張する場合には使用できません。コンテナーを拡張する場合は、最高水準点よりも上の位置にスペースを追加するときに限り、リバランスの処理が回避されます。たとえば、同じサイズのコンテナーがいくつかある状況で、それぞれのコンテナーを同じ量だけ拡張する場合、エクステントの相対位置は変わらないため、リバランスの処理は発生しません。

コンテナーを追加するときにリバランスの処理を回避するには、新しいストライプ・セット を追加します。表スペース内のいくつかのコンテナーにまたがってデータがストライピングされている場合、それらのコンテナーをまとめてストライプ・セットと呼んで、同じ表スペース内の他のコンテナーとは別個に扱うようにしています。表スペースにコンテナーを追加する場合、データのリバランスの処理を回避するには、そのストライプ・セットを新しく追加すればよいわけです。そうすれば、既存のストライプ・セット内の既存のコンテナーには触れずに、その新しいストライプ・セットにコンテナーを追加できます。

コンテナーを追加して、リバランスの処理を回避するには、 ALTER TABLESPACE ステートメントで BEGIN NEW STRIPE SET オプションを使用します。

例 5:

3 つのコンテナーを含んだ表スペースを作成するとします。エクステントのサイズは 10、 3 つのコンテナーのサイズはそれぞれ 30 ページ、40 ページ、 40 ページ (3 エクステント、4 エクステント、 4 エクステント) です。この場合の表スペースを図にすると、 121 ページの図 36 のようになります。

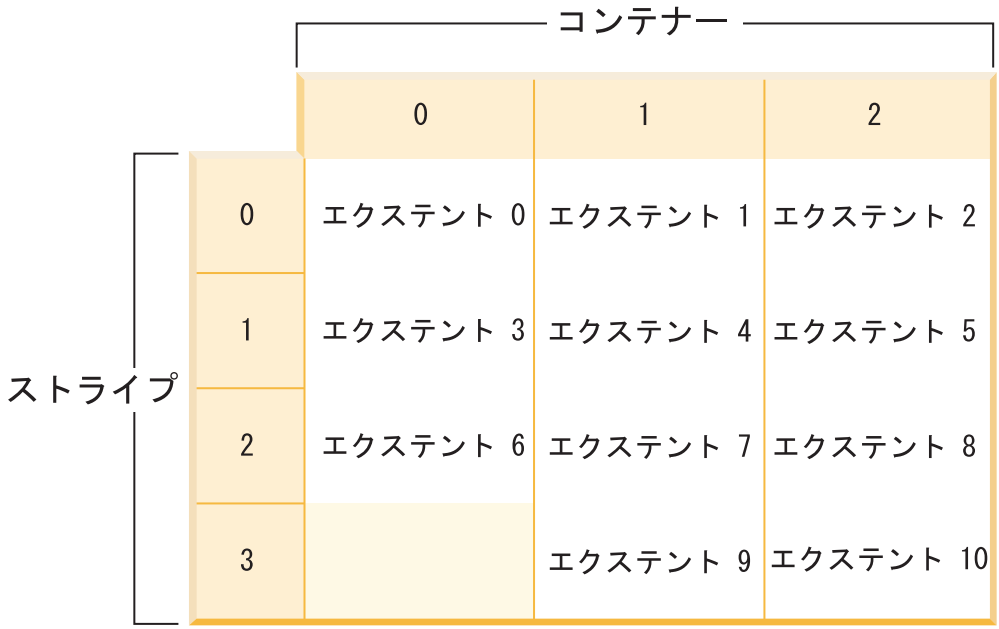

図 36. 3 つのコンテナー、11 のエクステントが含まれる表スペース

表スペースのスナップショットから分かるとおり、対応する表スペース・マップは、次のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       8       89        0        2    0   3 (0, 1, 2)
   [1]       [0]       0      10      109        3        3    0   2 (1, 2)
```

例 6:

BEGIN NEW STRIPE SET オプションを指定して、 30 ページと 40 ページ (それぞれ 3 エクステントと 4 エクステント) のコンテナーを 1 つずつ追加する場合、既存の範囲は影響を受けずに、新しい範囲セットが作成されます。この新しい範囲セットがストライプ・セットであり、最後に作成されたストライプ・セットが現在のストライプ・セットということになります。その 2 つのコンテナーを追加した後の表スペースは、 122 ページの図 37 のようになります。

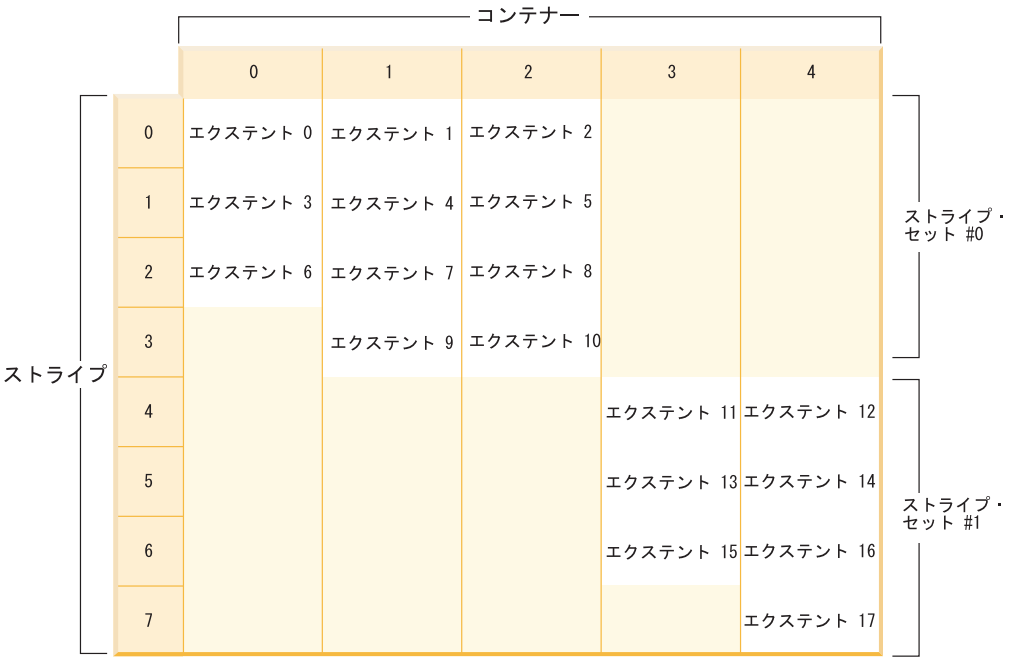

図 37. 2 つのストライプ・セットを含む表スペース

表スペースのスナップショットから分かるとおり、対応する表スペース・マップは、次のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       8       89        0        2    0   3 (0, 1, 2)
   [1]       [0]       0      10      109        3        3    0   2 (1, 2)
   [2]       [1]       4      16      169        4        6    0   2 (3, 4)
   [3]       [1]       4      17      179        7        7    0   1 (4)
```

表スペースに新しいコンテナーを追加するときに、 ADD 文節に TO STRIPE SET オプションを指定しない 場合は、コンテナーが現在のストライプ・セット (最高のストライプ・セット) に追加されます。 ADD TO STRIPE SET 文節を使用すれば、表スペース内の指定したストライプ・セットにコンテナーを追加できます。もちろん、有効なストライプ・セットを指定する必要があります。

DB2® Universal Database (DB2 UDB) は、表スペース・マップを使用してストライプ・セットをトラッキングします。したがって、新しいコンテナーを追加して、リバランスの処理を回避すると、通常は、コンテナーのリバランスの処理を実行した場合よりも、マップのサイズが早く大きくなっていきます。表スペース・マップのサイズが大きくなりすぎた場合は、コンテナーを追加しようとしたときに、エラー SQL0259N を受け取ることになります。

関連概念:

- 108 ページの『表スペース・マップ』

関連タスク:

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

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

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』
- 「コマンド・リファレンス」の『GET SNAPSHOT コマンド』
- 「システム・モニター *ガイドおよびリファレンス*」の『表スペース・アクティビティーに関するモニター・エレメント』

DMS 表スペースでコンテナーをドロップ/縮小する方法

DMS 表スペースでは、表スペースからコンテナーをドロップしたり、コンテナーのサイズを縮小することができます。そのためには、ALTER TABLESPACE ステートメントを使用します。

コンテナーのドロップや縮小が可能なのは、その操作によってドロップされるエクステントの数が、表スペースの最高水準点を超えた位置にあるフリー・エクステントの数以下である場合に限られます。要するに、その操作によってページ番号は変更できないため、最高水準点までのエクステントはすべて、表スペース内の同じ論理位置にとどまっている必要があるわけです。したがって、その操作によって生成される表スペースには、最高水準点までのデータをすべて収容するだけのスペースが必要になります。十分なフリー・スペースがない状況では、ステートメントの実行時にすぐにエラーを受け取ることになります。

この場合の最高水準点とは、表スペース内で割り振られている最高ページのページ番号です。たとえば、1000 ページの表スペースがあって、エクステントのサイズが 10 であれば、エクステントの数は 100 になります。その表スペース内で割り振られている最高エクステントが 42 番目のエクステントであれば、最高水準点は、42 * 10 = 420 ページということになります。これは、使用済みページの数と同じではありません。最高水準点よりも低い位置にあるエクステントの中には、再利用のために解放されるものもあるからです。

コンテナーのドロップや縮小を実行したときに、表スペースからドロップされるスペース内にデータが入っていると、リバランスという処理が発生します。このリバランスが始まる前に、コンテナーの変更内容に基づいて、新しい表スペース・マップが作成されます。リバランスとは、現在のマップで決められているロケーションから、新しいマップで決められているロケーションにエクステントを移す処理です。リバランスの処理は、最高水準点を含んでいるエクステントから始まり、エクステントを 1 つずつ移していき、最後にエクステント 0 を移します。エクステントを移すたびに、現在のマップが新しいマップに合わせて 1 箇所ずつ変更されていきます。現在のマップ内のエクステントのロケーションが新しいマップ内のロケーションと同じ場合は、エクステントの移動は行われず、 I/O も発生しません。この場合のリバランスは、割り振られている最高エクステントから始まり、表スペース内の最初のエクステントで終わるため、*リバース・リバランス* と呼ばれています (その逆に、コンテナーの追加または拡張の後に表スペースにスペースが追加される場合は、*フォワード・リバランス* が発生することになります)。

コンテナーをドロップすると、残りのコンテナーの番号が再割り振りされ、 0 から 1 ずつ増えていくようにコンテナー ID がきれいに調整されます。ストライプ・セ

ット内のすべてのコンテナーをドロップした場合は、そのストライプ・セットがマップから削除され、そのマップ内の残りのストライプ・セットがすべてシフトダウンして、ストライプ・セット番号の抜けがないように調整されます。

注: 以下の例では、コンテナー・サイズを決定する上でコンテナー・タグのサイズは考慮されていません。また、目的は単に例を示すことなので、かなり小さなコンテナー・サイズを使っていますが、そのサイズをお勧めするという意味ではありませんのでご注意ください。さらに、 1 つの表スペース内でいろいろなサイズのコンテナーを使っていますが、これも単に例を示すという目的なので、実際には、同じサイズのコンテナーを使用することをお勧めします。

たとえば、 3 つのコンテナーが入っている表スペースがあって、エクステントのサイズが 10 だとしましょう。 3 つのコンテナーのサイズは、それぞれ 20 ページ、50 ページ、50 ページ (2 エクステント、5 エクステント、5 エクステント) です。この表スペースの図を図 38 に示します。

ストライプ	コンテナー 0	コンテナー 1	コンテナー 2
0	エクステント 0	エクステント 1	エクステント 2
1	エクステント 3	エクステント 4	エクステント 5
2		エクステント 6	エクステント 7
3		x	x
4		x	x

図 38. 12 個のエクステントを含む表スペース (そのうち 4 個のエクステントにはデータが入っていない)

X というマークは、エクステントは存在するものの、そのエクステントにデータが入っていないという意味です。

2 つのエクステントが入っているコンテナー 0 をドロップするには、最高水準点よりも上の位置に、少なくとも 2 つのフリー・エクステントがなければなりません。この場合の最高水準点はエクステント 7 なので、 4 つのフリー・エクステントが残っています。したがって、コンテナー 0 のドロップは可能です。

表スペースのスナップショットから分かるとおり、対応する表スペース・マップは、次のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       5       59        0        1    0   3 (0, 1, 2)
   [1]       [0]       0      11      119        2        4    0   2 (1, 2)
```

ドロップ操作の後、表スペースはコンテナー 0 とコンテナー 1 だけになります。新しい表スペースの図を図 39 に示します。

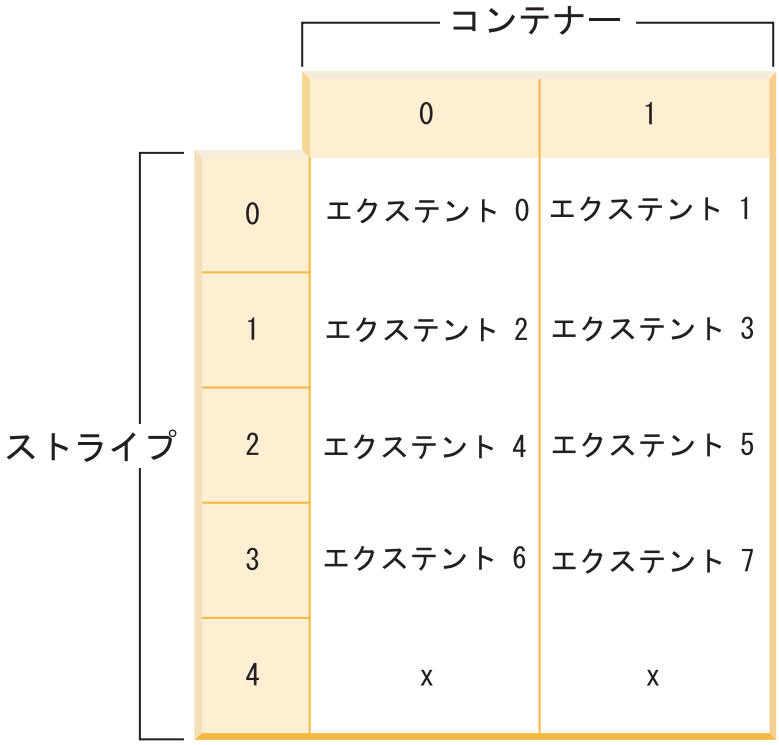

図 39. コンテナーをドロップした後の表スペース

表スペースのスナップショットから分かるとおり、対応する表スペース・マップは、次のようになります。

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]       0       9       99        0        4    0   2 (0, 1)
```

コンテナーのサイズを縮小する場合も、リバランスの動作は同じです。

コンテナーを縮小するには、 ALTER TABLESPACE ステートメントで REDUCE オプションか RESIZE オプションを使用します。コンテナーをドロップするには、ALTER TABLESPACE ステートメントで DROP オプションを使用します。

関連概念:

- 108 ページの『表スペース・マップ』

関連タスク:

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

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』
- 「*コマンド・リファレンス*」の『GET SNAPSHOT コマンド』
- 「*システム・モニター ガイドおよびリファレンス*」の『表スペース・アクティビティーに関するモニター・エレメント』

SMS 表スペースと DMS 表スペースの比較

データを格納するために使用する表スペースの種類を決定する際には、いくつかのトレードオフについて考慮する必要があります。

SMS 表スペースの利点:

- スペースが必要になる時点まで、スペースがシステムによって割り振られることはありません。
- コンテナーの事前定義が不要なため、表スペースの作成に必要な初期作業が少なくてすみます。

DMS 表スペースの利点:

- 表スペースのサイズは、 ALTER TABLESPACE ステートメントを使用してコンテナーを追加することによって増加または拡張できます。既存のデータは、最適な入出力効率を保つために、新しいコンテナーのセットにわたって自動的にリバランスすることができます。
- 格納するデータのタイプによっては、表を複数の表スペースに分割することができます。
  - ロング・フィールドおよび LOB データ
  - 索引
  - 通常表データ

  パフォーマンスを改善するため、または 1 つの表に格納できるデータの量を増やすために、表データを分割することができます。たとえば、1 つの表に通常表データ 64 GB、索引データ 64 GB、およびロング・データ 2 TB を入れることができます。 8 KB のページを使用している場合は、表データと索引データは 128 GB までとすることができます。 16 KB のページを使用している場合は 256 GB までとすることができます。 32 KB のページを使用している場合は、表データと索引データは 512 GB までとすることができます。
- ディスク上のデータのロケーションの制御は、オペレーティング・システムがこれをサポートしていれば可能です。
- すべての表データを 1 つの表スペースに入れた場合、表をドロップおよび再定義するよりも少ないオーバーヘッドで、表スペースをドロップおよび再定義することができます。
- 普通、十分に調整した DMS 表スペースの方が SMS 表スペースよりも性能が優れています。

注: Solaris™ オペレーティング環境では、パフォーマンス重視のワークロード用にはロー・デバイスを含む DMS 表スペースを使用することが強く勧められています。

普通、SMS 表スペースでは個人用の小さなデータベースを管理するのが一番簡単です。一方、サイズが大きく、これからも拡張するデータベースの場合は、 SMS 表スペースを 一時表スペースやカタログ表スペースとしてのみ使用し、 DMS 表スペースは分割して、各表に複数のコンテナーを割り当てるのが効果的です。さらに、ロング・フィールド・データおよび索引はそれぞれ独自の表スペースに保管するとよいでしょう。

装置コンテナーを使って DMS 表スペースを使用する場合は、使用環境を調整および管理する必要があります。

関連概念:

- 101 ページの『表スペースの設計』
- 104 ページの『システム管理スペース』
- 106 ページの『データベース管理スペース』

表スペースのディスク入出力

表スペースのタイプと設計によって、その表スペースに対して実行される入出力の効率が決まります。表スペースの設計と使用に関する課題をさらに考慮する前に、以下のような概念を理解しておく必要があります。

大ブロック読み取り
: 単一の要求で複数ページ (通常 1 エクステント) を検索する読み取り。一度に複数ページを読み取ることは、それぞれのページを別個に読み取るより効率的です。

プリフェッチ
: 照会でページが参照されるのに先立って行うページの読み取り。全体的な目標は、応答時間を削減することです。ページのプリフェッチを照会の実行と非同期に行うことができれば、この目標を達成することができます。最適な応答時間は、 CPU または入出力サブシステムのいずれかが最大容量で操作されている場合に達成されます。

ページ・クリーニング
: ページの読み取りおよび変更が行われると、これらのページはデータベース・バッファー・プールの中に累積されます。ページが読み込まれるときには、バッファー・プール・ページの中に読み込まれます。バッファー・プールが変更されたページでいっぱいである場合は、それらの変更されたページのいずれかをディスクに書き出さないと、新しいページを読み込むことができません。バッファー・プールがいっぱいになるのを防ぐために、ページ・クリーナー・エージェントは変更されたページを書き出して、読み取り要求の際にバッファー・プール・ページが利用できる状態にします。

DB2® Universal Database (DB2 UDB) は、大ブロック読み取りが効率的であるときには常にこれを行います。通常これは、順次または部分的に順次であるデータを検

索する場合に行われます。 1 回の読み取り操作で読み取られるデータの量はエクステント・サイズによって決まり、エクステント・サイズが大きければ大きいほど、1 回に読み取られるページ数が多くなります。

ディスクから、バッファー・プール内の連続ページにページを読み取ることができる場合、順次プリフェッチ・パフォーマンスをさらに向上させることができます。デフォルトでは、バッファー・プールはページ・ベースであるため、ディスクから連続ページに読み取るときに連続ページのセットを検出できるという保証はありません。ブロック・ベースのバッファー・プールは、ページ域だけでなく、連続ページのセット用のブロック域も含まれているため、この目的のために、ブロック・ベースのバッファー・プールを使用することができます。連続ページのそれぞれのセットには、ブロックが示されており、それぞれのブロックにはブロック・サイズとして参照されるページ数が含まれています。それぞれのブロック内のページ数だけでなく、ページ域およびブロック域のサイズが構成可能です。

エクステントがディスク上に格納される方法は、入出力の効率に影響を与えます。装置コンテナーを使用する DMS 表スペースの場合、データはディスク上で連続する傾向にあり、最小のシーク時間とディスク待ち時間で読み取ることができます。しかし、ファイルを使用する場合は、データがファイル・システムによって分割され、ディスク上の複数のロケーションに保管される可能性があります。これは、SMS 表スペースを使用し、ファイルが一度に 1 ページずつ拡張され、フラグメント化が起こりやすい場合に最もよく発生します。 DMS 表スペースで使用するために事前に割り振られた大きなファイルは、特にクリーンなファイル・スペースにファイルが割り振られた場合には、ディスク上で連続しやすくなります。

CREATE TABLESPACE または ALTER TABLESPACE ステートメント上の PREFETCHSIZE オプションを変更することによって、プリフェッチの程度を制御することができます。 (データベース内のすべての表スペースのデフォルト値は、データベース構成パラメーター *dft_prefetch_sz* によって設定されます。) PREFETCHSIZE パラメーターは、プリフェッチが起動されたときに、どれだけのページ数を読み取るかを DB2 UDB に指示します。 PREFETCHSIZE を CREATE TABLESPACE ステートメントの EXTENTSIZE パラメーターの倍数に設定すると、複数のエクステントを並列して読み取らせることができます。 (データベース内のすべての表スペースのデフォルト値は、データベース構成パラメーター *dft_extent_sz* によって設定されます。) EXTENTSIZE パラメーターは、次のコンテナーにスキップするまでに、あるコンテナーに書き込まれる 4 KB ページの数を指定します。

たとえば、3 つの装置を使用した表スペースがあるとします。 PREFETCHSIZE が EXTENTSIZE の 3 倍になるよう設定すれば、 DB2 UDB は各装置から大ブロックを並列して読み取ることができ、それによって入出力のスループットがかなり増加します。なお、ここでは、各装置が別々の物理装置であり、コントローラーが各装置からのデータ・ストリームを処理するために十分な処理速度を持っていることを想定しています。 DB2 UDB は、照会速度、バッファー・プール使用率、およびその他の要因に基づいて、ランタイムにプリフェッチ・パラメーターを動的に調整しなければならなくなる場合があることに注意してください。

一部のファイル・システム (AIX® のジャーナル・ファイル・システムなど) は、独自のプリフェッチ方式を使用します。場合によっては、 DB2 UDB プリフェッチよ

りもファイル・システムのプリフェッチの方が積極的に設定されます。この結果、ファイル・コンテナーを使用する SMS および DMS 表スペースのプリフェッチの方が、装置を使用する DMS 表スペースのプリフェッチよりも速く実行されるように見えることがあります。しかしこれは、ファイル・システム内で余分なレベルでのプリフェッチが行われているためにそう見えているに過ぎません。 DMS 表スペースは、同等のどの構成よりも速く実行できるはずです。

プリフェッチまたは均一な読み取りが効率的に行われるようにするために、十分な数のクリーン・バッファー・プール・ページが存在しなければなりません。たとえば、表スペースから 3 エクステントを読み取り、読み取られる各ページごとにバッファー・プールから変更ページを書き出さなければならない並列プリフェッチ要求が存在するとします。この並行プリフェッチ要求の効率が下がって、照会に対応できなくなる可能性があります。ページ・クリーナーは、プリフェッチ要求を満たす十分な数で構成されている必要があります。

関連概念:

- 101 ページの『表スペースの設計』
- 「*管理ガイド: パフォーマンス*」の『バッファー・プールへのデータのプリフェッチ』

関連資料:

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

表スペースの設計における処理に関する考慮事項

使用する表スペースのタイプ、および指定するページ・サイズを決定する際には、実際の環境で DB2® Universal Database (DB2 UDB) が管理している基本的な処理のタイプが影響する場合があります。オンライン・トランザクション処理 (OLTP) の処理の特徴は、データに対してランダム・アクセスを行う必要があり、多くの場合、頻繁に挿入を行ったり、または通常小さなデータ・セットを戻すアクティビティーおよび照会を更新するトランザクションです。アクセスがランダムであり、そのアクセスが 1 ないし数ページに対するものであるとすれば、プリフェッチは不可能です。

装置コンテナーを使用する DMS 表スペースは、この状態において最適に実行されます。最高のパフォーマンスが必要なければ、ファイル・コンテナーを使用した DMS 表スペースまたは SMS 表スペースもまた、 OLTP 処理に対する適切な選択です。順次入出力が少ないか、まったくない場合、 CREATE TABLESPACE ステートメントの EXTENTSIZE および PREFETCHSIZE パラメーターの設定値は、入出力の効率にとって重要ではありません。ただし、十分なページ・クリーナー数を設定し、 *chngpgs_thresh* 構成パラメーターを使用することが重要です。

照会処理の特徴は、データに対して順次アクセスまたは部分的な順次アクセスを行う必要のある、通常大きなデータ・セットを戻すトランザクションです。複数の装置コンテナーを使用する DMS 表スペースで、しかも各コンテナーが別々のディスクにある場合には、並列プリフェッチが最も効率的に行われる可能性があります。CREATE TABLESPACE の PREFETCHSIZE パラメーターの値は、 EXTENTSIZE パラメーターの値に装置コンテナーの数を乗算した値に設定しなければなりませ

ん。これによって、DB2 UDB は、すべてのコンテナーから並列にプリフェッチすることができます。コンテナー数が変更される場合、または多少なりとも積極的にプリフェッチする必要がある場合、 ALTER TABLESPACE ステートメントを使用して、それに応じて PREFETCHSIZE 値を変更することができます。

照会の処理の代わりの方法として、ファイル・システムが独自のプリフェッチを使用している場合には、ファイルを使用することもできます。ファイルは、ファイル・コンテナーを使用した DMS タイプ、または SMS タイプのいずれかが可能です。 SMS を使用する場合、入出力並列処理を達成するために、ディレクトリー・コンテナーを別々の物理ディスクにマップする必要があることに注意してください。

処理が混在している場合には、 OLTP 処理のために単一の入出力要求をできるだけ効率的にすると同時に、照会の処理のために並列入出力の効率を最大化することが目標となります。

表スペースのページ・サイズを決定するための考慮事項は次のとおりです。

- 行のランダム読み取りおよび書き込みを実行する OLTP アプリケーションについては、不必要な行に使用するバッファー・プールのスペースが少なくなるので、通常はページ・サイズは小さい方が望ましいです。
- 一度に多くの連続した行にアクセスする意思決定支援システム (DSS) アプリケーションについては、指定された数の行を読み取るのに必要な入出力要求が減るので、通常はページ・サイズは大きい方が望ましいです。しかし、これには例外があります。行サイズが以下より小さい場合、

  ```
  pagesize / 255
  ```

  各ページには無駄なスペースが生じます (1 ページの行数は最大で 255 です)。この場合、ページ・サイズは小さい方が適切でしょう。
- ページ・サイズが大きいと、索引のレベルの数を減らすことができます。
- 大きいページは、長い行をサポートします。
- デフォルトの 4 KB ページでは、表は 500 列に制限されますが、より大きなページ・サイズ (8 KB、16 KB、32 KB) は 1012 列をサポートします。
- 表スペースに使用できる最大サイズは、表スペースのページ・サイズに比例します。

関連概念:

- 104 ページの『システム管理スペース』
- 106 ページの『データベース管理スペース』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『chngpgs_thresh - 「変更済みページ数しきい値」構成パラメーター』
- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『SQL の制限値』

エクステント・サイズ

表スペースのエクステント・サイズは、次のコンテナーに書き込まれるまでに 1 つのコンテナーに書き込まれる表データのページ数を表します。エクステント・サイズを選択するときには、次のことを考慮してください。

- 表スペースの中の表のサイズとタイプ。

  DMS 表スペースの中のスペースは、 1 つの表に対して一度に 1 エクステントずつ割り当てられます。表に入力が行われ、エクステントがいっぱいになると、新しいエクステントが割り当てられます。 DMS 表スペース・コンテナー・ストレージは事前予約済みです。つまり、コンテナーを使い切るまで新しいエクステントが割り振られます。

  SMS 表スペースの中のスペースは、 1 つの表に対して一度に 1 エクステントずつ、または一度に 1 ページずつ割り振られます。表にデータが入れられ、エクステントまたはページがいっぱいになると、ファイル・システム内のすべてのエクステントまたはページが使用されるまで、新しいエクステントまたはページが割り振られます。 SMS 表スペースを使用している場合は、複数ページ・ファイル割り振りが可能です。複数ページ・ファイル割り振りの場合、一度に 1 ページずつではなく複数のエクステントを割り振ることができます。

  注: 複数ページ・ファイル割り振りは、 `db2empfa` ユーティリティーによってデータベース内のすべての SMS 表スペースについて有効にされます。複数ページのファイル割り振りがいったん使用可能になると、それを使用不能にすることはできません。バージョン 8.2 より前の場合、データベース作成時に複数ページ・ファイル割り振りはデフォルトで無効になっていました。バージョン 8.2 においてこのデフォルトは変更されており、データベース作成時に、複数ページ・ファイル割り振りはデフォルトで有効になっています。そのほうがパフォーマンスが向上するからです。バージョン 8.2 より前の動作にしたい場合は、 DB2_NO_MPFA_FOR_NEW_DB レジストリー変数の値を ON に設定してください。

  1 つの表は、以下の別個の表オブジェクトからなります。

  – 1 つのデータ・オブジェクト。これは、正規の列データが保管される場所です。
  – 1 つの索引オブジェクト。表に定義されたすべての索引がここに保管されます。
  – 1 つのロング・フィールド・オブジェクト。表に 1 つまたは複数の LONG フィールド・データがあれば、それらの列はここに保管されます。
  – 2 つの LOB オブジェクト。表に 1 つまたは複数の LOB 列がある場合、それらの列が以下の 2 つの表オブジェクトに保管されます。
    - LOB データ用の 1 つの表オブジェクト。
    - LOB データを記述するメタデータ用の 2 番目の表オブジェクト。
  – マルチディメンション表のブロック・マップ・オブジェクト。

  それぞれの表オブジェクトは別々に保管され、それぞれが必要に応じて新しいエクステントを割り当てます。また、それぞれの DMS 表オブジェクトは、 (その

表オブジェクトに属する表スペース内のすべてのエクステントを記述する) エクステント・マップ というメタデータ・オブジェクトとも関連付けられます。エクステント・マップ用のスペースも一度に 1 エクステントずつ割り当てられます。

したがって、DMS 表スペース内のオブジェクトに対するスペースの初期割り振りは、エクステント 2 個になります。 (SMS 表スペース内のオブジェクトのためのスペースの初期割り振りは 1 ページです。) このため、1 つの DMS 表スペース内に多くの小さな表がある場合は、比較的少ないデータ量を保管するのに、比較的大きな量のスペースを割り振ることになります。このような場合には、小さなエクステント・サイズを指定する必要があります。

そうでない場合に、急速に大きくなっていく大きな表に対して小さなエクステント・サイズの DMS 表スペースを使用するなら、追加のエクステントの頻繁な割り振りに伴う不必要なオーバーヘッドが生じる可能性があります。

- 表に対するアクセスの種類。

  表へのアクセスに、大量のデータを処理する照会やトランザクションが数多く使用される場合には、パフォーマンスの点で、表からデータを事前に取り出しておくのがよいでしょう。

- エクステントの必要最小数。

  表スペース用の 5 個のエクステントが入るスペースがコンテナーの中になければ、表スペースは作成されません。

関連概念:

- 101 ページの『表スペースの設計』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』
- 「コマンド・リファレンス」の『db2empfa - 複数ページ・ファイル割り振りの使用可能化コマンド』

表スペースとバッファー・プールとの間のリレーションシップ

各表スペースは、それぞれ特定の 1 つのバッファー・プールに関連付けられます。デフォルトのバッファー・プールは IBMDEFAULTBP です。別のバッファー・プールを表スペースと関連付けるためには、そのバッファー・プールが存在して (CREATE BUFFERPOOL ステートメントで定義されて) おり、また同じページ・サイズを持っている必要があり、 (CREATE TABLESPACE ステートメントを使用して) 表スペースが作成されたときに関連が定義されます。表スペースとバッファー・プールの関連は、ALTER TABLESPACE ステートメントを使用して変更することができます。

複数のバッファー・プールを使うと、全体のパフォーマンスが向上するようにデータベースの使用するメモリーを構成することができます。たとえば、ユーザーによってランダムにアクセスされる 1 つまたは複数の大きな (使用可能なメモリーより大きい) 表が入っている表スペースの場合、データ・ページをキャッシュしても有利ではないため、バッファー・プールのサイズを制限することができます。また、オンライン・トランザクション・アプリケーション用の表スペースに対してより大

きなバッファー・プールを関連付けると、アプリケーションの使用するデータ・ページが長くキャッシュされて、応答時間が速くなる場合があります。新しいバッファー・プールを構成する場合には、注意が必要です。

注: データベースで 8 KB、16 KB、または 32 KB のページ・サイズが必要であると決定したなら、そのいずれかのページ・サイズを持つ表スペースはすべて、同じページ・サイズのバッファー・プールにマップされなければなりません。

すべてのバッファー・プールに必要なストレージが、データベースが開始されたときに、データベース・マネージャーにとって使用可能でなければなりません。DB2® Universal Database (DB2 UDB) が必要なストレージを獲得できない場合、データベース・マネージャーはデフォルト・バッファー・プール (それぞれ 4 KB、8 KB、16 KB、および 32 KB のページ・サイズ) を使用して開始し、警告を出します。

パーティション・データベース環境では、データベース内のすべてのパーティションについて、同じサイズのバッファー・プールを作成することができます。異なるパーティションにそれぞれ別のサイズのバッファー・プールを作成することもできます。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『表スペースおよびその他のストレージ構造』

関連資料:

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

表スペースとデータベース・パーティション・グループとの間のリレーションシップ

パーティション・データベース環境では、各表スペースが特定のデータベース・パーティション・グループと関連付けられます。これによって、表スペースの特性をそのデータベース・パーティション・グループ内の各パーティションに適用することができます。データベース・パーティション・グループはすでに存在している必要があり (CREATE DATABASE PARTITION GROUP ステートメントを使用して定義される)、表スペースとデータベース・パーティション・グループの関連は、CREATE TABLESPACE ステートメントを使用して表スペースが作成されるときに定義されます。

ALTER TABLESPACE ステートメントを使用して表スペースとデータベース・パーティション・グループの間の関連を変更することはできません。データベース・パーティション・グループ内の個々のパーティションに対する表スペース仕様を変更できるだけです。単一パーティション・データベース環境では、各表スペースはデフォルト・データベース・パーティション・グループと関連付けられます。表スペースを定義するときのデフォルトのデータベース・パーティション・グループは

IBMDEFAULTGROUP です。ただし、システム一時表スペースが定義されている場合は IBMTEMPGROUP が使用されます。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『表スペースおよびその他のストレージ構造』
- 91 ページの『データベース・パーティション・グループ』
- 101 ページの『表スペースの設計』

関連資料:

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

ストレージ管理ビュー

「ストレージ管理」ビューは、パーティション・データベースのストレージ状態をモニターするために使用します。「ストレージ管理」ビューは、ストレージ管理ツールのグラフィカル・インターフェースです。ストレージ管理ビューでは、データベース、データベース・パーティション・グループ、または表スペースのストレージ・スナップショットをとることができます。表スペースのスナップショットをとる場合、指定された表スペースの有効範囲内に定義された表、索引、およびコンテナーに関して、システム・カタログおよびデータベース・モニターから統計情報が収集されます。データベースまたはデータベース・パーティション・グループのスナップショットをとると、指定されたデータベースまたはデータベース・パーティション・グループ内に定義されているすべての表スペースに関して、統計情報が収集されます。データベースのスナップショットをとると、そのデータベース内のすべてのデータベース・パーティション・グループに関して、統計情報が収集されます。以下のような異なるタイプのストレージ・スナップショットがあり、ストレージの異なった面をモニターするのに役立ちます。

- スペース使用量は、表スペースのスナップショットを使用してモニターできます。
- パーティション・データベースの場合のみ: データ・スキュー (データベース分散) は、データベース・パーティション・グループのスナップショットを介した場合に最適なモニターが可能です。
- 索引のクラスター化率は、データベース・パーティション・グループ・スナップショットと表スペース・スナップショットの両方でキャプチャーすることができます。索引のクラスター化率は、索引フォルダーの詳細ビューを介して提供されます。

「ストレージ管理」ビューによって、データ・スキュー、スペース使用量、および索引クラスター化率に関して、しきい値を設定することもできます。ターゲット・オブジェクトが指定されたしきい値を超えると、「ストレージ管理」ビューの中のそのオブジェクトとその親オブジェクトのアイコンに、警告フラグまたはアラーム・フラグのマークが付けられます。

注: パーティション・データベースのデータ・スキューしきい値のみを設定できます。

関連資料:

- 135 ページの『ストレージ管理ツール用のストアード・プロシージャー』
- 135 ページの『ストレージ管理ビュー表』

ストレージ管理ツール用のストアード・プロシージャー

以下の表には、ストレージ管理ツール用に作成されるストアード・プロシージャー機能が表示されています。ストアード・プロシージャーは、データベースが作成された時に、自動的に作成されます。またそれぞれのパッケージは、必要に応じてバインドされます。

表 21. ストレージ管理ツール用のストアード・プロシージャー

完全修飾名	パラメーター	機能
SYSPROC.CREATE_STORAGEMGMT_TABLES	in_tbspace VARCHAR(128) 入力値 - 表スペース名	すべてのストレージ管理表を入力値で指定した表スペース内の固定の ″DB2TOOLS″ スキーマに作成します。
SYSPROC.DROP_STORAGEMGMT_TABLES	dropSpec SMALLINT 入力値 - 0 / 1	すべてのストレージ管理表のドロップを試みます。dropSpec=0 の場合、エラーが検出されると処理は停止します。dropSpec=1 の場合、検出されたエラーは無視して、処理を続行します。
SYSPROC.CAPTURE_STORAGEMGMT_INFO	in_rootType SMALLINT 入力値 - STMG_OBJECT_TYPE 表に指定されている有効な全ての値 in_rootSchema VARCHAR(128) 入力値 - ストレージ・スナップショット・ルート・オブジェクトのスキーマ名 in_rootName VARCHAR(128) 入力値 - ルート・オブジェクトの名前	システム・カタログおよびスナップショットのために、指定されたルート・オブジェクトのストレージ関連情報、およびその有効範囲内に定義されたストレージ・オブジェクトの収集を試みます。すべてのストレージ・オブジェクトは、STMG_OBJECT_TYPE 表に指定されています。

関連資料:

- 134 ページの『ストレージ管理ビュー』

ストレージ管理ビュー表

STMG_OBJECT_TYPE 表:

STMG_OBJECT_TYPE 表には、モニター可能なサポートされたストレージ・タイプごとに、 1 つの行が含まれています。

表 22. STMG_OBJECT_TYPE 表

列名	データ・タイプ	NULL 可能	説明
OBJ_TYPE	INTEGER	N	ストレージ・オブジェクトのタイプに対応する整数値
TYPE_NAME	VARCHAR	N	ストレージ・オブジェクト・タイプの記述名

STMG_THRESHOLD_REGISTRY 表:

STMG_THRESHOLD_REGISTRY 表には、ストレージしきい値タイプごとに 1 つの行が含まれています。使用可能になったしきい値は、ストレージ・スナップショットが取られたときに、分析プロセスによって使用されます。

表 23. STMG_THRESHOLD_REGISTRY 表

列名	データ・タイプ	NULL 可能	説明
STMG_TH_TYPE	INTEGER	N	ストレージしきい値タイプに対応する整数値
ENABLED	CHARACTER	N	Y = しきい値は使用可能 N = しきい値は使用可能でないため、ストレージ分析中に比較されない
STMG_TH_NAME	VARCHAR	Y	ストレージしきい値の記述名

STMG_CURR_THRESHOLD 表:

STMG_CURR_THRESHOLD 表には、ストレージ・オブジェクトに対して明示的に設定されたしきい値タイプごとに、 1 つの行が含まれています。

表 24. STMG_CURR_THRESHOLD 表

列名	データ・タイプ	NULL 可能	説明
STMG_TH_TYPE	INTEGER	N	ストレージしきい値タイプに対応する整数値
OBJ_TYPE	INTEGER	N	ストレージ・オブジェクトのタイプに対応する整数値
OBJ_NAME	VARCHAR	N	ストレージ・オブジェクトの名前
OBJ_SCHEMA	VARCHAR	N	ストレージ・オブジェクトのスキーマ。 スキーマがオブジェクトに適用できない場合、「-」が使用される
WARNING_THRESHOLD	SMALLINT	Y	ストレージ・オブジェクトに設定された警告しきい値の値
ALARM_THRESHOLD	SMALLINT	Y	ストレージ・オブジェクトに設定されたアラームしきい値の値

STMG_ROOT_OBJECT 表:

STMG_ROOT_OBJECT 表には、ストレージ・スナップショットごとのルート・オブジェクトに、 1 つの行が含まれています。

表 25. STMG_ROOT_OBJECT 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
OBJ_TYPE	INTEGER	N	ストレージ・オブジェクトのタイプに対応する整数値
ROOT_ID	VARCHAR	N	ルート・オブジェクトの ID

STMG_OBJECT 表:

STMG_OBJECT 表には、ストレージ・スナップショットの取得によって分析されるストレージ・オブジェクトごとに、 1 つの行があります。

表 26. STMG_OBJECT 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
ROOT_ID	CHARACTER	N	ルート・オブジェクトの ID
OBJ_TYPE	INTEGER	N	ストレージ・オブジェクトのタイプに対応する整数値
OBJ_SCHEMA	VARCHAR	N	ストレージ・オブジェクトのスキーマ。 スキーマがオブジェクトに適用できない場合、「-」が使用される
OBJ_NAME	VARCHAR	N	ストレージ・オブジェクトの名前
DBPG_NAME	VARCHAR	Y	オブジェクトが常駐するデータベース・パーティション・グループの名前。該当しない場合は NULL
TS_NAME	VARCHAR	Y	オブジェクトが常駐する表スペースの名前。該当しない場合は NULL
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID

STMG_HIST_THRESHOLD 表:

STMG_HIST_THRESHOLD 表には、ストレージ・スナップショットが取られるときに、ストレージ・オブジェクトを分析するために使用される、それぞれのしきい値ごとに 1 つの行を含みます。

表 27. STMG_HIST_THRESHOLD 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
STMG_TH_TYPE	INTEGER	N	ストレージしきい値タイプに対応する整数値
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID
WARNING_THRESHOLD	SMALLINT	Y	ストレージ・スナップショットが取られるときにストレージ・オブジェクトに設定された警告しきい値の値。
ALARM_THRESHOLD	SMALLINT	Y	ストレージ・スナップショットが取られるときにストレージ・オブジェクトに設定されたアラームしきい値の値

STMG_DATABASE 表:

STMG_DATABASE 表には、データベース・ストレージ・スナップショットのそれぞれの詳細項目ごとに、 1 つの行が含まれています。

表 28. STMG_DATABASE 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID
COMPLETE_TIMESTAMP	TIMESTAMP	Y	OBJ_ID 列によって識別されたデータベースに対して、データ・キャプチャー・プロセスが完了したときのタイム・スタンプ。
SPACE_THRESHOLD_EXCEEDED	SMALLINT	Y	データベースのストレージ・スペース使用状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム
SKEW_THRESHOLD_EXCEEDED	SMALLINT	Y	データベースのデータ分散状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム

表 28. STMG_DATABASE 表 (続き)

列名	データ・タイプ	NULL 可能	説明
CR_THRESHOLD_EXCEEDED	SMALLINT	Y	データベースの索引クラスタリング状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム

STMG_DBPGROUP 表:

STMG_DBPGROUP 表には、データベース・パーティション・グループ・ストレージ・スナップショットのそれぞれの詳細項目ごとに、 1 つの行が含まれています。

表 29. STMG_DBPGROUP 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID
COMPLETE_TIMESTAMP	TIMESTAMP	Y	OBJ_ID 列によって識別されたデータベース・パーティション・グループに対して、データ・キャプチャー・プロセスが完了したときのタイム・スタンプ。
SPACE_THRESHOLD_EXCEEDED	SMALLINT	Y	データベース・パーティション・グループのストレージ・スペース使用状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム
SKEW_THRESHOLD_EXCEEDED	SMALLINT	Y	データベース・パーティション・グループのデータ分散状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム
PARTITON_COUNT	SMALLINT	Y	データベース・パーティション・グループに含まれているパーティションの数。
TARGET_LEVEL	BIGINT	Y	データベース・パーティション・グループに含まれている、すべてのパーティションにわたる平均データ・サイズ (バイト単位)。これは、均一なデータ分散のターゲット・レベルです。

表 29. STMG_DBPGROUP 表 (続き)

列名	データ・タイプ	NULL 可能	説明
DATA_SKEW	SMALLINT	Y	すべてのパーティション間の TARGET_LEVEL からの最大データ・サイズ偏差のパーセンテージ。この値は、データ・キャプチャーおよび分析プロセス中に、STMG_CURR_THRESHOLD 表内のデータベース・パーティション・グループに設定されたデータ分散スキューと比較するために使用されます。
TOTAL_SIZE	BIGINT	Y	データベース・パーティション・グループに含まれている、すべてのパーティションにわたる合計サイズ (バイト単位)。これは、データベース・パーティション・グループに定義されたすべての表スペースの合計サイズ (ページ・サイズで乗算されたページの数) の和です。 DMS 表スペースでは、合計サイズは割り振られたサイズです。SMS 表スペースでは、合計サイズは現在表スペースによって使用されているサイズです。
DATA_SIZE	BIGINT	Y	データベース・パーティション・グループに含まれている、すべてのパーティションにわたるデータ・サイズ (バイト単位)。これは、データベース・パーティション・グループに定義されたすべての表スペースのデータ・サイズ (ページ・サイズで乗算されたデータ・ページの数) の和です。
PERCENT_USED	SMALLINT	Y	合計サイズを超えるデータ・サイズのパーセンテージ値です。この値は、データ・キャプチャーおよび分析プロセス中に、スペース使用しきい値と比較されます。 SMS 表スペースの場合、表スペースまたはその親データベース・パーティション・グループのスペース使用しきい値は、不必要なアラームを避けるために、100 に設定する必要があります。

STMG_DBPARTITION 表:

STMG_DBPARTITION 表には、データベース・パーティション・ストレージ・スナップショットのそれぞれの詳細項目ごとに、 1 つの行が含まれています。これは STMG_DBPGROUP 表と一緒に使用されることを意図しています。

表 30. STMG_DBPARTITION 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID
PARTITION_NUM	INTEGER	Y	データベース・パーティション番号。
COMPLETE_TIMESTAMP	TIMESTAMP	Y	OBJ_ID 列によって識別されたデータベース・パーティションに対して、データ・キャプチャー・プロセスが完了したときのタイム・スタンプ。
DBPG_NAME	CHARACTER	Y	データベース・パーティション・グループの名前
IN_USE	CHARACTER	Y	ストレージ・スナップショットを取ったときのパーティションの状況。SYSCAT.NODEGROUPDEF 内の IN_USE 列と同様。
HOST_NAME	VARCHAR	Y	データベース・パーティションのホスト名。
HOST_SYSTEM_SIZE	BIGINT	Y	使用不可
EST_DATA_SIZE	BIGINT	Y	データベース・パーティション・グループ有効範囲内のデータベース・パーティションの見積もりデータ・サイズ。この値は、指定されたパーティションの表パーティション・データ・サイズの合計として計算されます。

STMG_TABLESPACE 表:

STMG_TABLESPACE 表には、表スペース・ストレージ・スナップショットのそれぞれの詳細項目ごとに、 1 つの行が含まれています。

表 31. STMG_TABLESPACE 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID
COMPLETE_TIMESTAMP	TIMESTAMP	Y	OBJ_ID 列によって識別された表スペースに対して、データ・キャプチャー・プロセスが完了したときのタイム・スタンプ。

表 31. STMG_TABLESPACE 表 (続き)

列名	データ・タイプ	NULL 可能	説明
SPACE_THRESHOLD_EXCEEDED	SMALLINT	Y	表スペースのストレージ・スペース使用状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム
TYPE	CHARACTER	Y	SYSCAT.TABLESPACES に定義されているのと同じ
DATATYPE	CHARACTER	Y	SYSCAT.TABLESPACES に定義されているのと同じ
TOTAL_SIZE	BIGINT	Y	SYSCAT.TABLESPACES に定義されているのと同じ
PERCENT_USED	SMALLINT	Y	SYSCAT.TABLESPACES に定義されているのと同じ。この値は、データ・キャプチャーおよび分析プロセス中に、STMG_CURR_THRESHOLD 表内のスペース使用しきい値と比較するために使用されます。
DATA_SIZE	BIGINT	Y	SYSCAT.TABLESPACES に定義されているのと同じ
DATA_PAGE	BIGINT	Y	SYSCAT.TABLESPACES に定義されているのと同じ
EXTENT_SIZE	INTEGER	Y	SYSCAT.TABLESPACES に定義されているのと同じ
PREFETCH_SIZE	INTEGER	Y	SYSCAT.TABLESPACES に定義されているのと同じ
OVERHEAD	DOUBLE	Y	SYSCAT.TABLESPACES に定義されているのと同じ
TRANSFER_RATE	DOUBLE	Y	SYSCAT.TABLESPACES に定義されているのと同じ
BUFFERPOOL_ID	INTEGER	Y	SYSCAT.TABLESPACES に定義されているのと同じ
PAGE_SIZE	INTEGER	Y	SYSCAT.TABLESPACES に定義されているのと同じ

STMG_CONTAINER 表:

STMG_CONTAINER 表には、コンテナー・ストレージ・スナップショットのそれぞれの詳細項目ごとに、 1 つの行が含まれています。

表 32. STMG_CONTAINER 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。

表 32. STMG_CONTAINER 表 (続き)

列名	データ・タイプ	NULL 可能	説明
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID
COMPLETE_TIMESTAMP	TIMESTAMP	Y	OBJ_ID 列によって識別されたコンテナーに対して、データ・キャプチャー・プロセスが完了したときのタイム・スタンプ
TS_ID	INTEGER	Y	コンテナーが割り当てられる表スペースの整数 ID
SPACE_THRESHOLD_EXCEEDED	SMALLINT	Y	コンテナーのストレージ・スペース使用状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム
PARTITION_NUM	INTEGER	Y	コンテナーが常駐するデータベース・パーティションのパーティション番号。
TYPE	CHARACTER	Y	’P’ = パス・コンテナー、’F’ = FILE コンテナー、’D’ = ロー・デバイス・コンテナー
TOTAL_PAGE	BIGINT	Y	コンテナーに割り振られた合計ページ
USABLE_PAGES	BIGINT	Y	コンテナー内の使用可能なページの数
PERCENT_USED	SMALLINT	Y	使用不可。この値は、データ・キャプチャーおよび分析プロセス中に、STMG_CURR_THRESHOLD 表内のスペース使用しきい値と比較するために使用されます。
DATA_SIZE	BIGINT	Y	使用不可
DATA_PAGE	BIGINT	Y	使用不可

STMG_TABLE 表:

STMG_TABLE 表には、表ストレージ・スナップショットのそれぞれの詳細項目ごとに、 1 つの行が含まれています。

表 33. STMG_TABLE 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID

表 33. STMG_TABLE 表 (続き)

列名	データ・タイプ	NULL 可能	説明
COMPLETE_TIMESTAMP	TIMESTAMP	Y	OBJ_ID 列によって識別された表に対して、データ・キャプチャー・プロセスが完了したときのタイム・スタンプ。
DBPG_NAME	VARCHAR	Y	表が常駐するデータベース・パーティション・グループの名前
SKEW_THRESHOLD_EXCEEDED	SMALLINT	Y	表のデータ分散状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム
TOTAL_ROW_COUNT	BIGINT	Y	表の合計行数
AVG_ROW_COUNT	BIGINT	Y	すべての表パーティションにわたった平均行数
TARGET_LEVEL	BIGINT	Y	各パーティション上の平均データ・サイズ (バイト単位)
DATA_SKEW	SMALLINT	Y	指定された表の、すべての表パーティションにわたった TARGET_LEVEL からはずれた ROW_COUNT 値の最大パーセンテージ。この値は、データ・キャプチャーおよび分析プロセス中に、STMG_CURR_THRESHOLD 表内のデータ・スキューしきい値と比較するために使用されます。
AVG_ROW_LENGTH	BIGINT	Y	表の平均の行の長さ。この統計が収集されると、この表内のすべての列の平均列長の和になります。統計データがない場合、この値は変数列長のパーセンテージに、固定列長を加算して計算されます。
COLCOUNT	INTEGER	Y	SYSCAT.TABLES に定義されているのと同じ
ESTIMATED_SIZE	BIGINT	Y	SYSCAT.TABLES に定義されているのと同じ
NPAGES	INTEGER	Y	SYSCAT.TABLES に定義されているのと同じ
FPAGES	INTEGER	Y	SYSCAT.TABLES に定義されているのと同じ
OVERFLOW	INTEGER	Y	SYSCAT.TABLES に定義されているのと同じ
MAIN_TBSPACE	VARCHAR	Y	SYSCAT.TABLES に定義されているのと同じ
INDEX_TBSPACE	VARCHAR	Y	SYSCAT.TABLES に定義されているのと同じ
LONG_TBSPACE	VARCHAR	Y	SYSCAT.TABLES に定義されているのと同じ

STMG_TBPARTITION 表:

STMG_TBPARTITION 表には、表パーティション・ストレージ・スナップショットのそれぞれの詳細項目ごとに、 1 つの行が含まれています。

表 34. STMG_TBPARTITION 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID
PARTITION_NUM	INTEGER	N	表パーティションが常駐するデータベース・パーティションのパーティション番号
COMPLETE_TIMESTAMP	TIMESTAMP	Y	OBJ_ID 列によって識別された表パーティションに対して、データ・キャプチャー・プロセスが完了したときのタイム・スタンプ
DBPG_NAME	VARCHAR	Y	表が常駐する場所のデータベース・パーティション・グループの名前
ROWCOUNT	BIGINT	Y	この表パーティション内の行数

STMG_INDEX 表:

STMG_INDEX 表には、索引ストレージ・スナップショットのそれぞれの詳細項目ごとに、 1 つの行が含まれています。

表 35. STMG_INDEX 表

列名	データ・タイプ	NULL 可能	説明
STMG_TIMESTAMP	TIMESTAMP	N	ストレージ・スナップショットのタイム・スタンプ。これはデータ・キャプチャー・プロセスが開始された時間を示します。
OBJ_ID	VARCHAR	N	指定されたストレージ・スナップショット・タイム・スタンプの下のストレージ・オブジェクトごとのユニーク ID
COMPLETE_TIMESTAMP	TIMESTAMP	Y	OBJ_ID 列によって識別された索引に対して、データ・キャプチャー・プロセスが完了したときのタイム・スタンプ
DBPG_NAME	VARCHAR	Y	索引が常駐する場所のデータベース・パーティション・グループの名前
TB_SCHEMA	VARCHAR	Y	SYSCAT.INDEXES に定義されている TABNAME と同じ

表 35. STMG_INDEX 表 (続き)

列名	データ・タイプ	NULL 可能	説明
TB_NAME	VARCHAR	Y	SYSCAT.INDEXES に定義されている TABSCHEMA と同じ
CR_THRESHOLD_EXCEEDED	SMALLINT	Y	索引クラスタリング状態を示すフラグ。 0 = 正常作動状態 1 = しきい値超過警告 2 = しきい値超過アラーム
COLCOUNT	INTEGER	Y	SYSCAT.INDEXES に定義されているのと同じ
ESTIMATED_SIZE	BIGINT	Y	SYSCAT.INDEXES に定義されているのと同じ
NLEAF	INTEGER	Y	SYSCAT.INDEXES に定義されているのと同じ
NLEVELS	SMALLINT	Y	SYSCAT.INDEXES に定義されているのと同じ
FIRSTKEYCARD	BIGINT	Y	SYSCAT.INDEXES に定義されているのと同じ
FIRST2KEYCARD	BIGINT	Y	SYSCAT.INDEXES に定義されているのと同じ
FIRST3KEYCARD	BIGINT	Y	SYSCAT.INDEXES に定義されているのと同じ
FIRST4KEYCARD	BIGINT	Y	SYSCAT.INDEXES に定義されているのと同じ
FULLKEYCARD	BIGINT	Y	SYSCAT.INDEXES に定義されているのと同じ
CLUSTERRATIO	SMALLINT	Y	SYSCAT.INDEXES に定義されているように、この値は、データ・キャプチャーおよび分析プロセス中に、指定された索引のしきい値セットと比較するために使用されます。
CLUSTERFACTOR	BIGINT	Y	SYSCAT.INDEXES に定義されているのと同じ
SEQUENTIAL_PAGES	INTEGER	Y	SYSCAT.INDEXES に定義されているのと同じ
DENSITY	INTEGER	Y	SYSCAT.INDEXES に定義されているのと同じ

関連資料:

- 134 ページの『ストレージ管理ビュー』

一時表スペースの設計

単一の SMS 一時表スペースの定義では、大半の REGULAR 表スペースで使用されているページ・サイズと等しいページ・サイズを指定することが推奨されています。そうすれば、一般的な環境と処理に適したサイズが得られます。しかし、一時表スペースの構成や処理を変えるとよい結果が得られる場合もあります。以下の点を考慮してください。

- 一時表はたいてい、ひとまとまりに順次アクセスされます。つまり、まとまった行が挿入されたり、ひとかたまりの順次行が取り出されたりします。このため、比較的大きなページ・サイズを指定すると、特定量のデータを読み取るために必要な論理/物理ページの入出力要求が少なくなるので、一般的にパフォーマンスは向上します。ただし、平均的な一時表行サイズが 255 で除算したページ・サイズより小さい場合には、必ずしもパフォーマンスが向上するとは限りません。各ページには、ページ・サイズに関係なく最大 255 行を配置できます。たとえば、15 バイト行の一時表が必要になる照会では、 4 KB の一時表スペース・ページ・サイズを使ったほうが効率がよくなります。該当する 255 行すべてを 4 KB ページ内に入れることができるからです。 8 KB (またはそれ以上) のページ・サイズを使用すると、それぞれの一時表ページごとに少なくとも 4 KB (またはそれ以上) のバイトのスペースが無駄になります。したがって、必要な入出力要求の数は減りません。
- データベース内の REGULAR 表スペースの 50 % 以上が同じページ・サイズを使用する場合は、一時表スペースの定義で同じページ・サイズを指定するほうが得策かもしれません。そのような指定を行えば一時表スペースは、 REGULAR 表スペースの大部分または全部と同じバッファー・プール・スペースを共用できるからです。この結果、バッファー・プールのチューニングが簡単になります。
- 一時表スペースを使って表を再編成する場合、一時表スペースのページ・サイズは表のページ・サイズと一致しなければなりません。このため、異なるページ・サイズごとに定義された一時表スペースが必要です。それら個々のページ・サイズは、一時表スペースを使って再編成できる既存の表によって使用されます。

  表を、ターゲット表スペースで直接再編成すれば、一時表スペースを使わずに再編成を行うことができます。言うまでもなく、このタイプの再編成では、ターゲット表スペースに再編成プロセス用の余分のスペースが必要になります。
- 実際の作業環境のために SMS システム一時表スペースの中のシステム一時表を頼りにしている場合には、レジストリー変数 DB2_SMS_TRUNC_TMPTABLE_THRESH を使用することも考慮できます。システム一時表が不要になった過去の時点では、ファイル・サイズが 0 に切り捨てられていました。新しいシステム一時表の必要には、パフォーマンス上のコストが伴う可能性があります。このレジストリー変数を使用するなら、システム上でシステム一時表を 0 (ゼロ) でないままにすることにより、システム一時表の作成と切り捨てを繰り返すことによるパフォーマンス上のコストを避けることができます。
- ページ・サイズの異なる複数の一時表スペースが存在すれば、通常、オプティマイザーは最大のバッファー・プールを持つ一時表スペースを選択します。その場合はたいてい、一時表スペースの 1 つに十分のバッファー・プールを割り当て、他の一時表スペースには小さめのバッファー・プールを割り当てるのが賢明です。そのようなバッファー・プール割り当ては、メイン・メモリーの使用効率を

向上させるのに役立ちます。たとえば、カタログ表スペースが 4 KB ページを使用し、残りの表スペースが 8 KB ページを使用する場合、最適な一時表スペースの構成として、 1 つの 8 KB 一時表スペースに大きなバッファー・プールを指定し、 1 つの 4 KB 一時表スペースに小さなバッファー・プールを指定することが考えられます。

注: カタログ表スペースに使用できるページ・サイズは 4 KB に制限されます。その場合、データベース・マネージャーは、カタログ表を再編成できる 4 KB システムの一時表スペースが必ず存在するようにします。

- 一般に、単一ページ・サイズの一時表スペースを複数定義しても、特に利点はありません。
- 一時表スペースに関してはたいてい、DMS よりも SMS を選択するほうが優れています。その理由は、以下のとおりです。
  - DMS を使用する場合は、 SMS を使用する場合よりも、一時表の作成時に多くのオーバーヘッドがあります。
  - SMS ではディスク・スペースをオンデマンドで割り振りますが、 DMS では事前に割り振っておく必要があります。事前割り振りは問題になる場合があります。たとえば、一時表スペースに保持する一時データのストレージ要件がピーク時に非常に高く、ストレージ要件の平均はとても低いという場合があります。 DMS では、ピーク時のストレージ要件は事前割り振りしておく必要がありますが、 SMS では、オフピーク時に余分のディスク・スペースを他の目的で使用することができます。
  - データベース・マネージャーは、一時表ページをディスクに書き出さずに、メモリー内に保持しようとします。結果として、DMS を使用してもパフォーマンス上の効果はあまり期待できません。

関連概念:

- 101 ページの『表スペースの設計』
- 104 ページの『システム管理スペース』
- 148 ページの『SMS 表スペースの中の一時表』

関連資料:

- 「コマンド・リファレンス」の『REORG INDEXES/TABLE コマンド』

SMS 表スペースの中の一時表

SMS 表スペースの中の一時表は、デフォルトでは不要になっても削除されません。その代わり、サイズが 1 エクステントを超える一時表に関連するファイルは、1 エクステントに切り捨てられます。一時表が繰り返し使用される場合には、このようにすることにより、一時表を削除してから再作成するためのパフォーマンス上のコストをいくらか回避することができます。

このようにして一時表を再利用することは、 Windows® NT (ファイル・システムの呼び出しのコストが比較的高い) などの場合のように、小さいシステムで数多くの小さい一時表を処理することがワークロードに含まれるユーザーや、ディスク・ストレージが分散しているためファイル・システム操作を完了するためにネットワーク・メッセージが必要なユーザーにとって、さまざまなメリットがあります。

デフォルトでは、1 エクステントより大きい一時表が含まれるファイルは、不要になった時点で 1 エクステントに切り捨てられます。その量は、DB2_SMS_TRUNC_TMPTABLE_THRESH レジストリー変数にもっと大きい値を指定することにより、大きくすることができます。実際の処理で大きな SMS 一時表を繰り返し使用する場合、反復使用を通じてスペースを割り振ったままにしておくことが可能であれば、このレジストリー変数に指定する値を大きくしてください。

DB2_SMS_TRUNC_TMPTABLE_THRESH レジストリー変数に値として 0 を指定すると、この機能はオフになります。システムのスペースがかなり限られていて、SMS 一時表スペースに関連してディスク不足エラーが何度も発生するという場合には、オフにするとよいでしょう。

データベースに最初に接続すると、それまでに割り振られていたファイルはすべて削除されます。既存の一時表を消去してしまいたい場合は、すべてのデータベース接続をドロップしてから再接続するか、またはデータベースを非アクティブにしてから再びアクティブにしてください。一時表のスペースを割り振ったままにするには、 ACTIVATE DATABASE コマンドを使用してデータベースを起動してください。それによって、データベースへの最初の接続において始動処理のコストの繰り返しを避けることができます。

関連概念:

- 147 ページの『一時表スペースの設計』

カタログ表スペースの設計

以下のような理由のために、データベース・カタログには SMS 表スペースを使用することが推奨されています。

- データベース・カタログは、サイズが異なる多くの表からなります。 DMS 表スペースを使用している場合、各表オブジェクトについて、最低 2 つのエクステントが割り当てられます。選択されるエクステント・サイズによっては、かなりの量が割り当てられ、その結果、未使用のスペースができてしまいます。 DMS 表スペースを使用する場合は小さなエクステント・サイズ (2 から 4 ページ) を選択し、そうでない場合は SMS 表スペースを使用してください。
- カタログ表の中にラージ・オブジェクト (LOB) 列があります。 LOB データは他のデータと一緒にバッファー・プールの中には保持されず、必要になるたびにディスクから読み取られます。ディスクから LOB を読み取るとパフォーマンスが低下します。ファイル・システムには通常、独自のキャッシュがあるため、 SMS 表スペースまたはファイル・コンテナー上に作成された DMS 表スペースを使用すれば、 LOB が以前に参照された場合に発生する入出力を回避することができます。

これらのことを考慮すれば、SMS 表スペースのほうが、カタログ用には望ましい選択です。

考慮する必要のある別の要素は、将来カタログ表スペースを拡張する必要があるかどうかということです。一部のプラットフォームは SMS コンテナー用の基礎となるストレージの拡張をサポートしており、リダイレクトされたリストアを使って

SMS 表スペースを拡張することも可能ですが、 DMS 表スペースを使用すれば新しいコンテナーを容易に追加することができます。

関連概念:

- 「*管理ガイド: インプリメンテーション*」の『システム・カタログ表の定義』
- 101 ページの『表スペースの設計』
- 104 ページの『システム管理スペース』
- 106 ページの『データベース管理スペース』

データが RAID 装置にある場合の表スペース・パフォーマンスの最適化

このセクションでは、データが Redundant Array of Independent Disks (RAID) 装置に置かれている場合に、パフォーマンスを最適化する方法について説明します。

手順:

RAID 装置を使用している表スペースごとに以下のことを行う必要があります。

- 表スペース (RAID 装置を使用している) ごとに 1 つのコンテナーを定義する。
- 表スペースの EXTENTSIZE を RAID ストライプ・サイズと同じか、その整数倍にする。
- 表スペースの PREFETCHSIZE を以下のようにする。
  - RAID ストライプ・サイズに RAID 並列装置の数を乗算する (または、この積の整数倍にする)。さらに、
  - EXTENTSIZE の整数倍にする。
- DB2_PARALLEL_IO レジストリー変数を使って、表スペースの並列入出力を使用可能にする。

DB2_PARALLEL_IO:

DB2 Universal Database™ (DB2 UDB) は表スペース・コンテナーでのデータの読み取りまたは書き込みを行う際に、データベース内のコンテナー数が複数であれば、並列入出力を使用することがあります。しかし、単一のコンテナー表スペースで並列入出力を実行するほうがよいと判断できる状況もあります。たとえば、複数の物理ディスクから成る単一の RAID 装置にコンテナーが作成される場合、並列読み取りおよび並列書き込みの呼び出しを発行することができます。

単一コンテナーを持つ表スペースの並列入出力を強制するには、DB2_PARALLEL_IO レジストリー変数を使用します。この変数は、各表スペースを意味する "*" (アスタリスク) に設定することも、コンマで区切った表スペース ID のリストに設定することもできます。たとえば、以下のとおりです。

```
db2set DB2_PARALLEL_IO=*        {turn parallel I/O on for all table spaces}
db2set DB2_PARALLEL_IO=1,2,4,8  {turn parallel I/O on for table spaces 1, 2,
                                 4, and 8}
```

レジストリー変数を設定したら、変数を有効にするために、 DB2 UDB を停止 (**db2stop**) した後、再始動 (**db2start**) する必要があります。

DB2_PARALLEL_IO は、複数のコンテナーが定義されている表スペースにも影響を与えます。レジストリー変数を設定しない場合、入出力並列処理は、表スペース内

のコンテナー数と等しくなります。レジストリー変数を設定する場合、入出力並列処理は、エクステント・サイズで割ったプリフェッチ・サイズの結果と等しくなります。表スペース内の個々のコンテナーが複数の物理ディスクを介してストライピングされる場合に、レジストリー変数を設定したい場合もあるでしょう。

たとえば、表スペースには 2 つのコンテナーがあり、プリフェッチ・サイズはエクステント・サイズの 4 倍です。レジストリー変数が設定されない場合、この表スペースに対するプリフェッチ要求は 2 つの要求に分けられます (それぞれの要求は 2 つのエクステントに対して行われます)。プリフェチャーが処理を行うことができる場合、2 つのプリフェチャーは、並列して 2 つの要求を処理することができます。レジストリー変数が設定される場合、4 つのプリフェチャーが並列して要求を出すことが可能であれば、この表スペースに対するプリフェッチ要求は 4 つの要求 (要求ごとに 1 つのエクステント) に分けられます。

この例では、2 つのコンテナーのそれぞれが専用の単一ディスクを持った場合、この表スペースに対するレジストリー変数を設定すると、 2 つのプリフェチャーが 2 つのディスクのそれぞれに同時にアクセスするので、それらのディスクにおける競合が生じます。ただし、2 つのコンテナーのそれぞれが複数のディスクを介してストライピングされた場合、レジストリー変数を設定すると、すぐに 4 つの異なるディスクへのアクセスが許可される可能性があります。

DB2_USE_PAGE_CONTAINER_TAG:

デフォルトでは、DB2 UDB は、それぞれの DMS コンテナー (ファイルまたは装置) の最初のエクステントを使用して、コンテナー・タグを保管します。コンテナー・タグは、コンテナー用の DB2 UDB のメタデータです。 DB2 UDB の以前のバージョンでは、最初のページは、最初のエクステントではなくコンテナー・タグのために使用されたので、タグを保管するために使用されたコンテナー内のスペースは少なくて済みました。 (DB2 UDB の以前のバージョンでは、DB2_STRIPED_CONTAINERS レジストリー変数は、エクステント・サイズのタグを使用して表スペースを作成するために使用されました。ただし、これは現在デフォルトの動作であるため、このレジストリー変数はいかなる効力もありません。)

DB2_USE_PAGE_CONTAINER_TAG レジストリー変数が ON に設定されると、作成された新規 DMS コンテナーは、 1 エクステントのタグ (デフォルト) ではなく、1 ページのタグを使用して作成されます。レジストリー変数を設定する前に作成された既存のコンテナーへの影響はありません。

非常に厳しいスペース制約があるか、またはバージョン 8 以前のデータベースと動作が一貫している必要があるのでない限り、このレジストリー変数を ON に設定することは推奨されていません。

表スペース・コンテナー用の RAID 装置が使用される場合、このレジストリー変数を ON に設定すると、入出力のパフォーマンスに不利な影響を与えます。表スペース・コンテナー用に RAID 装置を使用する場合、表スペースの作成では、RAID ストライプ・サイズと同じか、その整数倍のエクステント・サイズを指定することが提案されています。ただし、このレジストリー変数を ON に設定する場合、 1 ページのコンテナー・タグが使用されるため、エクステントは RAID ストライプと同列にはなりません。その結果、入出力要求中に最適と思える数よりも多くの物理デ

ィスクにアクセスする必要があるかもしれません。したがって、このレジストリー変数を設定しないように強く勧められています。

1 ページのコンテナー・タグを使用してコンテナーを作成するには、以下のように、このレジストリー変数を ON に設定してから、インスタンスを停止し、再始動してください。

```
db2set DB2_USE_PAGE_CONTAINER_TAG=ON
db2stop
db2start
```

1 ページのコンテナー・タグを使用してコンテナーの作成を停止するには、このレジストリー変数をリセットしてから、インスタンスを停止し、再始動してください。

```
db2set DB2_USE_PAGE_CONTAINER_TAG=
db2stop
db2start
```

コントロール・センター、LIST TABLESPACE CONTAINERS コマンド、および GET SNAPSHOT FOR TABLESPACES コマンドは、コンテナーが 1 ページのタグまたはエクステント・サイズのタグを使用して作成されたかどうかを表示しません。コンテナーが作成された方法にしたがって、「ファイル」または「装置」というラベルを使用します。コンテナーが 1 ページのタグまたはエクステント・サイズのタグを使用して作成されたかどうかを検査するには、 DB2DART の /DTSF オプションを使って表スペースとコンテナー情報をダンプし、それからタイプ・フィールドを見て問題のコンテナーを調べます。照会コンテナー API (sqlbftcq および sqlbtcq) を使用して、タイプを表示する簡単なアプリケーションを作成することもできます。

関連概念:

- 101 ページの『表スペースの設計』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『システム環境変数』

表の表スペースを選択する際の考慮事項

表を表スペースにマッピングする方法を決定するときは、次の点を考慮してください。

- 表のパーティション化。

  最低でも、選択する表スペースが、希望するパーティション化を使用するデータベース・パーティション・グループ内にあるようにする必要があります。

- 表内のデータの量。

  1 つの表スペースの中に多くの小さな表を保管する計画である場合、その表スペースに対して SMS を使用することを検討してください。入出力とスペース管理を効率的に行う DMS の利点は、小さな表ではそれほど重要ではありません。スペースを一度に 1 ページずつ、しかも必要なときだけ割り当てるという SMS の利点の方が、小さな表の場合はより魅力的です。表の 1 つがより大きいか、また

は表内のデータにより速くアクセスする必要がある場合には、小さなエクステント・サイズを持つ DMS 表スペースを検討してください。

非常に大きな表の場合は、それぞれに別個の表スペースを使用し、小さな表はすべて 1 つの表スペースにまとめるのが良いでしょう。このように分けると、表スペースの使用方法に基づいて適切なエクステント・サイズを選択することもできます。

- 表の中のデータのタイプ。

  たとえば、あまり頻繁に使用されない履歴データの入った表がある場合、このデータに対する照会については、応答時間が長くてもよいとエンド・ユーザーは考えるかもしれません。その場合、履歴データ表に別の表スペースを使用して、その表スペースをアクセス速度が遅く、費用のかからない物理装置に割り当てるのも一案です。

  あるいは、データが快適に使用できなければならなかったり迅速な応答が求められるいくつかの重要な表は、別扱いにするという方法もあります。そのような表は、そうした重要なデータ要件をサポートできる高速の物理装置に割り当てられた表スペースに入れておきます。

  また、DMS 表スペースを使用して、表データを 3 つの異なる表スペースに分けることもできます。つまり 1 つは索引データ用、1 つは LOB およびロング・フィールド・データ用、残る 1 つは通常表データ用です。これにより、データに最適な表スペース特性、およびそれらの表スペースをサポートする物理装置を選択することができます。たとえば、利用可能な最高速の装置に索引データを入れると、パフォーマンスはかなり向上します。複数の DMS 表スペースにまたがって 1 つの表を分割する場合、ロールフォワード・リカバリーが使用可能であれば、これらの表スペースをまとめてバックアップおよびリストアすることを考慮してください。 SMS 表スペースは、複数の表スペースにまたがるこのようなタイプのデータ分散をサポートしません。

- 管理上の問題。

  一部の管理機能は、データベースや表のレベルではなく、表スペースのレベルで実行できます。たとえば、データベースではなく表スペースのバックアップをとれば、時間とリソースの節約になります。こうすれば、大量の変更がある表スペースを頻繁にバックアップする一方、変更が非常に少ない表スペースは時折バックアップするだけにできます。

  データベースや表スペースはリストアすることができます。互いに無関係な表が表スペースを共有していない場合、データベースのごく一部だけをリストアして、コストを減らすことができます。

  互いに関連する表は一まとまりの表スペースに一緒に入れておくのがよいでしょう。そうした表は参照制約によって関連付けられる場合もあれば、定義された他の業務制約によって関連付けられる場合もあります。

  ある特定の表を頻繁にドロップおよび再定義する必要がある場合、表をドロップするよりは DMS 表スペースをドロップする方がより効率的であるため、その表を独自の表スペースの中に定義したほうがよい場合があります。

関連概念:

- 91 ページの『データベース・パーティション・グループ』
- 104 ページの『システム管理スペース』
- 106 ページの『データベース管理スペース』
- 126 ページの『SMS 表スペースと DMS 表スペースの比較』

DB2 UDB で使用される表

DB2® Universal Database (DB2 UDB) には、以下のタイプの表があります。

- 通常表。ヒープとしてインプリメントされます。
- アペンド・モードの表。通常表を、主に INSERT 用に最適化したものです。
- マルチディメンション・クラスタリング (MDC) 表。複数のキーまたはディメンション で同時に物理的なクラスタリングが行われる表としてインプリメントされます。
- レンジ・クラスター表 (RCT)。データの順次クラスターとしてインプリメントされ、高速かつ直接のアクセスを可能にします。

表の各タイプには、特定のビジネス環境で作業する際に有効な特性があります。使用する表ごとに、実際の必要にはどの表タイプが最適かを考慮してください。

索引付き通常表は、一般的に使用される表です。

通常表は、ALTER TABLE ステートメントを使用してアペンド・モードになります。アペンド・モードの表は、銀行で顧客の口座を扱うときのように、新規データの追加や既存データの検索を行う必要がある場合に適しています。この表では、引き出し、預け入れ、振り込みによる口座の各変更が記録されます。また、顧客が口座への変更の履歴を確認することを望む場合もあります。

マルチディメンション・クラスタリング表は、データウェアハウジングや大規模データベース環境で使用されます。通常表における索引のクラスタリングは、データの 1 ディメンションだけのクラスタリングをサポートします。 MDC 表では、複数のディメンション間でのデータ・クラスタリングを利用できます。

レンジ・クラスター表は、表内の 1 つまたは複数の列でデータが密にクラスタリングされている場合に使用できます。この表では、列内の最大値と最小値によって、有効な値の範囲が定義されます。表内のレコードへは、これらの列を使用してアクセスします。

関連概念:

- 159 ページの『マルチディメンション・クラスタリング表』
- 155 ページの『レンジ・クラスター表』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『表の作成とデータの読み込み』
- 「*管理ガイド: インプリメンテーション*」の『マテリアライズ照会表の作成』

レンジ・クラスター表

レンジ・クラスター表 (RCT) は、表の中でレコードを探索するために内部で使用される ID であるレコード ID (RID) が事前定義されている、表のレイアウト体系です。

データを保持する各表について、使用できるどの表タイプが実際の必要に対して最適かを考慮してください。たとえば、緩くクラスタリングされている (単調増加しない) データ・レコードがある場合は、通常表と索引を使用することを考慮してください。キー内に重複する (ユニークでない) 値を持つデータ・レコードがある場合は、レンジ・クラスター表を使用しないでください。使用したいレンジ・クラスター表に一定量のディスク・ストレージを事前割り当てする余裕がない場合は、このタイプの表は使用しないでください。これらの要素は、レンジ・クラスター表として使用できるデータがあるかどうかを判断するのに役立ちます。

レコードのキー値を表内の特定の行のロケーションと等価にするためのアルゴリズムが使用されます。基本のアルゴリズムは、かなりシンプルなものです。最も基本的な形 (複数列を使用せず、単一の列のみでキーが構成されている) では、アルゴリズムは、論理行番号にシーケンス番号をマップします。また、アルゴリズムは、レコードのキーを使用して、論理ページ番号とスロット番号を判別します。このプロセスは、レコード、つまり表内の特定の行へのアクセスをかなり高速なものにします。

アルゴリズムには、ハッシュは関係していません。ハッシュは、キー値の配列を保持しないからです。キー値の配列の保持は、過去の表データを再編成する必要を省いてくれるので、不可欠です。

表の各レコード・キーには、以下の特性が必要です。

- ユニーク
- 非ヌル
- 整数 (SMALLINT、INTEGER、または BIGINT)
- 単調に増加する
- 事前に決定された、キーの各列に基づく一連の範囲内にある

  ALLOW OVERFLOW オプションは、キー値が定義されている範囲を超える表を作成するのに使用できます。 DISALLOW OVERFLOW オプションは、キー値が定義されている範囲を超えない表を作成するのに使用できます。この場合、範囲によって示されている境界の外にレコードが挿入されると、 SQL エラー・メッセージが戻されます。

シーケンス・キー範囲がきつくクラスタリングされた (高密度な) アプリケーションは、レンジ・クラスター表に最適な候補といえます。このタイプのキーを使用してレンジ・クラスター表を作成する場合、表の行の論理ロケーションを生成するためにキーが使用されます。このプロセスにより、別個の索引が必要でなくなります。

レンジ・クラスター表の構造に関連した利点には、次のような要素が含まれます。

- 直接アクセス

アクセスは、レンジ・クラスター表のキー/RID マッピング機能を通して行われます。

- 保守が少ない

  B+ ツリーなどの 2 次構造は、それぞれの INSERT、UPDATE、または DELETE ごとに更新を行う必要がありません。

- ロギングが少ない

  同じサイズの通常表や関連する B+ ツリー索引と比較して、レンジ・クラスター表では、ロギングの実行が少なくなっています。

- 必要なバッファー・プール・メモリーが少ない

  2 次の構造の保管に、追加のメモリーが必要ありません。

- B+ ツリー表の配列プロパティー

  追加のレベルや B+ ツリーの次のキーのロッキング体系を使用しなくても、 B+ ツリー表に同じ配列のレコードが見つかります。 RCT では、通常の B+ ツリー索引に比べて、コード・パスの長さが短くなっています。ただし、この利点を活用するためには、DISALLOW OVERFLOW でレンジ・クラスター表を作成し、(まばらではなく) 高密度にデータを入れる必要があります。

- 1 つのより小さな索引

  各キーをディスク上のロケーションにマッピングすることによって、今まで別に必要だった索引が不要になり、より小さな 1 つの索引を使用して表を作成できるようになりました。レンジ・クラスター表では、表内のデータにアクセスするためのアプリケーション要件によっては、 2 番目の別の索引が必要になる場合があります。アプリケーションが必要とすれば、通常の索引を作成することは可能です。

索引は、以下の機能を実行するために使用されます。

- レコードのキーに基づいてレコードを探し出す
- 開始と停止のキー・スキャンを適用する
- 縦方向にパーティション化する

  RCT を使用する場合、考慮に入れられない索引の唯一のプロパティーは縦方向のパーティション化です。

レンジ・クラスター表を使用することにしたなら、通常表との違いである以下のさまざまな特性を考慮してください。

- レンジ・クラスター表には、フリー・スペース制御レコード (FSCR) がありません。
- スペースが事前割り振りされます。

  表のレコードがいっぱいになっていない場合であっても、表で使用するための表のスペースが事前割り振りおよび予約されています。表の作成時、表には何もレコードが入っていませんが、ページの範囲全体は事前割り振りされています。事前割り振りは、レコードのサイズと、保管されるレコードの最大数に基づいて決定されます。

- 各レコードで VARCHAR のような可変長フィールドが使用されるときは、フィールドの最大長が使用され、すべてのレコード・サイズが固定長になります。各レコードの固定長は、すべて、レコードの最大数と合わせて、必要なスペースを決定するために使用されます。
- そのため、効率的に利用できない余分のスペースが割り振られることになる場合があります。
- キー値がまばらである場合は、未使用のスペースが存在することになり、範囲スキャンのパフォーマンスが低くなります。
- 範囲スキャンでは、それらのキー値を含む行がまだデータベースに挿入されていない場合であっても、範囲内のレコードのうち可能性のあるあらゆるレコードをスキャンする必要があります。

• スキーマの変更はできません。

  レンジ・クラスター表に対してスキーマの変更が必要になった場合は、表を再作成して、新しいスキーマ名と以前の表のすべてのデータをその表に含める必要があります。特に、

  - キー範囲の変更はサポートされていません。

    この点は重要です。というのは、表の範囲を変更することが必要な場合には、目的の範囲の新しい表を作成しなければならず、新しい表に以前の表のデータを入れなければならないからです。

• 重複キー値は許されません。
• 定義されている範囲に含まれないキー値は許されません。

  これは、 DISALLOW OVERFLOW として定義されているレンジ・クラスター表についてのみ当てはまります。

  - NULL 値は明示的に禁止されています。

• レンジ・クラスター索引は物理的には作成されません。

  RCT キー・プロパティーがある索引は、システム・カタログに示され、オプティマイザーによって選択できますが、索引はディスク上に作成されません。通常表では、表に関連付けられた各索引に、スペースも与える必要があります。 RCT では、RCT 索引にスペースは必要ありません。オプティマイザーが、この RCT 索引を参照するシステム・カタログの情報を使用して、表への正しいアクセス方式が確実に選択されるようにします。

• レンジ・クラスター表と同じ定義に対して、主キーまたはユニーク・キーを作成することは、冗長であるため許されていません。
• レンジ・クラスター表は、オリジナルのキー値の配列を保存します。キー値の配列は、表内の行のクラスタリングを保証する重要な機構です。

これらの考慮事項に加えて、レンジ・クラスター表の使用できる場所が限定されるか、さもなければ他のユーティリティーがそれらの表についてうまく動作しなくなる非互換性がいくつかあります。レンジ・クラスター表に関する制限には、次のことがあります。

• 宣言済み一時表 (DGTT) はサポートされません。

  それらの一時表では、レンジ・クラスター・プロパティーを使用できません。

- 自動サマリー表 (AST) はサポートされません。

  それらの表では、レンジ・クラスター・プロパティーを使用できません。
- ロード・ユーティリティーはサポートされません。

  行は、インポート操作または並列挿入アプリケーションにより、一度に 1 個ずつ挿入する必要があります。
- REORG TABLE ユーティリティーはサポートされません。

  DISALLOW OVERFLOW として定義されているレンジ・クラスター表を再編成する必要はありません。それでも、ALLOW OVERFLOW として定義されているレンジ・クラスター表で、そのオーバーフロー領域内のデータを再編成することは許されません。
- レンジ・クラスター表は 1 個の論理マシンにおいてのみ可能です。

  データベース・パーティション化機能 (Database Partitioning Feature (DPF)) 付きの Enterprise Server Edition (ESE) の場合、レンジ・クラスター表が、複数のデータベース・パーティションを含むデータベース・パーティション・グループ内に存在することは許されません。これは、複数のパーティションを含むデータベース・パーティション・グループ内でのレンジ・クラスター表の作成を禁止することにより禁止されています。さらに、その表スペースの 1 つに含まれるレンジ・クラスター表を含むデータベース・パーティション・グループの再配分は許されません。
- 設計アドバイザーでは、レンジ・クラスター表は勧められていません。
- レンジ・クラスター表は、定義により既にクラスター化されています。

  したがって、次のクラスタリング・スキームは、レンジ・クラスター表と互換性がありません。
  - マルチディメンション・クラスタリング (MDC) 表
  - クラスター索引
- 値とデフォルトの圧縮はサポートされません。
- レンジ・クラスター表に対するリバース・スキャンはサポートされません。
- IMPORT コマンドの REPLACE オプションはサポートされません。
- ALTER TABLE ... ACTIVATE NOT LOGGED INITIALLY ステートメントの WITH EMPTY TABLE オプションは、サポートされません。

関連概念:

- 158 ページの『レンジ・クラスター表と範囲外のレコード・キー値』
- 「*管理ガイド: インプリメンテーション*」の『範囲クラスター表の例』

レンジ・クラスター表と範囲外のレコード・キー値

レコードのオーバーフローを可能にするレンジ・クラスター表 (RCT) の振る舞いは、 CREATE TABLE ステートメントと ALLOW OVERFLOW オプションを使用して制御します。この方法により、確実に、定義された範囲内で表に必要なすべてのページが即時に割り振られます。

一度作成されると、定義された範囲内に置かれたすべてのキー付きレコードは、表の作成時にオーバーフロー・オプションが有効になっていたかどうかに関係なく、同じ働きをします。違いが生じるのは、定義された範囲の外に置かれたキー付きレコードがある場合です。この場合、表でオーバーフローが有効になっていると、レコードは、動的に割り振られてオーバーフロー域の中に置かれます。定義された範囲の外からさらにレコードが追加されると、これらはオーバーフロー域に置かれ、オーバーフロー域は大きくなっていきます。すると、このオーバーフロー域に関係する、表に対するアクションが行われたとき、アクションにはオーバーフロー域へのアクセスが含まれるため、アクションの処理時間が長くかかるようになります。オーバーフロー域へのアクセスにかかる時間は、オーバーフロー域が大きくなるほど長くなります。ある程度長い期間オーバーフロー域を使用した後、新しい拡張された範囲を使用して定義した新しいレンジ・クラスター表に既存の表のデータをエクスポートして、既存の表のサイズを小さくすることを考慮してください。

挿入したいレコード・キー値がレンジ・クラスター表で定義された範囲外である場合があるかもしれません。このタイプの RCT を存在させるためには、 CREATE TABLE ステートメントで DISALLOW OVERFLOW オプションを使用する必要があります。このタイプの RCT を作成したときは、レコード・キー値が許可または定義された範囲の外に置かれることを伝えるエラー・メッセージを受け入れる必要があります。

関連資料:

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

レンジ・クラスター表のロック

通常の処理において、レコードのロックは常に 1 つのアプリケーションやユーザーしか、レコードまたはレコード・グループへアクセスできないようにする場合に行われます。レンジ・クラスター表では、キーのロックや Next Key ロックの代わりに、「離散ロック」が使用されます。この方式では、アプリケーションやユーザーが要求した操作によって影響を受ける (もしくは、影響を受ける可能性のある) すべてのレコードをロックします。行われるロックの数は、分離レベルに依存します。

現行で空になっていても事前割り振りされている、レンジ・クラスター表の限定行はロックされます。これにより、次のキーのロックは必要なくなります。結果として、高密度のレンジ・クラスター表に必要なロックは少なくなります。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『ロックおよび並行性の制御』

マルチディメンション・クラスタリング表

マルチディメンション・クラスタリング (MDC) は、マルチディメンションの表のデータ・クラスタリングを柔軟、連続的、かつ自動的に実行する優れた方式です。MDC によって、照会のパフォーマンスが大きく改善され、さらに再編成や、挿入、更新、削除中の索引保守操作などデータ保守操作のオーバーヘッドが大きく削減さ

れます。 MDC の主な目的は、データウェアハウジングおよび大規模データベース環境での使用です。そのほか、オンライン・トランザクション処理 (OLTP) 環境でもこれを使用することができます。

通常表と MDC 表の比較:

通常表には、レコードに基づく索引があります。それらの索引のクラスタリングは、単一のディメンションに制限されています。 DB2® Universal Database (DB2 UDB) のバージョン 8 より前は、クラスター索引を介した単一ディメンションのデータ・クラスタリングだけがサポートされていました。表でレコードが挿入および更新されるとき、 DB2 UDB はクラスター索引を使用して、データの物理的順序を索引のキー順序のページで保守します。クラスター索引は、述部にクラスター索引キーが (1 つまたは複数) 含まれるような照会範囲のパフォーマンスを大きく改善します。優れたクラスター索引により、表の一部分だけにアクセスするだけで済み、プリフェッチをさらに効率的に実行できるため、パフォーマンスが向上します。

しかし、クラスター索引を使用したデータ・クラスタリングには、いくつかの欠点もあります。第 1 の点として、時間が経過するにつれてスペースがデータ・ページで埋まるため、クラスタリングが保証されません。挿入操作においてレコードが追加されるページは、そのレコードと同一または類似のクラスター化キー値のレコードの近くですが、理想的な場所にスペースがないなら、それは表の中のどこか別の場所に挿入されることになります。したがって、表をクラスター化し直し、将来のクラスター化挿入要求に備えてフリー・スペースを余分に設けるようページをセットアップするために、定期的な表の再編成が必要になります。

第 2 の点として、データのクラスター化は 1 ディメンションに関してだけなので、「クラスター」索引として指定できるのは 1 個の索引だけであり、他のすべての索引は非クラスター索引になります。この制限は、バージョン 8.1 より前の索引がすべてそうであるようにクラスター索引がレコード・ベースであるということに関連しています。

第 3 の点として、レコード・ベースの索引の場合、表のあらゆる単一のレコードについてポインターが含まれているため、サイズが巨大になる場合があります。

クラスター化索引

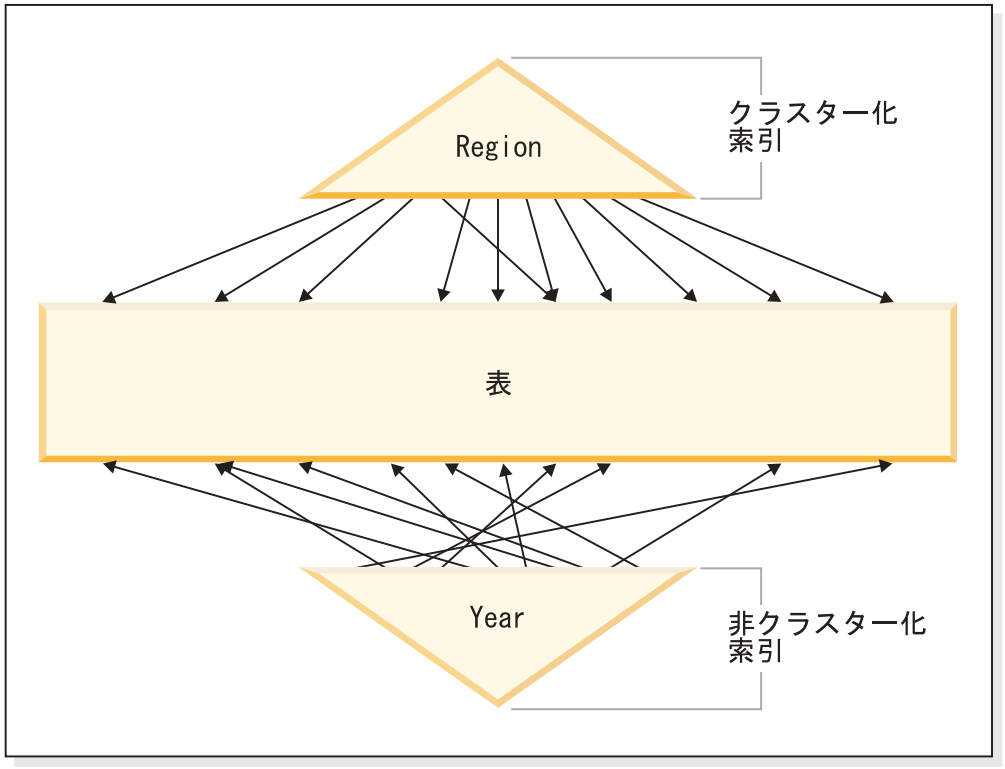

図 40. クラスター索引を伴う通常表

図 40 の表には、2 個のレコード・ベースの索引があります。

- “Region” 上のクラスター索引
- “Year” 上の別の索引

“Region” 索引は、クラスター索引です。したがって、その索引の中でキーをスキャンする際、対応するレコードは表の中の同じページか、その近くのページにほとんどが存在しているはずです。一方、“Year” 索引は非クラスター索引なので、その索引の中でキーをスキャンする際、表全体の中で対応するレコードが存在する位置はランダムです。クラスター索引のスキャンのほうが入出力パフォーマンスが高く、その索引に対してデータのクラスター化の程度が高いほど順次プリフェッチのメリットが大きくなります。

MDC には、ブロック・ベースの索引が導入されています。「ブロック索引」は、個々のレコードではなくレコードのブロック (グループ) へのポインターです。クラスター化値に従って MDC 表のデータを物理的に複数のブロックに編成し、ブロック索引を使用してそれらのブロックにアクセスすることによって、 MDC はクラスター索引の欠点を解決するだけでなく、さらにパフォーマンスが大きく改善されます。

第 1 の点として、MDC では、 1 つの表を同時に複数のキー (つまりディメンション) に基づいて物理的にクラスター化できます。 MDC では、単一ディメンションクラスタリングの利点が複数ディメンション、またはクラスター化キーにまで拡大されます。表のうち指定された 1 つ以上のディメンションのクラスタリングがあると、照会のパフォーマンスが向上します。このような照会は、正しいディメンショ

ン値を持つレコードが含まれるページにのみアクセスします。しかも、該当するページはブロックまたはエクステントごとにグループ化されます。

第 2 の点として、クラスター索引を持つ表の場合、時間の経過とともに非クラスタリングされる可能性があります。しかし MDC 表は、すべてのディメンションにわたって自動的かつ連続的にクラスタリングを保守することができます。したがって、データの物理的順序をリストアするために MDC 表を再編成する必要がありません。

第 3 の点として MDC の場合、クラスター索引はブロック・ベースです。そのような索引は、通常のレコード・ベースの索引に比べて格段に小さいため、ディスク・スペースがずっと少なくて済み、スキャンも高速です。

マルチディメンション・クラスター索引

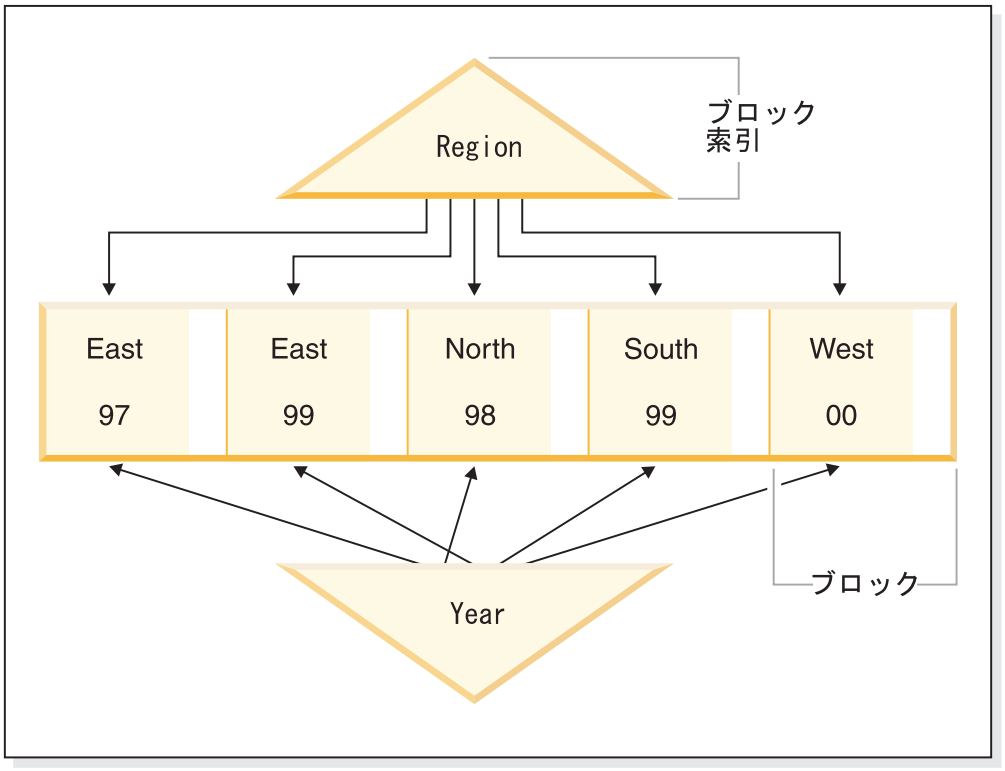

図 41. マルチディメンション・クラスター化表

ブロック索引:

図 41 の MDC 表は、 “Region” と “Year” の値が同じであるレコードをまとめて、いくつかの別々のブロック、つまり、エクステントとなるように、物理的に編成されています。 1 つのエクステントは、ディスク上のいくつかの連続したページで構成されているため、それらのレコード・グループは、物理的に連続したデータとしてクラスター化されています。表の各ページはそれぞれちょうど 1 個のブロックに属しており、どのブロックもすべて同じサイズ (同じページ数) です。ブロックのサイズは表スペースのエクステント・サイズと同じなので、ブロックの境界はエクステントの境界と揃うようになっています。この例の場合、2 個のブロック索引が作成されており、 1 個は “Region” ディメンション、もう 1 個は “Year” ディメンションに関するものです。それらのブロック索引には、表の中のブロックのみに対す

るポインターが含まれています。 “Region” ブロック索引により “Region” が “East“ である全レコードをスキャンすると、条件を満たすブロックが 2 個見つかります。それらの 2 個のブロックには “Region” が “East” であるレコードのすべて、そしてそのようなレコードだけが含まれており、連続したページまたはエクステントからなるセット 2 個に対してクラスター化されます。同時に、またそれとは完全に独立して、“Year” 索引によって、 1999 年から 2000 年までのレコードをスキャンすると、条件を満たすブロックが 3 個見つかります。それら 3 個のブロックのそれぞれに対するデータ・スキャンでは、 1999 年から 2000 年までのレコードがすべて、そしてそのようなレコードのみ戻され、それらのレコードは、各ブロック内で連続したページ上にクラスター化されています。

クラスター化に関するこれらの改善点以外に、 MDC 表には次のようなメリットがあります。

- ブロック索引は、レコード・ベースの索引と比較して大幅にサイズが小さいため、プローブとスキャンはずっと高速になります。
- ブロック索引とそれに対応するデータ編成により、きめ細かい「パーティション除去」、あるいは選択的表アクセスが可能になります。
- ブロック索引を利用した照会では、索引サイズが小さく、ブロックのプリフェッチが最適化されており、および対応するデータのクラスター化が保証されているというメリットがあります。
- 一部の照会においては、ロッキングおよび述部評価が少なくなります。
- ブロック索引では、ロギングおよび保守のためのオーバーヘッドが非常に少なくて済みます。ブロック索引の更新が必要になるのは、ブロックに最初のレコードを追加した時点か、ブロックから最後のレコードが除去された時点だけだからです。
- ロールインされるデータは、以前にロールアウトされたデータによって残された連続したスペースを再利用できます。

注: 単一ディメンションで定義された MDC 表であっても、 MDC のこれらの属性によるメリットがあります。また、クラスター索引を伴う通常表のための発展的な代替手段となり得ます。それは、ワークロードを構成する照会、表のデータの性質や分布など、数多くの要因に基づいて決定する必要があります。『ディメンション選択での考慮事項』と『DB2 Design Advisor と比較した場合の MDC Advisor のフィーチャー』を参照してください。

表を作成するとき、1 つまたは複数のキーを、データ・クラスター化に関連したディメンションとして指定することができます。それぞれの MDC ディメンションは、通常の索引キーと同様に 1 つまたは複数の列から構成されます。指定したそれぞれのディメンションごとにディメンションブロック索引 が自動的に作成され、オプティマイザーはこれを使用して、各ディメンションのデータにすばやく効率的にアクセスします。さらに、すべてのディメンションにわたってすべての列を含む複合ブロック索引 も自動的に作成されます。これは、挿入および更新アクティビティーにおいてデータのクラスター化を維持するために使われます。複合ブロック索引が作成されるのは、単一のディメンションにすべてのディメンションキー列が含まれないような場合のみです。また、複合ブロック索引は、列ディメンションの一部または全部の値を満たすデータに効率的にアクセスするために、オプティマイザーによって選択されることがあります。

注: 照会処理におけるこの索引の便利さは、そのキー部分の順序に依存します。キー部品の順序は、 CREATE TABLE ステートメントの ORGANIZE BY 文節で指定されるディメンションをパーサーが解析する際に、パーサーが検出する列の順序によって決まります。詳しくは、『MDC 表のブロック索引に関する考慮事項』を参照してください。

ブロック索引は、レコードではなくブロックを指すという点を除けば、その構造は通常の索引と同じです。ブロック索引は、ブロック・サイズに 1 ページの平均レコード数を乗算した分だけ通常の索引よりも小さくなります。 1 ブロック内のページ数は表スペースのエクステント・サイズと同じであり、 2 ページから 256 ページの範囲になります。ページ・サイズとしては 4 KB、8 KB、16 KB、または 32 KB が可能です。

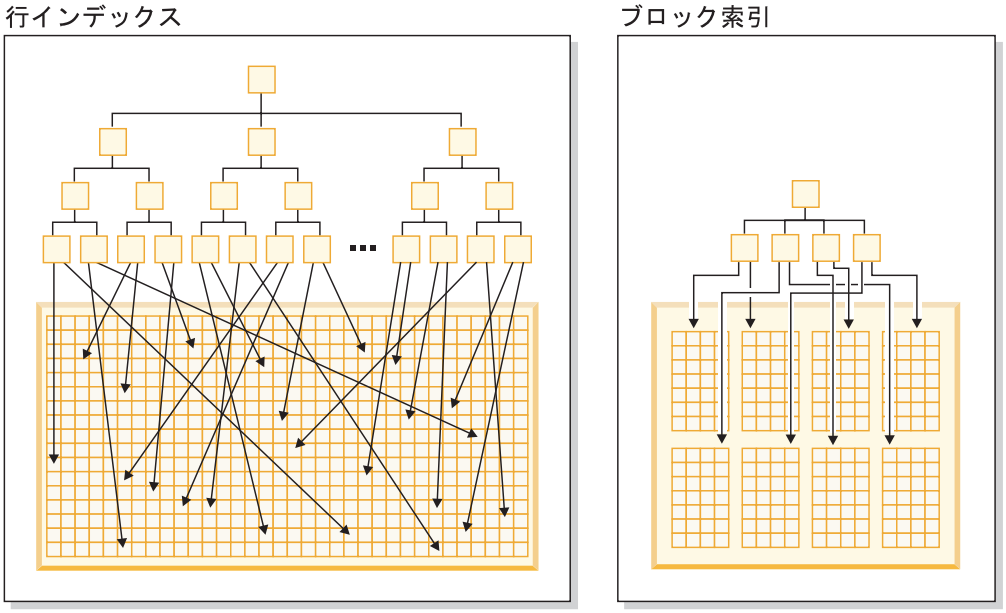

図 42. 行索引とブロック索引の違い

図 42 からわかるように、ブロック索引では、1 行に 1 つの項目ではなく、 1 ブロックに対して 1 個の索引項目が対応します。そのため、ブロック索引では、ディスク使用量が大幅に少なくなり、データ・アクセスははるかに高速です。

MDC 表では、複数のディメンション値のあらゆるユニークな組み合わせが論理セルを形成します。これは、物理的には 1 つ以上のページ・ブロックで構成される場合があります。論理セルには、その論理セルと同じディメンション値のレコードを入れるのに必要な数のブロックだけが含まれています。特定の論理セルと同じディメンション値の表の中にレコードが何も含まれていないなら、その論理セルにはブロックが割り振られません。特定のディメンションキー値のデータを含むブロックの集合は、スライス と呼ばれます。

MDC 表の処理:

たとえば、全国的な小売店の販売データを記録するための “Sales” という MDC 表があるとします。この表は、 “YearAndMonth” (年月) および “Region” (地域) という 2 つのディメンションでクラスター化されます。この表のレコードはブロック内

に保管されます。それらのブロックには、エクステントを入れるための十分な数の連続したディスク上のページが含まれます。 図 43 では、1 つのブロックが長方形として表され、表内に割り振られたエクステントの論理順序に従ってブロックに番号が付いています。図の中のグリッド (格子) はこれらのブロックの論理パーティションを示し、それぞれの四角は論理セルを表します。グリッド内の列または行は、特定のディメンションのスライスを示します。たとえば、“Region” 列に値 ’South-central’ (中南部) が含まれるすべてのレコードは、グリッドの ’South-central’ 列として定義されたスライスに含まれるブロックの中にあります。実際、このスライス内の各ブロックには、 “Region” フィールドが ’South-central’ であるレコードのみが含まれます。このように、 “Region” フィールドが ’South-central’ であるレコードを含むブロックのみが、このスライス (つまりグリッドの列) に含まれます。

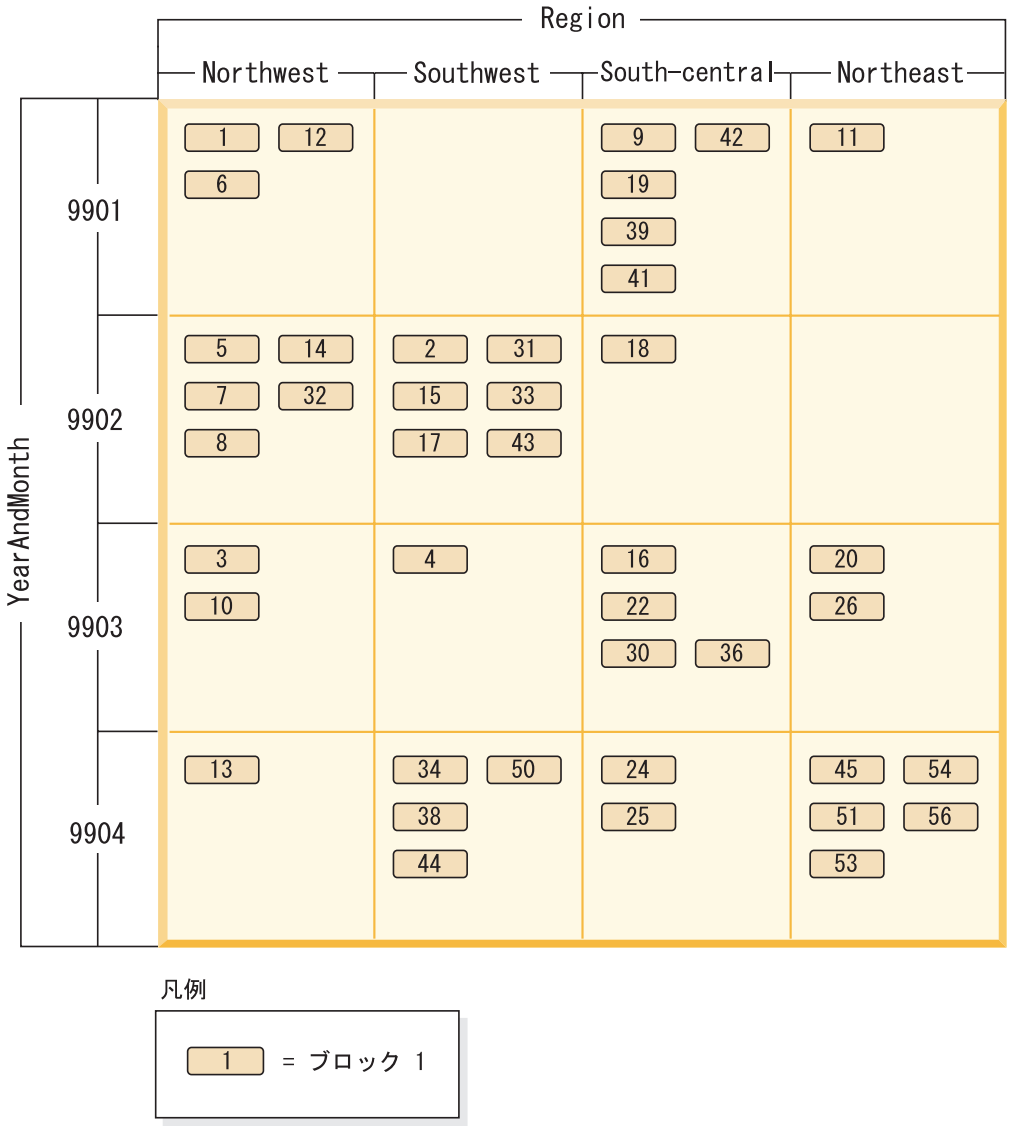

図 43. ディメンション ’Region’ および ’YearAndMonth’ が含まれるマルチディメンション表 Sales

スライスを構成するブロックを判別するために、あるいは特定のディメンションキー値を持つすべてのレコードを含むブロックを判別するために、表の作成時に、各ディメンションごとにディメンションブロック索引が自動的に作成されます。

167 ページの図 44 では、ディメンション “YearAndMonth” (年月) および “Region” (地域) に関するディメンションブロック索引がそれぞれ作成されています。それぞれのディメンションブロック索引の構造は従来の RID 索引と同様です。ただし、リーフ・レベルでは、キーがレコード ID (RID) でなくブロック ID (BID) をポイントします。 RID は、物理ページ番号とスロット番号 (レコードの存在するページ上のスロット) によって、表の中のレコードの位置を特定します。 BID は、そのエクステントの最初のページの物理ページ番号、およびダミー・スロット (0) によってブロックを表します。ブロック内のすべてのページはそこから始まって物理的に連続しており、ブロックのサイズがわかっているので、この BID を使用することにより、ブロック内のすべてのレコードを検索できます。

スライス (つまり、ディメンションの特定のキー値を持つすべてのレコード・ページを含むブロックの集まり) は、関連するディメンションブロック索引において、そのキー値に関する BID リストによって表されます。

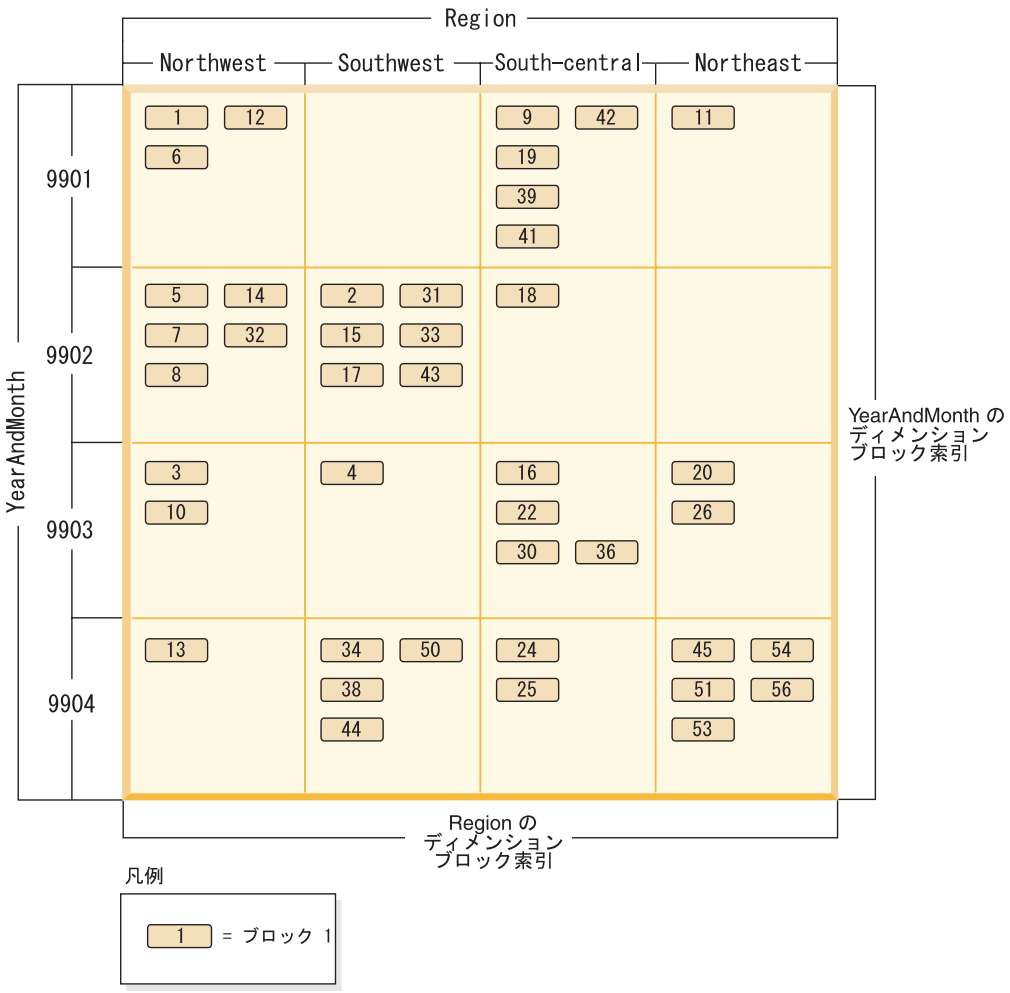

図 44. 'Region' および 'YearAndMonth' のディメンションを含む Sales 表とディメンションブロック索引

図 45 は、 "Region" に関するディメンションブロック索引のキーを例示しています。このキーは、キー値 ('South-central') と BID リストで構成されています。各 BID にはブロックのロケーションが含まれています。 図 45 にリストされているブロック番号は、 Sales 表 (165 ページの図 43 を参照) のグリッドにある 'South-central' スライスに含まれる番号と同じです。

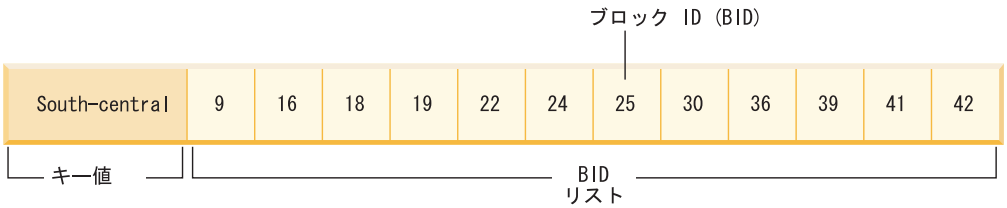

図 45. 'Region' に関するディメンションブロック索引のキー

同様に、ディメンション "YearAndMonth" の値が '9902' のすべてのレコードを含むブロックをリストするには、この値を "YearAndMonth" ディメンションブロック索引で検索します (168 ページの図 46 を参照)。

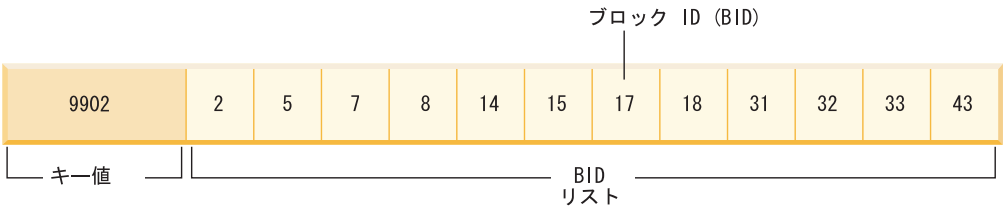

図 46. 'YearAndMonth' に関するディメンションブロック索引のキー

ブロック索引と照会のパフォーマンス:

MDC 表のブロック索引のいずれかに関するスキャンでは、表のうち、指定されたディメンション値のデータを含むことが保証されている連続したページの集合に、各 BID が対応しているため、クラスター化データ・アクセスが提供されます。さらに、ディメンションまたはスライスには、他のディメンションまたはスライスのクラスター因子に関係なく、そのブロック索引によって互いに独立してアクセスできます。これにより、マルチディメンション・クラスター化のマルチディメンション性が実現されます。

ブロック索引アクセスを利用する照会では、パフォーマンスを向上させるさまざまな要素があります。第 1 の点として、ブロック索引は通常の索引に比べてはるかに小さいため、ブロック索引のスキャンは非常に効率的です。第 2 の点として、データ・ページのプリフェッチは、ブロック索引を使用した順次検出に依存していません。 DB2 UDB は索引を先読みし、大ブロック入出力を使用してブロックのデータ・ページをメモリーにプリフェッチすることにより、スキャンで表の中のデータ・ページにアクセスする際に入出力が発生しないようにします。第 3 の点として、表のデータが連続したページ上でクラスター化されているため、入出力が最適化され、結果セットが表の選択した部分に配置されます。ブロック・ベースのバッファー・プールが使用されていて、そのブロック・サイズがエクステント・サイズの場合、ディスク上の連続したページから MDC ブロックがプリフェッチされて、メモリー内の連続ページに入れられます。そのようにして、クラスター化によるパフォーマンスがさらに向上します。最後の点として、各ブロックのレコードは、そのデータ・ページの小規模なリレーショナル・スキャンを使用して取り出されます。これは、RID ベースの取り出しに比べて、データ・スキャンのためのより高速な方法であることが少なくありません。

照会でブロック索引を使用するなら、表のうち、特定のディメンション値または値の範囲を含む部分に限定することができます。これにより、きめ細かい「パーティション除去」、つまりブロック除去が提供されます。その場合、他の照会、ロード、挿入、更新、および削除の操作では、この照会のデータ・セットとのやり取りをすることなく他のブロックにアクセスできるため、表の並行性がさらに高くなる可能性があります。

さらに、Sales 表が 3 つのディメンションに基づいてクラスター化されている場合には、表のすべてのディメンションのサブセットに対する照会を満たすようなレコードを含むブロック・セットを見つけるために、個々のディメンションブロック索引を使用することもできます。表のディメンションが "YearAndMonth"、"Region"、および "Product" (製品) であれば、 169 ページの図 47 のように、これを論理キューブと考えることができます。

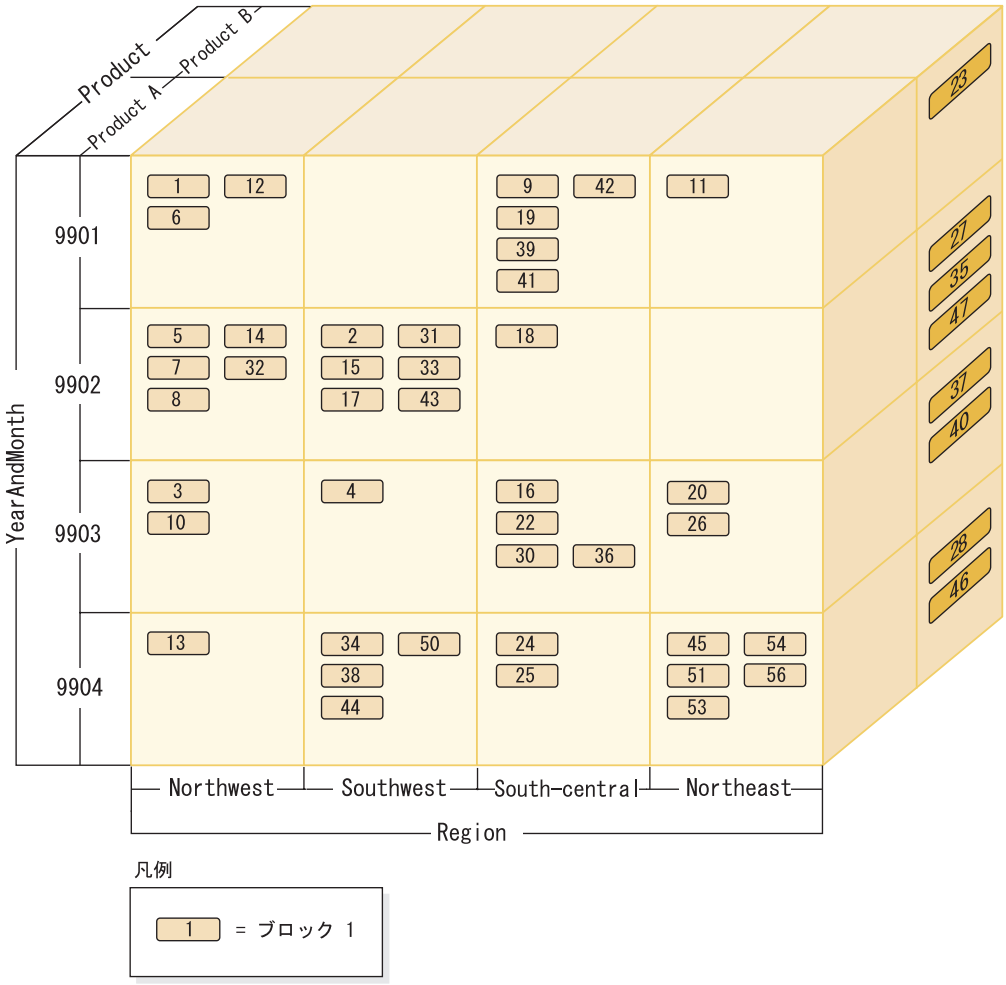

図 47. ディメンション 'Region'、'YearAndMonth'、および 'Product' が含まれるマルチディメンション表

図 47 に示されている MDC 表に対して、 4 個のブロック索引が作成されます。それぞれのディメンション "YearAndMonth"、 "Region"、および "Product" に対応するブロック索引が 3 個、そしてすべてのディメンション列をキーとするブロック索引が 1 個です。 "Product" が "ProductA"、かつ "Region" が "Northeast" であるすべてのレコードを取り出す場合、 DB2 UDB はまず "Product" ディメンションブロック索引から ProductA キーを検索します。 (図 48 を参照。) その後、DB2 UDB は "Region" ディメンションブロック索引で "Northeast" キーを検索することにより、 "Region" が "Northeast" であるすべてのレコードを含むブロックを判別します。 (170 ページの図 49 を参照。)

Product A	1	2	3	. . .	11	. . .	20	22	24	25	26	30	. . .	56

図 48. 'Product' に関するディメンションブロック索引のキー

Northeast	11	20	23	26	27	28	35	37	40	45	46	47	51	53	54	56

図 49. 'Region' に関するディメンションブロック索引のキー

論理 AND および論理 OR の演算子を使用すれば、複数のブロック索引スキャンを組み合わせることができます。また、スキャンするブロックの結果リストでも、クラスター化データ・アクセスが提供されます。

上の例で、 2 つのディメンション値を持つすべてのレコードを含むブロックのセットを見つけるには、この 2 つのスライスの交点を検出する必要があります。そのためには、 2 つのブロック索引キーの BID リストに対して論理 AND 演算を使用します。共通している BID 値は 11、20、26、45、54、51、53、および 56 です。

次の例は、2 ディメンションが関係する述部を含む照会において、ブロック索引での論理 OR 演算の使用方法を示すものです。 171 ページの図 50 では、 "Color" および "Nation" という 2 つのディメンションを含む MDC 表を使用していることが前提になっています。目標は、この MDC 表の中から、 "Color" が "blue" であるか、または "Nation" の名前が "USA" であるという条件を満たすレコードをすべて検索することです。

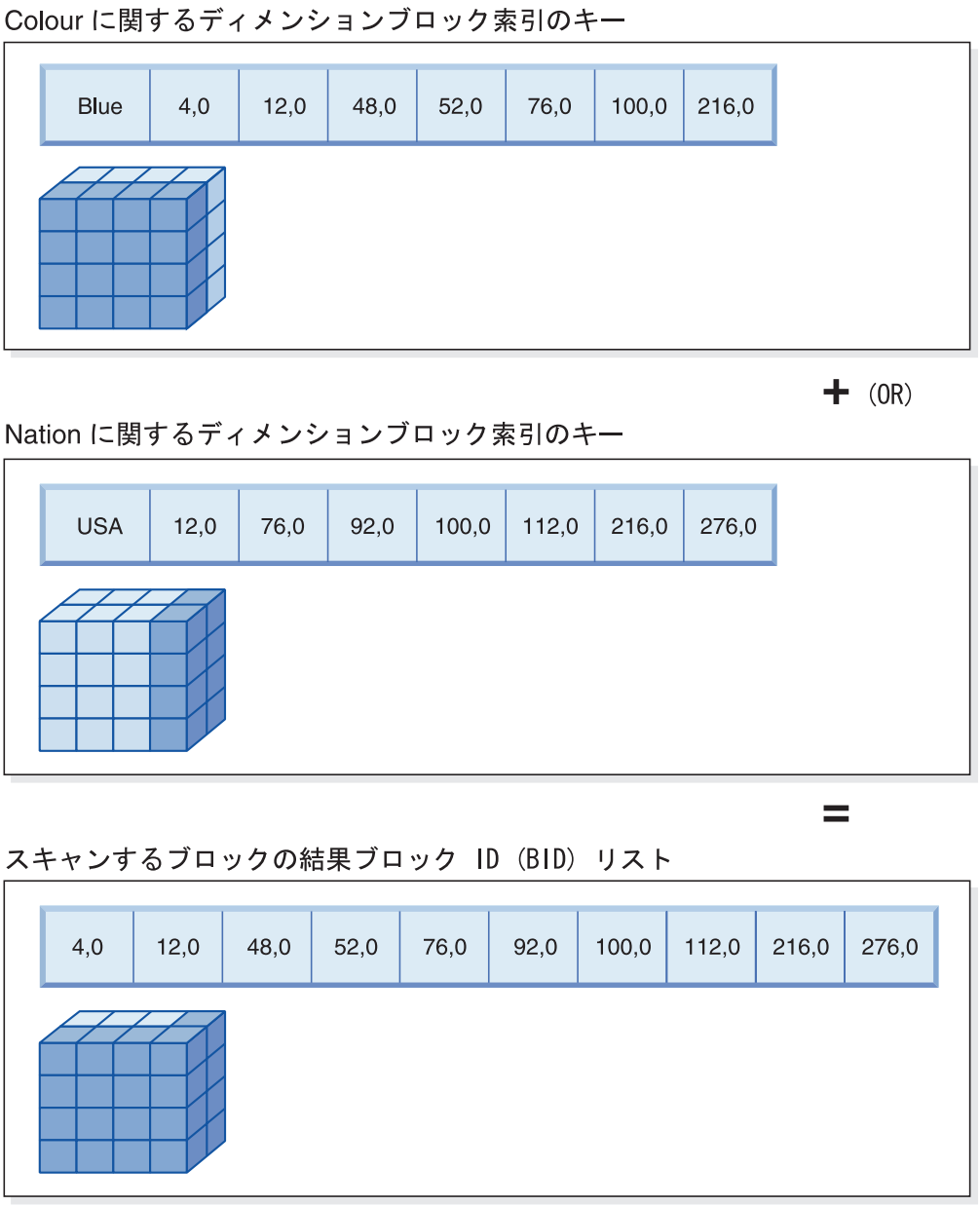

図 50. ブロック索引での論理 OR 演算の使用方法

この図には、2 つの別個のブロック索引スキャンの結果を結合して、述部制約を満たす値の範囲を決定する方法が示されています。

SELECT ステートメントの述部に基づいて、 2 個の別個のディメンションブロック索引スキャンが実行されます。 1 つは blue スライス、もう 1 つは USA スライスに対してです。 2 つのスライスの論理和を得て、 2 つのスライス内のブロックの和集合 (重複ブロックの除去を含む) を求めるためにメモリー内で論理 OR 演算が実行されます。

DB2 UDB にスキャン対象のブロックのリストがある場合、 DB2 UDB は各ブロックについて小規模なリレーショナル・スキャンを実行できます。ブロックのプリフェッチを実行することができます。各ブロックはディスク上のエクステントとして保管されており、単体としてバッファー・プールに読み取られるので、実行される

入出力処理は各ブロックごとにただ 1 つだけです。データに述部を適用することが必要なら、ブロック内のすべてのレコードのディメンションキー値は同じであることが確実なため、ディメンション述部で必要なのは単にブロック内の 1 レコードの述部を再適用することだけです。他の述部が存在する場合、 DB2 UDB はブロック内の残りのレコードに関してそれを検査するだけです。

MDC 表では、通常の RID ベースの索引もサポートされています。 RID とブロック索引は論理 AND 演算または論理 OR 演算で索引と結合できます。ブロック索引によりオプティマイザーは、アクセス・プランの選択肢が増えます。従来のアクセス・プラン (RID スキャン、結合、表スキャンなど) も使用できます。ブロック索引プランは、特定の照会に関して可能性のある他のすべてのアクセス・プランと共にコスト計算され、コストが最低のプランがオプティマイザーによって選択されます。

DB2 Design Advisor では、 MDC 表に対して RID ベースの索引を推奨したり、表に対して MDC ディメンションを推奨したりできます。

INSERT 操作中に自動的にクラスター化を維持する:

複合ブロック索引を使用すると、 MDC 表のデータ・クラスタリングが自動的に維持されます。これは、INSERT 操作の際、表のディメンションに基づいて物理的なデータ・クラスタリングを動的に管理および保守するために使われます。この複合ブロック索引には、レコードを含む表の各論理セルに関してのみ、キーが存在します。したがって、INSERT 中にこのブロック索引が使用されることにより、論理セルが表の中に存在するかどうかが短時間で効率的に判別され、存在する場合には、そのセルの特定のディメンション値セットを含むレコードが含まれるブロックが正確にどれであるかが判別されます。

挿入操作が実行されると、

- 挿入するレコードのディメンション値に対応する論理セルに対して、複合ブロック索引がプローブされます。
- 論理セルのキーが索引内にあるなら、そのブロック ID (BID) のリストにより、表のうち、論理セルと同じディメンション値のブロックの完全なリストが提供されます。 (173 ページの図 51 を参照。) これにより、表の中で、レコードを挿入するためのスペースを検索する対象となるエクステントの数が限定されます。
- 論理セルのキーが索引内にない場合、またはそれらの値を含むエクステントに余地がない場合、新しいブロックがその論理セルに割り当てられます。可能なら、新しいページ・エクステント (新しいブロック) で表を拡張する前に、表の中の空ブロックが再利用されます。

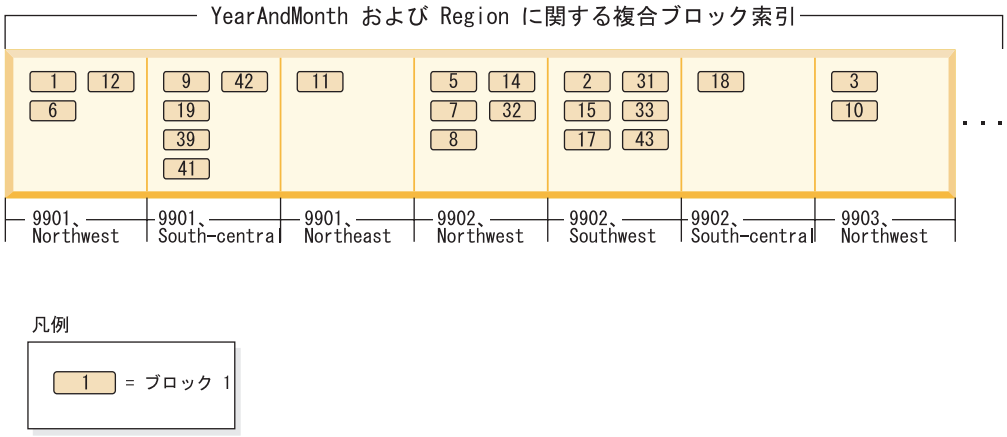

図 51. 'YearAndMonth' および 'Region' に関する複合ブロック索引

特定のディメンション値のデータ・レコードは、その値を含むすべてのレコードだけを含むブロック・セットの中にあることが保証されていることに注意してください。ブロックは、ディスク上の連続したページで構成されています。そのため、それらのレコードへのアクセスは順次アクセスであり、クラスター化が提供されます。そのレコードと同じディメンション値のセルのブロックにのみレコードが挿入されるようにすることにより、そのクラスター化が自動的に維持されます。論理セル中の既存のブロックに余地がないなら、空ブロックが再利用されるか、または新しいブロックが割り振られてその論理セルのブロック・セットに追加されます。あるブロックにデータ・レコードがなくなって空になると、そのブロック ID (BID) がブロック索引から除去されます。それにより、そのブロックはどの論理セル値とも関連がなくなり、将来別の論理セルによって再利用できるようになります。このように、セルとそれに関連するブロック索引項目は、表内に実際に存在するデータだけを入れるようにするため、必要に応じて表に動的に追加および削除されます。それを管理するために複合ブロック索引が使用されます。複合ブロック索引は、論理セル値を、それらの値のレコードが含まれるブロックにマップするからです。

クラスター化がこのようにして自動的に維持されるため、データの再クラスター化のために MDC 表の再編成が必要になることは決してありません。しかし、スペースの再利用のために再編成を使用することは可能です。たとえば、セルに数多くの低密度のブロックがあって、もっと少ないブロックでもデータが入る場合、あるいはオーバーフロー・レコードがある場合には、表を再編成することにより、各論理セルに属するレコードを圧縮すると共に、ブロックが必要最小限の数になるようにし、オーバーフロー・レコードを除去することができます。

以下の例は、照会処理のために複合ブロック索引を使用する方法を示しています。たとえば、Sales 表の中で、 "Region" が 'Northwest' で "YearAndMonth" が '9903' のレコードをすべて検出したい場合、 DB2 UDB はキー値 (9903,Northwest) を複合ブロック索引内で検索します (174 ページの図 52 を参照)。キーはキー値 (つまり '9903,Northwest') と BID リストで構成されます。ここでリストされている BID は 3 と 10 のみですが、実際、この 2 つの値を持つレコードを含むブロックは、Sales 表の中に 2 つしかありません。

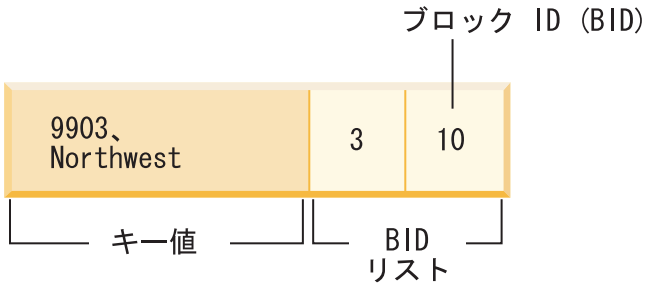

図 52. 'YearAndMonth' および 'Region' に関する複合ブロック索引のキー

挿入の際に複合ブロック索引がどのように使われるかを理解するために、ディメンション値が 9903 および Northwest である追加のレコードを挿入する場合を考えてください。 DB2 UDB はこのキー値を複合ブロック索引内で検索し、ブロック 3 および 10 の BID を検出します。これらのブロックに、この 2 つのディメンションキー値を持つレコードがすべて含まれます (他のブロックには含まれません)。利用可能なスペースがあるなら、 DB2 UDB は新しいレコードをそれらのブロックのいずれかに挿入します。これらのブロックのどのページにもスペースがない場合、DB2 UDB は新しいブロックを表に割り振るか、表内のすでに空になっているブロックを使用します。この例の場合、ブロック 48 が現在どの表によっても使用されていないことに注意してください。 DB2 UDB はレコードをブロックに挿入し、そのブロックの BID を複合ブロック索引と各ディメンションブロック索引に追加することによって、そのブロックを現在の論理セルに関連付けます。ブロック 48 を追加した後のディメンションブロック索引のキーが、 図 53 に示されています。

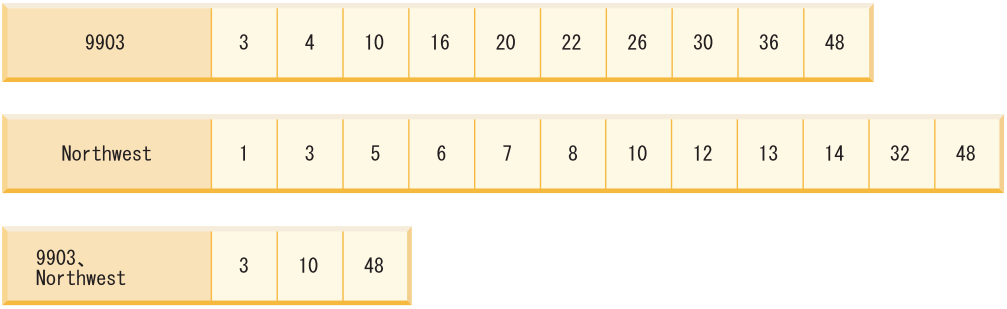

図 53. ブロック 48 追加後のディメンションブロック索引のキー

ブロック・マップ:

ブロックが空になった場合には、その BID がブロック索引から除去されることにより、その現在の論理セル値との関連が解除されます。そのブロックは、別の論理セルによって再利用できるようになります。それにより、新しいブロックを追加して表を拡張する必要が少なくなります。新しいブロックが必要になった場合には、以前に空になったブロックを、表から検索することなく短時間で見つかるようになっていなければなりません。

ブロック・マップは、 MDC 表の中から空のブロックを検索するために使用される新しい構造です。ブロック・マップは、別個のオブジェクトとして格納されます。

- SMS の場合、別個の .BKM ファイルとして
- DMS の場合、オブジェクト表の中の新しいオブジェクト記述子として

ブロック・マップは、表の各ブロックごとに 1 個ずつの項目を含む配列です。各項目は、1 つのブロックの一連の状況ビットで構成されます。状況ビットには、次のものが含まれます。

- 使用中。このブロックは、論理セルに割り当てられています。
- ロード。このブロックは最近ロードされました。スキャンではまだ見えません。
- 制約。このブロックは最近ロードされました。制約チェックはまだこれから実施されるところです。
- リフレッシュ。このブロックは最近ロードされました。マテリアライズされた照会ビューを更新する必要があります。

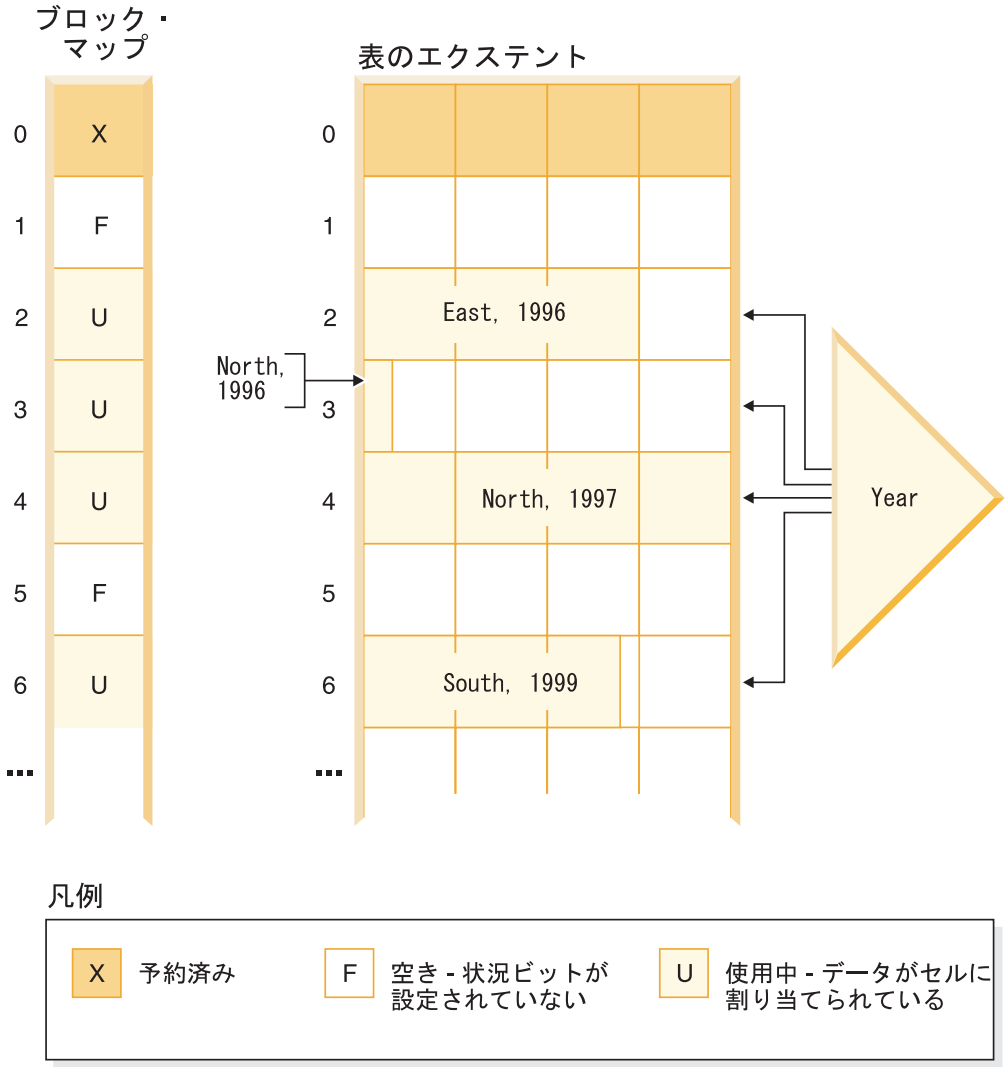

図 54. ブロック・マップの動作

図54 の左側には、表の各ブロックごとのさまざまな項目を含むブロック・マップ配列が示されています。図54 の右側には、表の各エクステントの使用状況が示されています。空きはいくらかありますが、ほとんどは使用中です。レコードは、ブロック・マップの中で使用中のマークが付けられたブロックだけに入っています。単純にするため、この図には 2 つのディメンションブロック索引のうち 1 つだけが示されています。

注:

1. ブロック索引に含まれるポインターは、ブロック・マップの中で IN USE とマークされているブロックに対するものだけです。
2. 最初のブロックは予約済みです。このブロックには、表のシステム・レコードが含まれています。

セルの中で使用できる空きブロックを見つけるのは簡単です。ブロック・マップをスキャンして FREE ブロック、つまりどのビットもセットされていないブロックを検索するだけです。

また、表スキャンでは、現在データが含まれているエクステントだけにアクセスするためにもブロック・マップが使用されます。使用中でないエクステントは、表スキャンに含める必要がありません。たとえばこの例 (175 ページの図 54) の場合、表スキャンは、最初の予約済みエクステントとその次の空エクステントをスキップして、表の 3 番目のエクステント (エクステント 2) から始まります。表のブロック 2、3、および 4 をスキャンした後、次のエクステントをスキップし (そのエクステントのデータ・ページには何も触らない)、その次からスキャンを続けます。

MDC 表での削除操作:

MDC 表でレコードを削除する場合、それがブロック内の最後のレコードでないなら、 DB2 UDB は単にそのレコードを削除し、その表にレコード・ベースの索引が定義されているならそこからそのレコードの RID を削除します。しかし、ブロック内の最後のレコードを削除する場合、DB2 UDB は、その IN_USE 状況を変更して、そのブロックの BID をすべてのブロック索引から削除することによって、ブロックを解放します。また、レコード・ベースの索引も存在するなら、RID をそこから削除します。

注: したがって、ブロック索引の項目の削除は、ブロックが完全に空である場合にのみブロック全体について 1 回しか必要ではありません。レコード・ベースの索引で削除する行ごとに実行する必要はありません。

MDC 表での更新操作:

MDC 表の場合、非ディメンション値の更新は、通常表の場合のように、その同じ場所で実行されます。その更新操作で可変長列が影響を受け、そのレコードがページに入らなくなる場合には、十分なスペースのある別のページが検索されます。その新しいページの検索は、まず同じブロックから開始されます。そのブロックにスペースがないなら、新しいレコード挿入のアルゴリズムが使用されて、十分なスペースを含む論理セルの中からページが検索されます。セル内にスペースがないために新しいブロックをセルに追加することが必要になるのでない限り、ブロック索引を更新する必要はありません。

ディメンション値の更新操作では、レコードの属する論理セルが変わるため、現在のレコードを削除した後、変更後のレコードを挿入するという操作として処理されます。現在のレコードを削除するとブロックが空になる場合には、ブロック索引を更新する必要があります。同じように、新しいレコードの挿入操作で新しいブロックに挿入することが必要な場合にも、ブロック索引を更新する必要があります。

ブロック索引を更新することが必要なのは、ブロックに最初のレコードを挿入する時点と、ブロックから最後のレコードを削除する時点だけです。したがって、ブロック索引の保守とロギングに関連した索引オーバーヘッドは、通常の索引に関連した索引オーバーヘッドに比べてずっと小さくなります。通常の索引としても可能な索引をブロック索引にするなら、そのようなブロック索引すべてに関して、保守とロギングのオーバーヘッドが大幅に削減されます。

MDC 表は既存の他の表と同様に扱われます。つまり、トリガー、参照保全、ビュー、およびマテリアライズ照会表を MDC 表に対して定義することができます。

MDC 表のロードに関する考慮事項:

データを定期的にデータウェアハウスにロールインする場合、 MDC 表の利点を活用することができます。 MDC 表では、ロードの際に空のブロックがまず再使用され、その後で、残りのデータ用に表を拡張して新しいブロックが追加されます。あるデータの集合 (たとえば 1 か月分の全データ) を削除した後、ロード・ユーティリティーを使って翌月のデータをロールインすれば、 (コミット済み) 削除によって空になったブロックを再使用できます。

データを MDC 表にロードした時点で、入力データはソートされた状態またはソートされていない状態のいずれかにすることができます。ソートされていない状態の場合には、次の点に注意してください。

- *util_heap* 構成パラメーターの設定を大きくしてください。

  ユーティリティー・ヒープのサイズを大きくすると、それはデータベースのすべてのロード操作 (およびバックアップとリストアの操作) に影響します。
- LOAD コマンドの DATA BUFFER 文節に指定する値を大きくしてください。

  この値を大きくすると、単一ロード要求に影響します。ユーティリティー・ヒープ・サイズは、複数の並行ロード要求の可能性に対応できるだけの大きさでなければなりません。
- バッファー・プールのために使用されるページ・サイズは、一時表スペースの最大ページ・サイズと同じでなければなりません。

ロードはブロック境界から始まるため、それは新しいセルに属するデータのため、または表の初期データのために使用するのが最善です。

MDC 表のロギングに関する考慮事項:

以前に RID で索引付けされた列を新たにディメンションにして、ブロック索引を使って索引付けすれば、索引の保守とロギングが大きく削減されます。ブロック全体の最後のレコードが削除された場合に限って、 DB2 UDB は BID をブロック索引から除去し、この索引操作をログに記録します。同様に、新しいブロックにレコードが挿入された場合 (つまり、それが論理セルの最初のレコードである場合、またはブロックいっぱいになった論理セルに挿入する場合) に限って、 DB2 UDB は BID をブロック索引に挿入し、その操作をログに記録します。ブロック内のレコード・ページの数は 2 から 256 までであるため、このようなブロック索引の保守およびロギングは比較的小規模です。表と RID 索引の追加と削除は、引き続きログに記録されます。

MDC 表のブロック索引に関する考慮事項:

MDC 表のディメンションを定義すると、ディメンションブロック索引が作成されます。さらに、マルチディメンションが定義されている場合には、複合ブロック索引も作成されることがあります。しかし、MDC 表にディメンションをただ 1 つだけ定義した場合、 DB2 UDB はブロック索引を 1 つだけ作成します。これは、ディメンションブロック索引および複合ブロック索引となります。同様に、列 A および (列 A、列 B) をディメンションとする MDC 表を作成した場合、 DB2 UDB は列 A に関するディメンションブロック索引と、 (列 A、列 B) に関するディメンションブロック索引を作成します。複合ブロック索引は表内のすべてのディメンションに関するブロック索引であるため、 (列 A、列 B) に関するディメンションブロック索引もまた複合ブロック索引として機能します。

さらに複合ブロック索引は、照会処理において、表内の特定の複数のディメンション値を持つデータにアクセスするためにも使われます。複合ブロック索引のキー部分の順序は、その使用法、または照会処理の適用度に影響する場合があることに注意してください。キー部分の順序は、 MDC 表の作成時に使われる ORGANIZE BY DIMENSIONS 文節全体の中にある列の順序によって決定されます。たとえば、次のステートメントを使用して表が作成された場合、

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

列 (c1,c4,c3,c2) に対して複合ブロック索引が作成されます。 c1 は DIMENSIONS 文節で 2 度指定されていますが、複合ブロック索引のキー部分では 1 度しか使用されず、最初に検出された順序で使用されることに注意してください。挿入処理の場合、複合ブロック索引内のキー部分の順序は無関係ですが、照会処理の場合には順序が影響を与える可能性があります。したがって、複合ブロック索引の列の順序を (c1,c2,c3,c4) としたい場合には、以下のステートメントを使って表を作成してください。

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『索引』
- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『マルチディメンション・クラスタリングの考慮事項』
- 179 ページの『マルチディメンション・クラスタリング (MDC) 表の設計』
- 「*管理ガイド: パフォーマンス*」の『MDC 表のための表および索引管理』
- 「*管理ガイド: パフォーマンス*」の『MDC 表の最適化ストラテジー』
- 189 ページの『マルチディメンション・クラスタリング (MDC) 表の作成、配置、および使用』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『MDC 表の表および RID 索引スキャンのロック・モード』
- 「*管理ガイド: パフォーマンス*」の『MDC 表のブロック索引スキャンのロック』

マルチディメンション・クラスタリング (MDC) 表の設計

マルチディメンション・クラスタリング表を使用することにした場合、選択するディメンションは、それらの表を使用したりブロック・レベルのクラスタリングを利用したりする照会のタイプに依存するだけでなく、もっと重要なこととして実際のデータの量と分布に依存します。 MDC 表の設計のそのような面に関する説明と、適切なディメンションおよびブロック・サイズの選択に関する指針を以下に示します。

MDC の利点を生かせる照会:

ご使用の表のクラスタリングディメンションの選択の際の 1 番目の考慮事項は、ブロック・レベルでクラスタリングによる効果のある照会の判別です。データに対して実施する作業を構成する照会に基づいてディメンションを選択する場合、複数の候補があるのが一般的です。それらの候補のランキングは重要です。列、特にカーディナリティーの低い列のうち、等式条件または範囲条件の述部を含む照会に関係したものは、クラスタリングディメンションのメリットを最大限活用でき、実際それらは候補として考慮するべきです。ディメンション表を持つ Star Join に含まれている MDC ファクト表内の外部キーのディメンションの作成を考慮したい場合もあるでしょう。複数のディメンションの自動および継続クラスタリング、およびエクステント・レベルまたはブロック・レベルでのクラスタリングのパフォーマンス上の利点に留意する必要があります。

マルチディメンション・クラスタリングを使用できる照会はたくさんあります。そのような照会の例を以下に示します。以下に示す例のいくつかでは、ディメンション c1、c2、および c3 の MDC 表 t1 を前提としています。その他の例では、ディメンション color および nation の MDC 表 mdctable を前提としています。

例 1:

```
SELECT .... FROM t1 WHERE c3 < 5000
```

この照会には、単一ディメンション上に範囲述部が含まれているので、 c3 上のディメンションブロック索引を使用して表にアクセスするように内部に再書き込みすることができます。索引は 5000 より少ない値をキーとするブロック ID (BID) のためにスキャンされ、小規模なリレーショナル・スキャンは、実際のレコードを検索するためにブロックの結果セットに適用されます。

例 2:

```
SELECT .... FROM t1 WHERE c2 IN (1,2037)
```

この照会には、単一ディメンション上に IN 述部が含まれており、ブロック索引ベースのスキャンをトリガーすることができます。この照会は、 c2 上のディメンションブロック索引を使用して表にアクセスするように内部に再書き込みすることができます。索引は、1 および 2037 の値を持つキーの BID のためにスキャンされ、小規模なリレーショナル・スキャンは、実際のレコードを検索するためにブロックの結果セットに適用されます。

例 3:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND NATION='USA'
```

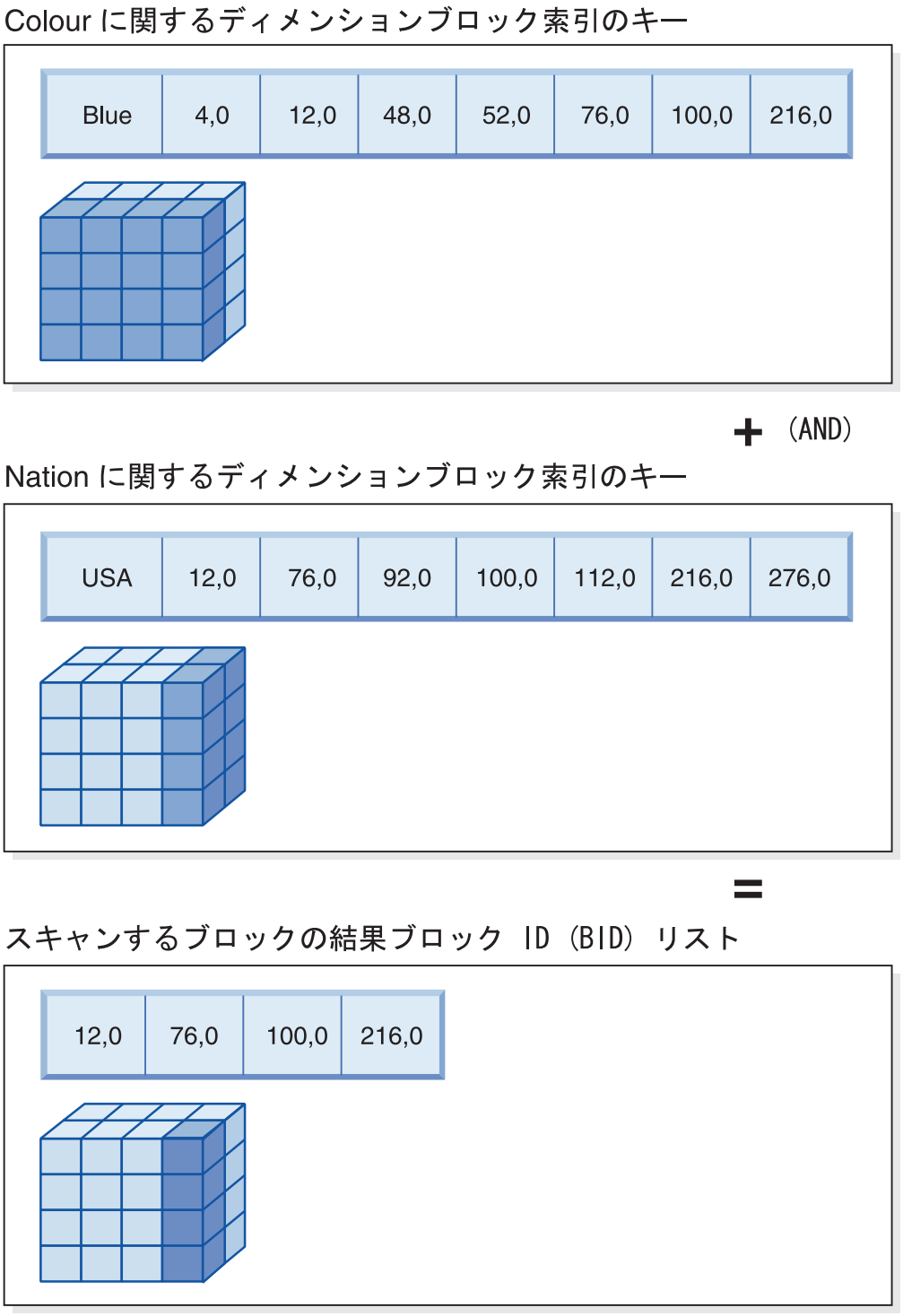

図 55. 2 個のブロック索引との論理 AND 演算を使用する照会要求

この照会要求を実行するため、次のようにします (図 55 を参照)。

- ディメンションブロック索引が検査されます。 1 つは Blue スライスのため、もう 1 つは USA スライスのためです。

- ブロック論理 AND 演算を実行して、 2 つのスライスの論理積を求めます。つまり、この論理 AND 演算により、 2 つのスライスの両方にあるブロックだけが判別されます。
- 表内の結果ブロックの小規模なリレーション・スキャンが実行されます。

例 4:

```
SELECT ... FROM t1
   WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 AND c3 < 5000
```

この照会には、c2 と c3 に関する範囲述部と、c1 に関する等式述部、そして論理 AND 演算が含まれています。これは、内部的にはディメンションブロック索引のそれぞれに対する表アクセスとして書き直すことができます。

- 値が 100 より大きいキーの BID を検索するため、 c2 ブロック索引のスキャンが実行されます。
- 値が 1000 と 5000 の間にあるキーの BID を検索するため、 c3 ブロック索引のスキャンが実行されます。
- 値が '16/03/1999' であるキーの BID を検索するため、 c1 ブロック索引のスキャンが実行されます。

次に、各ブロック・スキャンの結果 BID に対して論理 AND 演算が実行されて、その論理積が求められ、実際のレコードを検索するため、ブロックの結果セットに対して小規模なリレーショナル・スキャンが適用されます。

例 5:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR NATION='USA'
```

この照会要求を実行するため、次のようにします (171 ページの図 50 を参照)。

- ディメンションブロック索引が検査されます。各スライスに対して 1 つずつ実行されます。
- 論理 OR 演算により、2 つのスライスの和集合が求められます。
- 表内の結果ブロックの小規模なリレーション・スキャンが実行されます。

例 6:

```
SELECT .... FROM t1 WHERE c1 < 5000 OR c2 IN (1,2,3)
```

この照会には、 c1 ディメンションに関する範囲述部、c2 ディメンションに対する IN 述部、そして論理 OR 演算が含まれています。この照会は、ディメンションブロック索引 c1 および c2 に関する表アクセスを実行するように内部的に書き直すことができます。 5000 未満の値の検索のため c1 ディメンションブロック索引のスキャンが実行された後、値 1、2、および 3 の検索のため c2 ディメンションブロック索引の別のスキャンが実行されます。各ブロック索引スキャンの結果 BID に対して論理 OR 演算が実行された後、実際のレコードを検索するため、ブロックの結果セットに対して小規模なリレーショナル・スキャンが適用されます。

例 7:

```
SELECT .... FROM t1 WHERE c1 = 15 AND c4 < 12
```

この照会には、c1 ディメンションに関する等式述部、ディメンションでない列に関する範囲述部、および論理 AND 演算が含まれています。これは、c1 に 15 という

値を持つ表のスライスからブロックのリストを入手するために、 c1 上のディメンションブロック索引にアクセスするように内部的に書き直すことができます。 c4 上に RID 索引がある場合、 12 より少ない c4 を持つレコードの RID を検索するために索引スキャンを実行してから、レコードのこのリストとブロックの結果リストの AND 演算を実行できます。この論理積は c1 に 15 という値を持つブロック内にない RID を除去し、修飾するブロック内にあるリスト済み RID のみを表から検索します。

c4 上に RID 索引がない場合、ブロック索引は、修飾するブロックのリストをスキャンでき、それぞれのブロックの小規模なリレーショナル・スキャンの際に、見つかったそれぞれのレコードに述部 c4 < 12 を適用できます。

例 8:

color、year、nation、および部品番号 (PARTNO) に対する行 ID (RID) 索引のためのディメンションがあるというシナリオでは、次の照会が可能です。

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND PARTNO < 1000
```

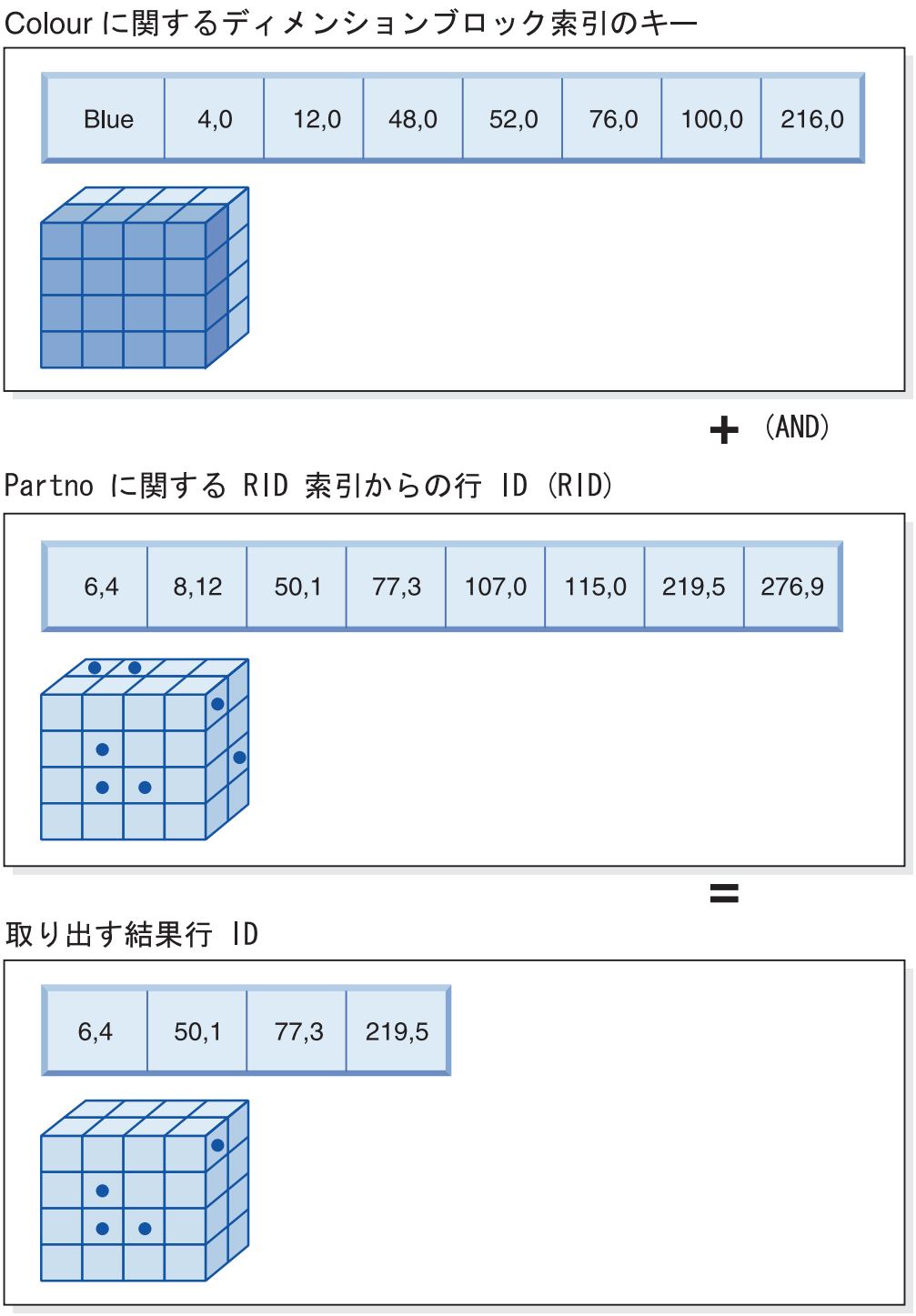

図 56. 1 つのブロック索引および行 ID (RID) 索引に対する論理 AND 演算を使用する照会要求

この照会要求を実行するため、次のようにします (図 56 を参照)。

- ディメンションブロック索引と RID 索引が検査されます。
- ブロックと RID の論理 AND 演算を使用することにより、スライスと、述部条件を満たす行の論理積を求めます。
- その結果は、修飾ブロックにも属している RID だけです。

例 9:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR PARTNO < 1000
```

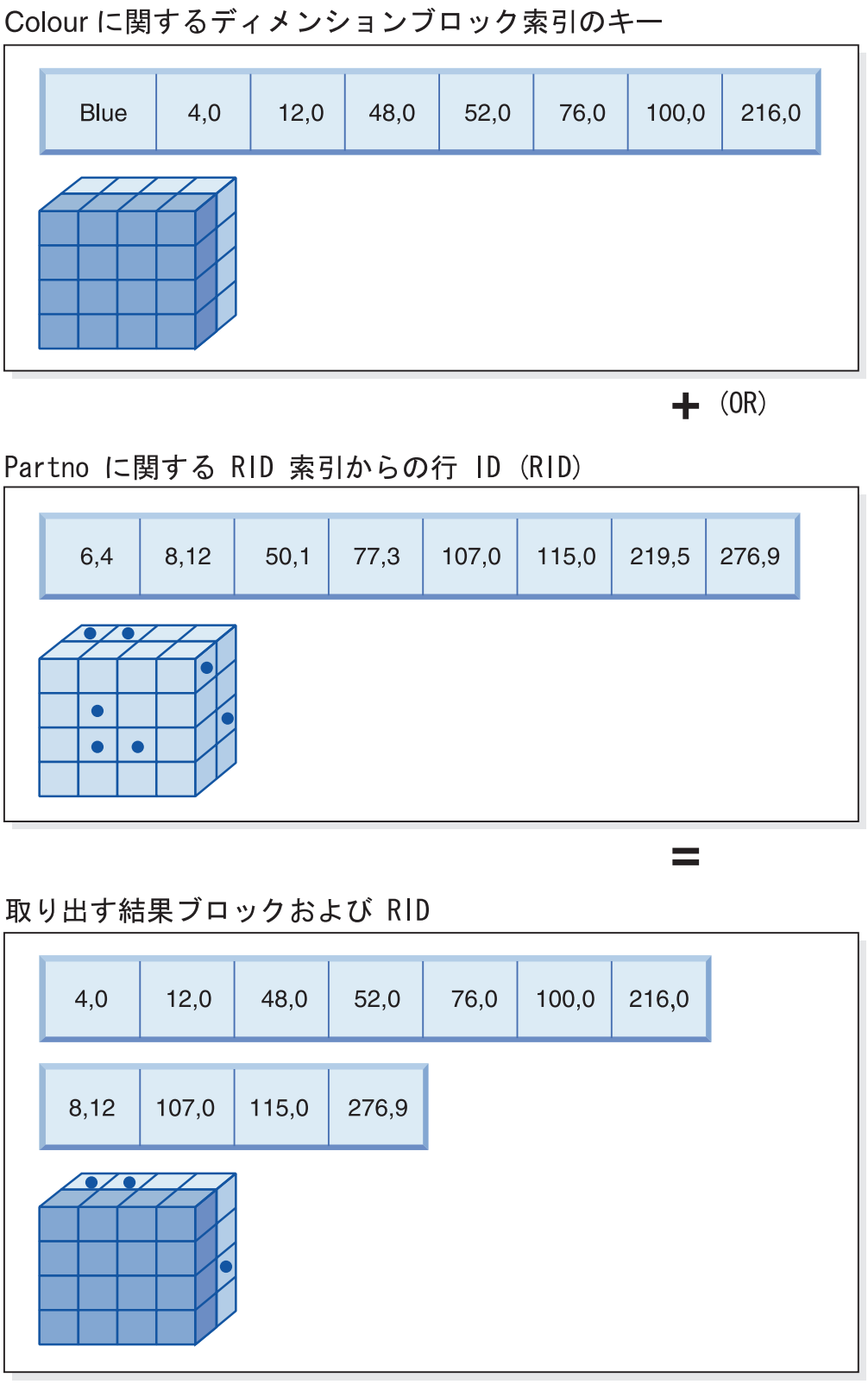

図 57. 論理 OR 演算を使用したブロック索引と行 ID の動作

この照会要求を実行するため、次のようにします (図 57 を参照)。

- ディメンションブロック索引と RID 索引が検査されます。

- ブロックと RID の論理 OR 演算を使用することにより、スライスと、述部条件を満たす行の和集合を求めます。
- 結果は、条件を満たすブロックに含まれる行と、該当ブロックに入らない付加的な RID のうち述部条件を満たすもののすべてです。各ブロックにおいて小規模なリレーショナル・スキャンが実行されることにより、そのレコードが取り出され、さらにそれらのブロックには含まれない付加的なレコードが別個に取り出されます。

例 10:

```
SELECT ... FROM t1 WHERE c1 < 5 OR c4 = 100
```

この照会には、 c1 ディメンションに関する範囲述部、非ディメンション列 c4 に対する等式述部、そして論理 OR 演算が含まれています。 c4 列に対する RID 索引があるなら、これは、 c1 に関するディメンションブロック索引と c4 に関する RID 索引の論理 OR 演算を実行するように内部的に書き直すことができます。 c4 に対する索引がない場合には、すべてのレコードをチェックする必要があるため、表スキャンを選択できます。 c1 に対する 4 未満の値のブロック索引スキャン、そして c4 に対する値 100 の RID 索引スキャンが、論理 OR 演算で結合されることになります。該当するブロックの中のすべてのレコードが条件を満たし、さらにそれらのブロックには含まれないレコードからも付加的な RID が取り出されるため、該当する各ブロックに対して小規模なリレーショナル・スキャンが実行されます。

例 11:

```
SELECT .... FROM t1,d1,d2,d3
   WHERE t1.c1 = d1.c1 and d1.region = 'NY'
      AND t2.c2 = d2.c3 and d2.year='1994'
      AND t3.c3 = d3.c3 and d3.product='basketball'
```

この照会には、Star Join が含まれています。この例では、t1 はファクト表で、主キー d1、d2、および d3 (ディメンション表) に対応する外部キー c1、c2、および c3 を持っています。ディメンション表は、MDC 表である必要がありません。region、year、および product は、 (ディメンション表が MDC 表である場合) 通常の (正規) 索引またはブロック索引を使用して索引化できるそれぞれのディメンション表の列です。 c1、c2、および c3 値上のファクト表にアクセスする際には、これらの列に対してディメンションブロック索引のブロック索引スキャンを実行し、次に、結果 BID の索引の AND 演算を実行できます。ブロックのリストがある場合は、レコードを入手するためにそれぞれのブロック上で小規模なリレーショナル・スキャンを実行できます。

セルの密度:

該当するディメンションとエクステント・サイズの選択は、 MDC の設計において**非常に**重要です。それらの要因により、表に関して予期されるセル密度が決まります。セル内のレコード数には関係なくすべての既存のセルに対してエクステントが割り振られるため、それらの要因は重要です。適切な値を選択するなら、ブロック・ベースの索引付けとマルチディメンション・クラスタリングを活用して、パフォーマンスが向上します。目指すところは、密度の高いブロックでマルチディメンション・クラスタリングを最大限に活用し、スペースの使用率を最大にすることです。

したがって、マルチディメンション表の設計の際に非常に重要になる考慮事項は、現在のデータおよび予測データに基づく、表内のセルの予期される密度です。照会パフォーマンスに基づいてディメンションのセットを選択し、それぞれのディメンションごとの可能な値の数に基づいて、表内のセルの潜在的な数が非常に大きくなるようにします。表内の可能なセルの数は、それぞれのディメンションのカーディナリティーのカルテシアン積と等しくなります。たとえば、ディメンション Day、Region、および Product に関する表をクラスター化し、 5 年分のデータを包含する場合、表内に `1821 days * 12 regions * 5 products = 109 260` とは異なる可能なセルを持つ場合があります。 2、3 のレコードしか含んでいないセルは、そのセル用のレコードを保管するために、そのセルに割り振られたページのブロック全体が必要です。ブロック・サイズが大きい場合、この表は、実際に必要なサイズよりもさらに大きくなる可能性があります。

セルの密度を最適にすることに貢献する可能性のある設計上の要因がいくつかあります。

- ディメンション数。
- 1 つ以上のディメンションの細分度。
- 表スペースのブロック (エクステント) サイズとページ・サイズ。

設計を最高のものにするため、次のステップを実行してください。

1. 候補となるディメンションを識別します。

   どの照会についてブロック・レベル・クラスタリングを利用できるかを判断します。次の特性のうちの一部または全部が当てはまる列があるかどうかに関して、潜在的なワークロードを調べてください。

   - IN リスト述部の範囲と等価性
   - データのロールインまたはロールアウト
   - GROUP BY 文節と ORDER BY 文節
   - 結合文節 (特にスタースキーマ環境の場合)

2. セルの数を見積もります。

   候補となるディメンションセットにより編成された表の中に、どれだけのセルが可能かを識別します。データの中に出現するディメンション値のユニークな組み合わせの数を判別します。表が存在するなら、その表に関してディメンションとなる各列の中で、異なる値の種類数を単に選択することによって、現在のデータに関して正確な数を判別できます。あるいは、表の統計データしかない場合には、ディメンション候補の列のカーディナリティーを乗算することにより概数を計算できます。

   注: パーティション・データベース環境の表の場合、考慮するどのディメンションにもパーティション・キーが関連しないなら、データの全体をパーティションの数で除算することによって、 1 セル当たりの平均データ量を計算することが必要になります。

3. スペースの占有度または密度を見積もります。

   平均して、格納されている行の数が少ないため一部しかデータが入っていないブロックが 1 セルにつき 1 個あることを考慮してください。 1 セル当たりの行数が小さいほど、データが一部しか入っていないブロックが多くなります。ま

た、(データの偏りがほとんど、あるいはまったくないとして) 平均して、 1 セル当たりのレコード数は、表の中のレコード数をセル数で除算して得られることに注意してください。しかし、パーティション・データベース環境の表の場合は、各パーティションごとに 1 セル当たりのレコード数を考慮する必要があります。ブロックは、パーティションごとのデータに対して割り振られるからです。データ・パーティション環境においてスペースの占有率と密度を見積もる際には、表全体ではなく各パーティションごとに 1 セル当たりの平均レコード数を考慮する必要があります。詳しくは、『マルチディメンション・クラスタリング (MDC) 表の作成、配置、および使用』を参照してください。

密度を改善するには、いくつかの方法があります。

- ブロック・サイズを小さくすることにより、一部しかデータが入っていないブロックの占めるスペースを少なくする。

  エクステント・サイズをある程度小さく設定することにより、各ブロックのサイズを小さくする。一部しかデータが入っていないブロックを含むセル、あるいはいくらかのレコードが含まれるだけのブロックを 1 個だけ含むセルに関して、各セルごとに無駄になるスペースが少なくなります。しかし、レコードの数の多いセルの場合、レコードを含めるために必要なブロック数が増えるというデメリットがあります。これにより、ブロック索引内のセル用のブロック ID (BID) の数が増加します。これらの索引がより大きくなり、これらの索引への挿入や削除がさらに増えるようになると、ブロックはより迅速に空になったり、満杯になったりします。また、追加して取り込まれたセル値用の表内のクラスター化されたデータはより小さいグループに分けられるのに対して、クラスター化されたデータはより少数のより大きいグループに分けられます。

- ディメンション数を少なくすることにより、または生成される列でのセルの細分度を高めることにより、セルの数を少なくする。

  1 つ以上のディメンションをロールアップして細分度を低下させることにより、カーディナリティーを下げることができます。たとえば、Region および Product における以前の例でデータのクラスター化を継続することができますが、 Day というディメンションを YearAndMonth のディメンションに置き換えます。これは、3600 という可能なセル数を使用して、YearAndMonth、Region、および Product に 60 (12 か月× 5 年)、12、および 5 というカーディナリティーを提供します。それにより、各セルに含まれる値の範囲が広くなり、レコード数が少なくなる可能性が低くなります。

  さらに、関係する列に対して頻繁に使用される述部を考慮する必要があります。たとえば、日付、四半期、あるいは日に対するものが大半であるかどうか、などです。これは、ディメンションの細分度の変更希望に影響を与えます。たとえば、ほとんどの述部が特定の Day 上にあり、Month 上に表をクラスター化した場合、 DB2® Universal Database (DB2 UDB) は、YearAndMonth 上のブロック索引を使用して、希望する Day が含まれている Month を迅速に限定し、関連したブロックのみにアクセスすることができます。しかし、ブロックのスキャンでは、どの日が条件を満たすかを判別するために Day 述部を適用する必要があります。ただし、Day 上でクラスター化する場合 (そして Day のカーディナリティーが高い場合)、スキャンするブロックを判別するために Day 上のブロック索引を使用でき、 Day 述部は修飾するそれぞれのセルの最初のレコードにのみ再適用されなければなりません。こ

の場合、ユーザー定義関数を使用して、 Region 列 (12 の異なる値を含む) を Regions West、North、South、および East にロールアップするように、セルの密度を増やすためにその他のディメンションの 1 つをロールアップすることを考慮する必要があります。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『設計アドバイザー』
- 159 ページの『マルチディメンション・クラスタリング表』
- 189 ページの『マルチディメンション・クラスタリング (MDC) 表の作成、配置、および使用』

マルチディメンション・クラスタリング (MDC) 表の作成、配置、および使用

MDC 表を作成する際には、さまざまな要素を考慮する必要があります。以下のセクションでは、 MDC 表を作成、配置、および使用する方法の決定に対して、現在使用しているデータベース環境 (たとえばパーティション・データベースかどうかなど)、また MDC 表のためのディメンションの選択がどう影響するかについて説明します。さらに、DB2® Design Advisor と、それを使用して前述の問題に関するアドバイスを提供する方法についても説明します。

既存の表のデータをマルチディメンション・クラスタリング (MDC) 表に移動する:

データウェアハウスまたは大規模データベース環境において、照会のパフォーマンスを向上させ、データ保守操作のオーバーヘッドを少なくするため、通常表のデータを、マルチディメンション・クラスタリング (MDC) 表に移動することができます。既存の表から MDC 表へデータを移動するには、データをエクスポートし、元の表をドロップし (オプション)、 (ORGANIZE BY DIMENSIONS 文節を含む CREATE TABLE ステートメントを使用して) マルチディメンション・クラスタリング (MDC) 表を作成し、その MDC 表にデータをロードします。

SYSPROC.ALTOBJ という ALTER TABLE プロシージャーを使用すると、既存の表から MDC 表へのデータ変換を実行できます。このプロシージャーは、DB2 Design Advisor から呼び出されます。表と表の間でのデータ変換にはかなりの時間が必要になる場合があり、それは表のサイズや変換の必要なデータの量によって異なります。

ALTOBJ プロシージャーは、表変更において次のことを実行します。

- 表の従属オブジェクトをすべてドロップします。
- 表の名前を変更します。
- 新しい定義を使用して表を作成します。
- 表の従属オブジェクトをすべて再作成します。
- 表に既存のデータを、新しい表で必要なデータに変換します。つまり、変換前の表のデータを選択し、そのデータを新しい表にロードします。その際には、変換前のデータ・タイプから変換後のデータ・タイプに変換するため、列関数が使用される場合があります。

SMS 表スペースのマルチディメンション・クラスタリング (MDC) 表:

MDC 表を SMS 表スペースに格納することを予定している場合には、複数ページ・ファイル割り振りを使用することをお勧めします。

注: 複数ページ・ファイル割り振りは、バージョン 8.2 以降で新たに作成されるデータベースではデフォルトです。

そのようにお勧めするのは、 MDC 表は常にエクステントを単位として拡張されるため、それらのエクステント内のすべてのページが物理的に連続していることが重要になるからです。したがって、複数ページ・ファイル割り振りを無効にすることには、スペースに関するメリットは何もありません。それだけでなく、それを有効にすることにより、各エクステント内のページが物理的に連続する可能性が格段に大きくなります。

DB2 Design Advisor と比較した場合の MDC Advisor のフィーチャー:

DB2 Design Advisor (db2advis) (旧称 Index Advisor) には、 MDC フィーチャーが含まれています。このフィーチャーでは、処理のパフォーマンスを向上させるため、基本列に対する低密度化を含め、MDC 表でクラスタリングディメンションを使用することを推奨します。*低密度化* という語は、クラスタリングディメンションのカーディナリティー (値の種類数) を少なくすることに関する数学用語です。低密度化の一般的な例としては、日付、曜日、日、四半期による低密度化があります。

推奨事項には、ディメンションの低密度化を定義する潜在的な生成列を識別することが含まれています。推奨事項には、可能性のあるブロック・サイズは含まれていません。 MDC 表の推奨事項作成では、表スペースのエクステント・サイズが使用されます。推奨される MDC 表が既存の表と同じ表スペース内で作成され、したがってエクステント・サイズが同じであることが前提になっています。 MDC ディメンションに関する推奨事項は、表スペースのエクステント・サイズによって変わることがあります。というのは、エクステント・サイズは、 1 個のブロックまたはセルに入るレコード数に影響するからです。それはセルの密度に直接影響します。

表に対して単一ディメンションでも複数ディメンションでも推奨されることがありますが、複合列ディメンションではなく単一列ディメンションだけが考慮されます。 MDC フィーチャーは、結果となる MDC ソリューションでのセルのカーディナリティーを小さくすることを目標として、サポートされているほとんどのデータ・タイプに関して低密度化を推奨します。データ・タイプの例外には CHAR、VARCHAR、GRAPHIC、および VARGRAPH のデータ・タイプが含まれます。サポートされているデータ・タイプは、すべて INTEGER にキャストされ、生成される式によって低密度化されます。

DB2 Design Advisor の MDC フィーチャーの目標は、パフォーマンスが改善される MDC ソリューションを選択することです。第 2 の目標は、データベースのストレージ拡張を控え目なレベルに保つことです。表ごとの最大ストレージ拡張の判別には、統計的手法が使用されます。

Advisor 内の分析操作には、ブロック索引アクセスのメリットだけでなく、表のディメンションに対する挿入、更新、および削除操作における MDC の影響も含まれます。表に対するそれらのアクションには、セル間でレコードを移動させることにな

る可能性があります。また、分析操作は、特定の MDC ディメンションに関するデータの編成処理の結果としての表拡張による潜在的なパフォーマンスの影響をモデル化します。

MDC フィーチャーを有効にするには、 db2advis ユーティリティーで `-m <advise type>` フラグを使用します。マルチディメンション・クラスタリング表を指定するため、アドバイス・タイプとして "C" を使用します。アドバイス・タイプには、"I" (索引)、"M" (マテリアライズ照会表)、 "C" (MDC)、および "P" (データベース・パーティション) があります。アドバイス・タイプは、互いに組み合わせて使用できます。

注: DB2 Design Advisor は、サイズが 12 エクステントより小さい表を考慮しません。

Advisor は、推奨事項提供において MQT と通常表の両方を分析します。

MDC フィーチャーからの出力には、次のものが含まれます。

- MDC ソリューションに出現する低密度化されたディメンションについて、表ごとに生成される生成列。
- 各表について推奨される ORGANIZE BY 文節。

推奨事項は、 stdout と、EXPLAIN 機能の一部である ADVISE 表の両方に報告されます。

マルチディメンション・クラスタリング (MDC) 表とデータベース・パーティション:

マルチディメンション・クラスタリングは、データベース・パーティションと組み合わせて使用することができます。実際、MDC はデータベース・パーティションを補います。データベース・パーティションは、複数の物理または論理ノードの間に表データを分散させるために使用されます。その目的は、次のとおりです。

- 複数のマシンを利用して、並列処理される要求の数を増やす。
- 単一パーティションの限界を超えて、表の物理サイズを大きくする。
- データベースのスケーラビリティーを改善する。

表をパーティション化する理由は、表が MDC 表か通常表かということとは別です。たとえば、パーティション・キーを構成する列を選択するための規則は同じです。 MDC 表のパーティション・キーには、列がその表のディメンションの一部であるかどうかによらず、どんな列でも関係することがあります。

パーティション・キーが表のディメンションと同一なら、各データベース・パーティションにはそれぞれ表の異なる部分が入れられます。たとえば、この章で使用しているサンプルの MDC 表は、 2 つのパーティションにわたって color によりパーティション化され、その後、Color 列を使用してデータが分割されます。その結果、Red スライスと Blue スライスが 1 つのパーティションに、 Yellow スライスがもう 1 つのパーティションになることがあります。パーティション・キーが表のディメンションと同一でない場合には、各データベース・パーティションには、それぞれ各スライスのデータのサブセットが入れられることになります。ディメンションを選択し、セル占有度を見積もる際には (『セルの密度』のセクションを参

照)、平均として 1 セル当たりの合計データ量は、データ全体をパーティション数で除算して得られることに注意してください。

マルチディメンションのマルチディメンション・クラスタリング (MDC) 表:

照会で一部の特定の述部が頻繁に使用されることがあらかじめわかっている場合、ORGANIZE BY DIMENSIONS 文節を使って、関連する列に基づいて表をクラスター化することができます。

例 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

例 1 の表は、論理キューブ (つまり 3 ディメンション) を形成する 3 つの固有な列における値に基づいてクラスター化されています。こうすれば、照会処理の際、1 つまたは複数のディメンションにおいて表をスライスすることにより、適切なスライスまたはセルに含まれるブロックだけがリレーショナル演算子によって処理されるようにすることができます。ブロック・サイズ (ページ数) が、表のエクステント・サイズになることに注意してください。

複数列に基づくディメンションのマルチディメンション・クラスタリング (MDC) 表:

それぞれのディメンションを、1 つまたは複数の列で構成することが可能です。一例として、2 つの列を含むディメンションに基づいてクラスター化される表を作成することができます。

例 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

例 2 では、c1 および (c3, c4) の 2 つのディメンションに基づいて表がクラスター化されます。こうすると、照会処理においては、ディメンション c1 もしくは複合ディメンション (c3, c4) に基づいて表を論理的にスライスすることができます。この表のブロック数は例 1 の表と同じですが、ディメンションブロック索引の数は 1 つ少なくなります。例 1 では、列 c1、c3、 c4 それぞれに関する 3 つのディメンションブロック索引が作成されます。例 2 の場合は、列 c1 に関するディメンションブロック索引と、列 c3 および c4 に関するディメンションブロック索引の 2 つが作成されます。この 2 つの方法の主な違いは、例 1 の場合、 c4 のみを扱う照会が c4 に関するディメンションブロック索引を使用して、関連するデータ・ブロックにすばやく直接的にアクセスできることです。例 2 では c4 がディメンションブロック索引の 2 番目のキー部分であるため、 c4 のみを扱う照会はより多くの処理を行います。ただし、例 2 の場合、 DB2 Universal Database™ (UDB) が保守および保管するブロック索引の数は 1 つ少なくなります。

DB2 Design Advisor は、複数列を含むディメンションに関する推奨事項を作成しません。

生成列をディメンションとするマルチディメンション・クラスタリング (MDC) 表:

ディメンションをクラスタリングする際、生成列を使用することもできます。生成列に基づくクラスター化は、細分性を粗くしてディメンションをロールアップする場合に便利です。たとえば、住所を地理上の場所または地域にロールアップする場合や、日付を週、月、または年にロールアップする場合などです。このようなディメンションのロールアップを実装するには、生成された列を使用することができます。このタイプの列定義では、ディメンションを表す式を使って列を作成することができます。例 3 のステートメントによって作成される表は、基本となる 1 つの列、および 2 つの生成列に基づいてクラスター化されます。

例 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
   c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1))
      ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

例 3 では、列 c5 が c3 および c4 に基づく式であり、列 c6 は列 c1 を時間的に粗い細分性にロールアップします。このステートメントによって、列 c2、c5、および c6 の値に基づいて表がクラスター化されます。

生成列ディメンションに対する範囲照会では単調列関数が必要:

生成列に対してディメンションの範囲述部を派生させるには、式が単調でなければなりません。生成列に対してディメンションを作成する場合、基本列に対する照会では、その生成列に対するブロック索引を利用することによりパフォーマンスが改善されます。ただし、1 つの例外があります。基本列 (日付など) に対する範囲照会で、ディメンションブロック索引による範囲スキャンを使用するには、 CREATE TABLE ステートメントで列を生成するために使用する式が単調でなければなりません。列式には任意の有効な式 (ユーザー定義関数 (UDF) を含む) を含めることができますが、その式が単調でない場合、基本列に対する述部のうち、照会を満たすためにブロック索引を使用できるのは等式述部または IN 述部だけです。

たとえば、month = INTEGER (date)/100 という生成列に対してディメンションが定義されている MDC があるとしましょう。ディメンション (month) に対する照会では、ブロック索引スキャンが可能です。基本列 (date) に対する照会でも、ブロック索引スキャンを実行することによってスキャン対象のブロックを限定してから、それらのブロックだけに含まれる行に対して、date に関する述部を適用することができます。

コンパイラーは、ブロック索引スキャンで使用する付加的な述部を生成します。たとえば、次の照会の場合、

```
SELECT * FROM MDCTABLE WHERE DATE > "19999/03/03" AND DATE < "2000/01/15"
```

コンパイラーは、“month >= 199903” かつ “month < 200001” という、付加的な述部を生成します。これは、ディメンションブロック索引スキャンの述部として使用できます。結果ブロックのスキャンにおいては、オリジナルの述部がブロック中の行に適用されます。

非単調式の場合、そのディメンションに対して適用できるのは等式述部だけです。非単調関数の 1 つの例として、例 3 の列 c6 の定義に出てきた MONTH( ) があります。列 c1 が日付、タイム・スタンプ、または日付やタイム・スタンプを表す有効なストリング表記である場合、この関数は 1 から 12 までの範囲の整数値を戻し

ます。この関数の出力は決まりきったものですが、実際には以下のようなステップ関数と同じような出力 (つまり循環パターン) を生成します。

```
MONTH(date('99/01/05')) = 1
MONTH(date('99/02/08')) = 2
MONTH(date('99/03/24')) = 3
MONTH(date('99/04/30')) = 4
...
MONTH(date('99/12/09')) = 12
MONTH(date('00/01/18')) = 1
MONTH(date('00/02/24')) = 2
...
```

この例の一連の日付は単調に大きくなっていますが、 MONTH(date) はそうではありません。より厳密に言うと、date1 が date2 よりも大きいとき、 MONTH(date1) が MONTH(date2) よりも常に大または等しくなるとは限りません。単調性を考慮する必要があるのは、このような場合です。このような非単調性は許可されていますが、基本となる列の範囲述部がそのディメンションの範囲述部を生成できないという点で、ディメンションを制限してしまいます。しかし、式の範囲述部 (たとえば `where month(c1) between 4 and 6`) は使用できます。こうすれば、開始キーを 4、終了キーを 6 として、ディメンションに関する索引を通常の方法で使用できます。

この関数を単調にするには、月の上位部として年を使用する必要があります。 DB2 UDB の組み込み関数 INTEGER には、日付に関する単調式を定義するのに役立つ拡張機能があります。 INTEGER(date) は日付の整数表記を戻します。この表記をさらに分割して、年および月を表す表記を検出することができます。たとえば、`INTEGER(date('2000/05/24'))` は `20000524` を戻し、`INTEGER(date('2000/05/24'))/100 = 200005` となります。関数 INTEGER(date)/100 は単調です。

同様に、組み込み関数 DECIMAL および BIGINT にもまた、単調関数を作成できる拡張機能があります。 DECIMAL(timestamp) はタイム・スタンプの 10 進表記を戻します。この表記を単調式で使用して、月、日、時間、分などに関して単調増加する値を生成することができます。 BIGINT(date) は INTEGER(date) と同様に、日付に関する BIGINT 表記を戻します。

表において生成された列を作成するとき、あるいはディメンション文節の式からディメンションを作成するときに、 DB2 UDB は可能な限り式の単調性を判別します。DATENUM( )、 DAYS( )、 YEAR( ) などの一部の関数は、単調性を保持する関数として認識されます。さらに、さまざまな数式 (たとえば列や定数の除算、乗算、加算) が単調性を保持します。式の単調性が保持されないと DB2 UDB が判断した場合、あるいは判別が不可能な場合には、ディメンションの基本列では等式述部のみを使用することができます。

関連概念:

- 131 ページの『エクステント・サイズ』
- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『マルチディメンション・クラスタリングの考慮事項』
- 159 ページの『マルチディメンション・クラスタリング表』
- 179 ページの『マルチディメンション・クラスタリング (MDC) 表の設計』

関連タスク:

- 「*管理ガイド: インプリメンテーション*」の『表のディメンションの定義』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 「*コマンド・リファレンス*」の『db2empfa - 複数ページ・ファイル割り振りの使用可能化コマンド』

第 6 章 分散データベースの設計

1 つのトランザクションでの単一のデータベースの更新

トランザクションの最も単純な形は、単一の作業単位内で 1 つのデータベースに対して読み書きを行うことです。この種類のデータベース・アクセスは、*リモート作業単位* と呼ばれます。

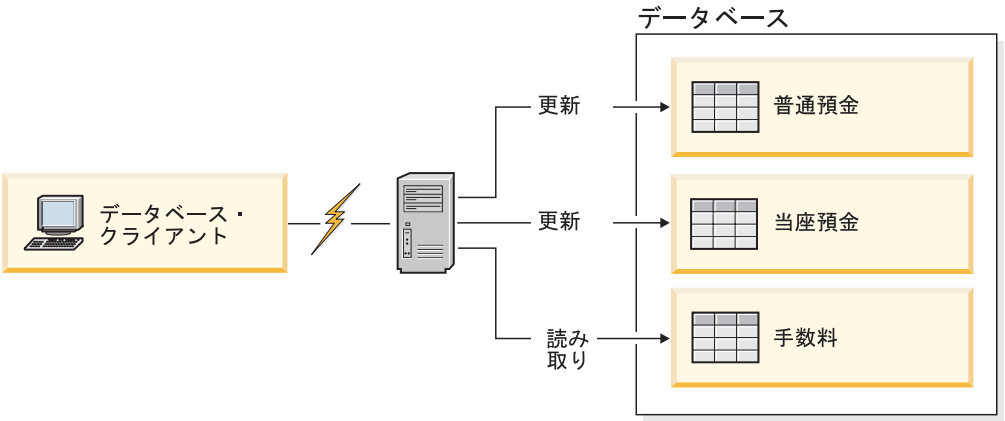

図 58. 1 つのトランザクションでの単一のデータベースの使用

図 58 には、当座預金の表、普通預金の表、および銀行手数料料金表からなるデータベースにアクセスして、基金振替アプリケーションを実行するデータベース・クライアントが示されています。このアプリケーションで行う必要のある処理は次のとおりです。

- ユーザー・インターフェースから、振替金額を受け取る。
- 普通預金から上記の金額を減算し、新規の残高を計算する。
- 手数料料金表を読んで、その金額での普通預金の手数料を調べる。
- 普通預金から手数料を減算する。
- 振替金額を当座預金に加算する。
- このトランザクション (作業単位) をコミットする。

手順:

このようなアプリケーションをセットアップするには、以下のようにします。

1. 普通預金、当座預金、および手数料料金表を、同じデータベースの中に作成する。
2. 物理的にリモートの場合、適切な通信プロトコルを使用するようにデータベース・サーバーを設定する。
3. 物理的にリモートの場合、データベース・サーバー上のデータベースを識別するようにノードとデータベースのカタログを作成する。

4. アプリケーション・プログラムをプリコンパイルし、タイプ 1 接続を指定する。つまり、 PRECOMPILE PROGRAM コマンドで CONNECT 1 (デフォルト) を指定する。

関連概念:

- 30 ページの『作業単位』

関連タスク:

- 198 ページの『複数のデータベース・トランザクションでの単一データベースの更新』
- 199 ページの『トランザクションで複数のデータベースを更新する』

関連資料:

- 「コマンド・リファレンス」の『PRECOMPILE コマンド』

単一トランザクションでの複数のデータベースの使用

単一トランザクションで複数のデータベースを使用している場合は、トランザクション内で更新されるデータベースの数によって、環境の設定と管理に求められる要件が異なってきます。

複数のデータベース・トランザクションでの単一データベースの更新

データが複数のデータベースに分散している場合には、 1 つのデータベースを更新中に、それ以外のデータベースから読み取ることもできます。このようなタイプのアクセスは、単一の作業単位内 (トランザクション) で実行できます。

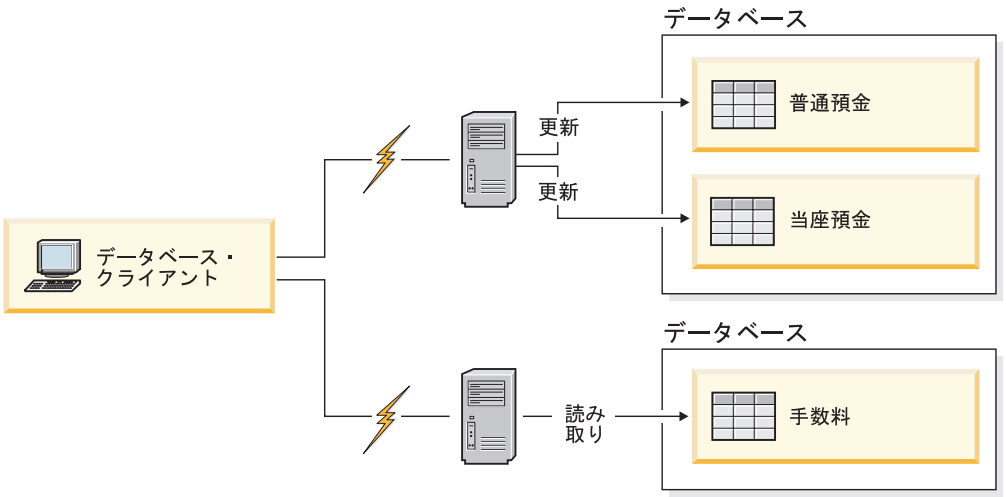

図 59. 単一トランザクションでの複数のデータベースの使用

図 59 には、 2 つのデータベース・サーバー (1 つは当座預金の表と普通預金の表を含み、もう 1 つは銀行手数料料金表を含む) にアクセスして、基金振替アプリケーションを実行するデータベース・クライアントが示されています。

手順:

この環境で基金振替アプリケーションをセットアップするには、以下の処理を行う必要があります。

1. 該当するデータベースの中に必要な表を作成する。
2. 物理的にリモートの場合、適切な通信プロトコルを使用するようにそれぞれのデータベース・サーバーを設定する。
3. 物理的にリモートの場合、データベース・サーバー上のデータベースを識別するようにそれぞれのノードとデータベースのカタログを作成する。
4. アプリケーション・プログラムをプリコンパイルし、タイプ 2 接続を指定 (つまり PRECOMPILE PROGRAM コマンドで CONNECT 2 を指定) して、 1 フェーズ・コミットを指定 (つまり、 PRECOMPILE PROGRAM コマンドで SYNCPOINT ONEPHASE を指定) する。

データベースがホストまたは iSeries データベース・サーバーにある場合、これらのサーバーへの接続には DB2 Connect™ が必要です。

関連概念:

- 30 ページの『作業単位』

関連タスク:

- 197 ページの『1 つのトランザクションでの単一のデータベースの更新』
- 199 ページの『トランザクションで複数のデータベースを更新する』

関連資料:

- 「*コマンド・リファレンス*」の『PRECOMPILE コマンド』

トランザクションで複数のデータベースを更新する

データが複数のデータベースに分散している場合は、単一のトランザクションで複数のデータベースの読み取りと更新を行いたいと思う場合があります。このタイプのデータベース・アクセスは、*マルチサイト更新* と呼ばれます。

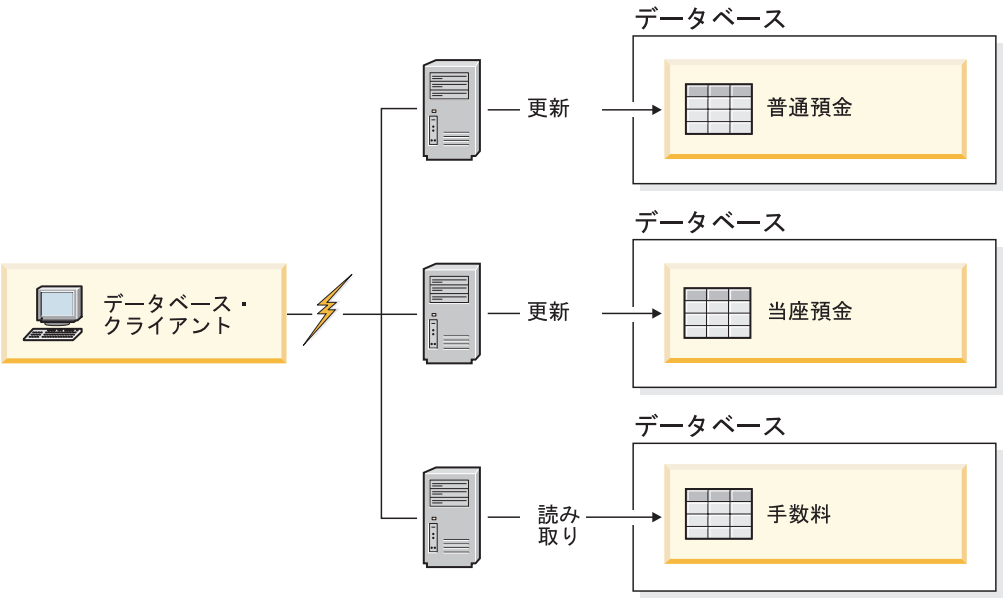

図 60. 単一トランザクションでの複数のデータベースの更新

図 60 には、3 つのデータベース・サーバー (それぞれ当座預金の表、普通預金の表、および銀行手数料料金表を含む) にアクセスして、基金振替アプリケーションを実行するデータベース・クライアントが示されています。

手順:

この環境での基金振替アプリケーションのセットアップには、2 つの方法があります。

1. DB2 Universal Database™ (DB2 UDB) トランザクション・マネージャー (TM) を使用する方法:
   a. 該当するデータベースの中に必要な表を作成する。
   b. 物理的にリモートの場合、適切な通信プロトコルを使用するようにそれぞれのデータベース・サーバーを設定する。
   c. 物理的にリモートの場合、データベース・サーバー上のデータベースを識別するようにそれぞれのノードとデータベースのカタログを作成する。
   d. アプリケーション・プログラムをプリコンパイルし、タイプ 2 接続を指定して、 (つまり PRECOMPILE PROGRAM コマンドで CONNECT 2 を指定) 2 フェーズ・コミットを指定 (つまり、 PRECOMPILE PROGRAM コマンドで SYNCPOINT TWOPHASE を指定) する。
   e. DB2 UDB トランザクション・マネージャー (TM) を構成する。
2. XA 準拠のトランザクション・マネージャーを使用する方法:
   a. 該当するデータベースの中に必要な表を作成する。
   b. 物理的にリモートの場合、適切な通信プロトコルを使用するようにそれぞれのデータベース・サーバーを設定する。
   c. 物理的にリモートの場合、データベース・サーバー上のデータベースを識別するようにそれぞれのノードとデータベースのカタログを作成する。

d. アプリケーション・プログラムをプリコンパイルし、タイプ 2 接続を指定 (つまり PRECOMPILE PROGRAM コマンドで CONNECT 2 を指定) して、1 フェーズ・コミットを指定 (つまり、 PRECOMPILE PROGRAM コマンドで SYNCPOINT ONEPHASE を指定) する。

e. DB2 UDB データベースを使用するように XA 準拠のトランザクション・マネージャーを構成する。

関連概念:

- 30 ページの『作業単位』
- 201 ページの『DB2 トランザクション・マネージャー』

関連タスク:

- 197 ページの『1 つのトランザクションでの単一のデータベースの更新』
- 198 ページの『複数のデータベース・トランザクションでの単一データベースの更新』

関連資料:

- 「*コマンド・リファレンス*」の『PRECOMPILE コマンド』

DB2 トランザクション・マネージャー

DB2® Universal Database (DB2 UDB) トランザクション・マネージャー (TM) は、トランザクションに ID を割り当て、進行状況を監視し、トランザクションの完了と障害時の処理を実行します。 DB2 UDB および DB2 Connect™ は、トランザクション・マネージャーを提供します。 DB2 UDB TM は、指定された TM データベース内にトランザクション情報を保管します。

データベース・マネージャーには、単一の作業単位内で複数のデータベースを更新する作業を調整するために使用できる、トランザクション・マネージャー機能があります。データベース・クライアントは作業単位を自動的に調整し、トランザクション・マネージャー・データベース を使用して、それぞれのトランザクションを登録し、その完了状況を記録します。

DB2 UDB データベースを使用して DB2 UDB トランザクション・マネージャーを使用することができます。 2 フェーズ・コミット・トランザクションに参加させたいリソースが DB2 UDB 以外のデータベースである場合、 XA 準拠のトランザクション・マネージャーを使用することができます。

関連概念:

- 30 ページの『作業単位』
- 202 ページの『DB2 Universal Database トランザクション・マネージャーの構成』
- 205 ページの『2 フェーズ・コミット』

DB2 Universal Database トランザクション・マネージャーの構成

IBM® WebSphere®、BEA Tuxedo、または Microsoft® Transaction Server などの XA 準拠のトランザクション・マネージャーを使用している場合は、その製品の構成に関する説明に従ってください。

UNIX® ベースのシステム用の DB2® Universal Database (DB2 UDB) または Windows® オペレーティング・システムを使ってトランザクションを調整する場合は、いくつかの構成上の要件を満たす必要があります。通信に TCP/IP だけを使用し、トランザクションに組み込まれているデータベース・サーバーが DB2 UDB for UNIX、DB2 UDB for Windows、 DB2 UDB for iSeries™ V5、 DB2 UDB for z/OS™、または DB2 UDB for OS/390® だけである場合は、単純な構成を使用できます。

DB2 Connect™ では、ホストまたは iSeries サーバーへの SNA 2 フェーズ・コミット・アクセスがサポートされなくなりました。

TCP/IP 接続を使用した DB2 Universal Database for UNIX および Windows、DB2 for z/OS、DB2 for OS/390、および DB2 for iSeries V5

以下のすべてが当てはまる環境では、マルチサイト更新のための構成ステップは単純です。

- リモート・データベース・サーバー (DB2 UDB for z/OS、DB2 UDB for OS/390、 DB2 UDB for iSeries V5 を含む) とのすべての通信で TCP/IP だけが使用される。
- トランザクションに組み込まれているデータベース・サーバーは、 DB2 UDB for UNIX 基本システム、Windows オペレーティング・システム、 DB2 UDB for z/OS、DB2 UDB for OS/390、または DB2 UDB for iSeries V5 のみである。

トランザクション・マネージャー・データベースとして使用されるデータベースは、データベース構成パラメーター *tm_database* によって、データベース・クライアントで決められます。このパラメーターを設定するときは、以下の要素を考慮に入れてください。

- トランザクション・マネージャー・データベースには、以下のデータベースがあります。
  - DB2 UDB for UNIX または DB2 UDB for Windows バージョン 8 データベース
  - DB2 for z/OS and OS/390 バージョン 7 データベースまたは DB2 for OS/390 バージョン 5 または 6 データベース
  - DB2 for iSeries V5 データベース

    DB2 for z/OS、DB2 for OS/390、および DB2 for iSeries V5 は、トランザクション・マネージャー・データベースとして使用するための、推奨されたデータベース・サーバーです。一般的に、z/OS システム、OS/390 システム、および iSeries V5 システムはワークステーション・サーバーよりも安全で、偶発的

な電源遮断、リブート、などの可能性が低くなっています。したがって、再同期時に使われるリカバリー・ログは、より確かなものとなります。

- 構成パラメーター *tm_database* に値 `1ST_CONN` が指定されている場合、アプリケーションが最初に接続したデータベースが、トランザクション・マネージャー・データベースとして使用されます。

  `1ST_CONN` を使用するときには注意が必要です。この構成を使用するのは、参加するすべてのデータベースのカタログを適切に作成するのが容易な場合だけ、つまり、トランザクションを開始したデータベース・クライアントが、参加データベース (トランザクション・マネージャー・データベースを含む) があるのと同じインスタンス内にある場合だけにしてください。

トランザクション・マネージャー・データベースとして使用されているデータベースからの切断をアプリケーションが試みると、警告メッセージを受け取り、作業単位がコミットされるまで接続はそのまま保持されることに注意してください。

構成パラメーター

環境を設定する場合は、次の構成パラメーターを考慮してください。

データベース・マネージャー構成パラメーター

- *tm_database*

  このパラメーターは、各 DB2 UDB インスタンスのトランザクション・マネージャー (TM) データベースの名前を識別します。

- *spm_name*

  このパラメーターは、データベース・マネージャーに DB2 Connect 同期点マネージャーのインスタンス名を知らせます。再同期が正常に実行されるには、この名前はネットワーク全体でユニークなものでなければなりません。

- *resync_interval*

  このパラメーターは、DB2 トランザクション・マネージャー、 DB2 UDB サーバー・データベース・マネージャー、および DB2 Connect (または DB2 UDB) 同期点マネージャーが、未解決の未確定トランザクションのリカバリーを試みる時間間隔を秒数で指定します。

- *spm_log_file_sz*

  このパラメーターは、SPM ログ・ファイルのサイズを 4 KB ページ数の単位で指定します。

- *spm_max_resync*

  このパラメーターは、再同期操作を同時に実行することができるエージェント数を指定します。

- *spm_log_path*

  このパラメーターは、SPM ログ・ファイルのログ・パスを識別します。

データベース構成パラメーター

- *maxappls*

このパラメーターは、アクティブなアプリケーションの許容最大数を指定します。この値は、接続されるアプリケーションの合計数に、2 フェーズ・コミットまたはロールバックを同時に完了しようとしている可能性のあるアプリケーションの数と、同時に存在することが予想される未確定トランザクションの数を加えた値より大きいか等しくなければなりません。

- *autorestart*

  このデータベース構成パラメーターには、必要に応じて RESTART DATABASE ルーチンを自動的に呼び出すかどうかを指定します。デフォルト値は YES (呼び出す) です。

  未確定トランザクションが含まれているデータベースは、データベースの再始動操作によって始動する必要があります。データベースへの最後の接続がドロップされるときに *autorestart* が使用可能でない場合、その次の接続は失敗し、再び RESTART DATABASE を明示的に呼び出す必要があります。この状態は、トランザクション・マネージャーの再同期操作によって、または管理者が開始したヒューリスティックな操作によって、未確定トランザクションが除去されるまで続きます。 RESTART DATABASE コマンドが発行されるとき、データベースに未確定トランザクションが存在していれば、メッセージが戻されます。管理者は、LIST INDOUBT TRANSACTIONS コマンドなどのコマンド行プロセッサー (CLP) のコマンドを使用することによって、それらの未確定トランザクションについての情報を検索できます。

関連概念:

- 201 ページの『DB2 トランザクション・マネージャー』

関連タスク:

- 234 ページの『IBM TXSeries CICS の構成』
- 235 ページの『IBM TXSeries Encina の構成』
- 237 ページの『BEA Tuxedo の構成』
- 234 ページの『IBM WebSphere Application Server の構成』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『spm_log_path - 「同期点マネージャー・ログ・ファイル・パス」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『autorestart - 「自動再始動使用可能」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『maxappls - 「アクティブ・アプリケーションの最大数」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『resync_interval - 「トランザクション再同期インターバル」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『tm_database - 「トランザクション・マネージャー・データベース名」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『spm_name - 「同期点マネージャー名」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『spm_log_file_sz - 「同期点マネージャー・ログ・ファイル・サイズ」構成パラメーター』

- 「*管理ガイド: パフォーマンス*」の『spm_max_resync - 「同期点マネージャー再同期エージェント数の限度」構成パラメーター』

ホストまたは iSeries クライアントからのデータベースの更新

ホストまたは iSeries 上で実行されているアプリケーションは、 DB2 Universal Database™ (DB2 UDB) データベース・サーバー上にあるデータにアクセスできます。このアクセスで使用されるプロトコルは TCP/IP だけです。どのプラットフォームの DB2 UDB サーバーにおいても、リモート・クライアントからの SNA アクセスはサポートされなくなりました。

バージョン 8 より前は、ホストまたは iSeries クライアントからの TCP/IP アクセスでは、 1 フェーズ・コミット・アクセスだけがサポートされていました。現在では、DB2 UDB において、ホストまたは iSeries クライアントから TCP/IP 2 フェーズ・コミット・アクセスが可能になりました。 TCP/IP 2 フェーズ・コミット・アクセス使用時に Syncpoint Manager (SPM) を使用する必要はありません。

ホストまたは iSeries クライアントからアクセスするには、サーバー上で DB2 UDB TCP/IP listener がアクティブでなければなりません。 TCP/IP Listener がアクティブであることを確認するには、 db2set コマンドを使用して、レジストリー変数 DB2COMM の値が “tcpip” であることを確認します。また、GET DBM CFG コマンドを使用することによって、データベース・マネージャーの構成パラメーター **svcename** がサービス名に設定されていることを確認できます。 listener がアクティブでない場合は、 db2set コマンドと UPDATE DBM CFG コマンドを使用することによってアクティブにすることができます。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『spm_name - 「同期点マネージャー名」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『通信変数』

2 フェーズ・コミット

206 ページの図 61 はマルチサイト更新に関連したステップを示しています。トランザクションが管理される仕方を理解しておけば、 2 フェーズ・コミット処理中にエラーが発生した場合に問題の解決に役立ちます。

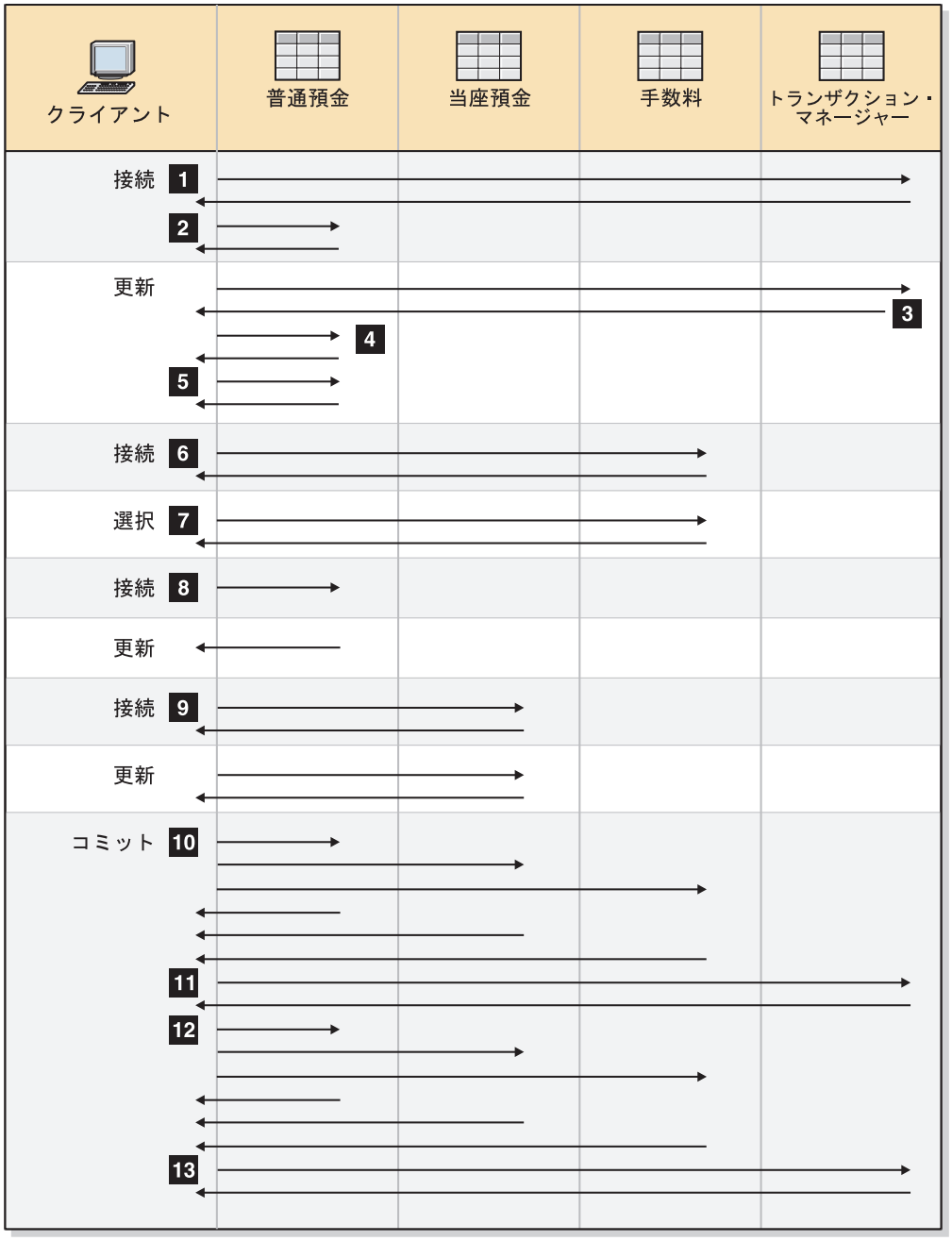

図 61. 複数のデータベースの更新

0 アプリケーションは 2 フェーズ・コミットに備えて準備済みになります。これはプリコンパイル・オプションを使用して行われます。また、DB2® Universal Database (DB2 UDB) CLI (コール・レベル・インターフェース) 構成を使用してこれを行うこともできます。

1 データベース・クライアントがデータベース SAVINGS_DB に接続しようとするとき、まず内部でトランザクション・マネージャー (TM) データベースに接続します。 TM データベースは、データベース・クライアントに対し肯定応答を戻します。データベース・マネージャー構成パラメーター

tm_database が 1ST_CONN に設定されていれば、このアプリケーション・インスタンスの存続期間中は SAVINGS_DB がトランザクション・マネージャーになります。

2 データベース SAVINGS_DB に対する接続が行われ、肯定応答が受信されます。

3 データベース・クライアントは SAVINGS_ACCOUNT 表の更新を開始します。これにより、作業単位が始まります。 TM データベースはデータベース・クライアントに対し応答し、作業単位のトランザクション ID を送信します。作業単位が登録されるのは、接続の確立時ではなく、作業単位内の最初の SQL ステートメントの実行時であることに注意してください。

4 データベース・クライアントは、トランザクション ID を受信すると、この作業単位を SAVINGS_ACCOUNT 表を含むデータベースに登録します。作業単位登録が正常完了したことを通知する応答がクライアントに送り返されます。

5 データベース SAVINGS_DB に対し出された SQL ステートメントは、通常の方法で処理されます。各ステートメントに対する応答は、プログラムに組み込まれている SQL ステートメントの処理時に SQLCA に戻されます。

6 作業単位中で最初にそのデータベースにアクセスするとき、TRANSACTION_FEE 表が入っている FEE_DB データベースにトランザクション ID が登録されます。

7 FEE_DB データベースに対するすべての SQL ステートメントが通常の方法で処理されます。

8 接続を適切に設定することによって、データベース SAVINGS_DB に対してさらに SQL ステートメントを実行できます。作業単位はすでにデータベース SAVINGS_DB に登録されているため **4**、データベース・クライアントは再び登録ステップを実行する必要はありません。

9 データベース CHECKING_DB に接続してそれを使用する方法は、 **6** および **7** と同じ規則に従います。

10 データベース・クライアントからこの作業単位をコミットするよう要求が出されると、この作業単位に関係しているすべてのデータベースに対して*準備*メッセージが送信されます。各データベースは ″PREPARED″ レコードをログ・ファイルに書き込み、データベース・クライアントに対して応答を戻します。

11 すべてのデータベースから肯定応答を受信すると、データベース・クライアントはトランザクション・マネージャー・データベースにメッセージを送信して、作業単位のコミット準備が完了した (PREPARED) ことを通知します。トランザクション・マネージャー・データベースは、 ″PREPARED″ レコードをそのログ・ファイルに書き込み、クライアントに応答メッセージを送信して、コミット・プロセスの第 2 フェーズが開始可能になったことを通知します。

12 コミット・プロセスの第 2 フェーズ実行時に、データベース・クライアントはすべての参加データベースに、コミットするよう依頼するメッセージを送信します。それぞれのデータベースが ″COMMITTED″ レコードをそのロ

グ・ファイルに書き込み、この作業単位に保持していたロックを解放します。データベースが変更内容のコミットを完了すると、クライアントに応答を送信します。

13 すべての参加データベースから肯定応答を受信すると、データベース・クライアントはトランザクション・マネージャー・データベースに対して、作業単位が完了したことを通知するメッセージを送ります。その後、トランザクション・マネージャー・データベースは作業単位が完了したことを示す ″COMMITTED″ レコードをログ・ファイルに記録し、クライアントに対して作業単位が完了したという応答を送ります。

関連概念:

- 30 ページの『作業単位』
- 201 ページの『DB2 トランザクション・マネージャー』

2 フェーズ・コミット中のエラー・リカバリー

エラー条件からのリカバリーは、アプリケーション・プログラミング、システム管理、データベース管理、およびシステム操作において通常に行われる作業です。データベースを複数のリモート・サーバーに分散させると、ネットワークや通信の障害のためにエラーの発生する可能性が高くなります。データ保全性を確保するため、データベース・マネージャーは 2 フェーズ・コミット・プロセスを提供しています。以下では、2 フェーズ・コミット・プロセス中のエラーをデータベース・マネージャーが処理する方法を説明します。

- **第 1 フェーズでのエラー**

  作業単位のコミット準備に失敗したことをいずれかのデータベースが伝えた場合、データベース・クライアントはコミット・プロセスの第 2 フェーズで作業単位をロールバックします。この場合、準備メッセージはトランザクション・マネージャー・データベースに送信されません。

  第 2 フェーズ中に、クライアントは、第 1 フェーズでのコミットが正常に作成されたすべての参加データベースに、ロールバック・メッセージを送信します。それぞれのデータベースは、″ABORT″ レコードをログ・ファイルに書き込み、この作業単位のために保持していたロックを解放します。

- **第 2 フェーズでのエラー**

  この段階でのエラー処理は、第 2 フェーズでトランザクションがコミットされるか、それともロールバックされるかによって違います。最初のフェーズでエラーが検出された場合、第 2 フェーズではトランザクションのロールバックしか実行されません。

  参加データベースのうちの 1 つが (おそらく通信障害が原因で) 作業単位のコミットに失敗すると、トランザクション・マネージャー・データベースによって、失敗したデータベースでのコミットが再試行されます。しかし、アプリケーションに対しては、コミットが成功したと SQLCA を通して通知されます。 DB2® Universal Database (DB2 UDB) では、データベース・サーバー内のコミットされなかったトランザクションは、確実にコミットされます。トランザクション・マネージャー・データベースが作業単位のコミットを試行してから次にコミットを

試行するまでの待機インターバルは、データベース・マネージャー構成パラメーター *resync_interval* によって指定されます。すべてのロックは、作業単位がコミットされるまでデータベース・サーバーにおいて保持されます。

トランザクション・マネージャー・データベースに障害が起こった場合は、データベースの再開時に作業単位が再同期化されます。再同期化プロセスでは、*未確定* トランザクション、つまり第 1 フェーズは完了したが第 2 コミット・フェーズは完了していないトランザクションをすべて完了することが試行されます。トランザクション・マネージャー・データベースに関連したデータベース・マネージャーは、以下のようにして再同期化を実行します。

1. コミット・プロセスの第 1 フェーズ中にコミットの "PREPARED" を示したすべてのデータベースに接続する。
2. それらのデータベースで、未確定トランザクションのコミットを試行する。(未確定トランザクションが見つからない場合、データベース・マネージャーは、コミット・プロセスの第 2 フェーズでデータベースがトランザクションを正常にコミットしたものと見なします。)
3. 参加データベース内ですべての未確定トランザクションがコミットされたら、トランザクション・マネージャー・データベース内の未確定トランザクションをコミットする。

参加データベースのうちの 1 つに障害が起こって再始動した場合、そのデータベースのデータベース・マネージャーはトランザクション・マネージャー・データベースに対してこのトランザクションの状況を照会して、トランザクションをロールバックするべきかどうかを判別します。ログにトランザクションがない場合、データベース・マネージャーはトランザクションがロールバックされたと見なして、このデータベース内の未確定トランザクションをロールバックします。ログにトランザクションがあれば、データベースはトランザクション・マネージャー・データベースからのコミット要求を待ちます。

トランザクション処理モニター (XA 準拠のトランザクション・マネージャー) によってトランザクションが調整された場合は、データベースは常に TP モニターによって再同期を開始します。

何らかの理由で、トランザクション・マネージャーが自動的に未確定トランザクションを解決するまで待てない場合は、未確定トランザクションを手動で解決するための措置がいくつかあります。この手動の処理は、「ヒューリスティック判定」と呼ばれることもあります。

autorestart=off の場合のエラー・リカバリー

データベース構成パラメーター *autorestart* が `OFF` に設定されていて、 TM データベースまたは RM データベースのいずれかに未確定トランザクションがある場合、再同期プロセスを始動するには RESTART DATABASE コマンドが必要です。コマンド行プロセッサーから RESTART DATABASE コマンドを発行する時には、別のセッションを使用してください。同じセッションから別のデータベースを再始動すると、前の呼び出しで確立された接続はドロップされ、もう一度再始動する必要があります。 LIST INDOUBT TRANSACTIONS コマンドによって戻される未確定トランザクションがなくなったら、 TERMINATE コマンドを発行して接続をドロップします。

関連概念:

- 205 ページの『2 フェーズ・コミット』

関連タスク:

- 226 ページの『未確定トランザクションの手動での解決』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『autorestart - 「自動再始動使用可能」構成パラメーター』
- 「コマンド・リファレンス」の『LIST INDOUBT TRANSACTIONS コマンド』
- 「コマンド・リファレンス」の『TERMINATE コマンド』
- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』

第 7 章 XA 準拠のトランザクション・マネージャー用の設計

2 フェーズ・コミット・トランザクションに参加させたいリソースが DB2 以外のデータベースである場合、 XA 準拠のトランザクション・マネージャーを用いてそのデータベースを使用することもできます。ご使用のトランザクションでは DB2 データベースにしかアクセスできない場合には、 199 ページの『トランザクションで複数のデータベースを更新する』に説明されている DB2 トランザクション・マネージャーを使用してください。

以下のトピックは、 IBM WebSphere や BEA Tuxedo などの XA 準拠のトランザクション・マネージャーを用いてデータベース・マネージャーを使用する上で役立ちます。

Microsoft Transaction Server の詳細については、「*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*」を参照してください。

XA 準拠のトランザクション・マネージャーをすでに使用している場合、またはこれからインプリメントする場合には、次の技術サポート Web サイトで詳しい情報を参照できます。

`http://www.ibm.com/software/data/db2/udb/winos2unix/support`

この Web サイトから「DB2 Universal Database」を選択します。次にキーワード ″XA″ を使用して Web サイト内を探索し、 XA 準拠のトランザクション・マネージャーの最新情報を入手してください。

X/Open 分散トランザクション処理のモデル

X/Open 分散トランザクション処理 (DTP) のモデルには、次のような互いに関連する 3 つのコンポーネントがあります。

- アプリケーション・プログラム (AP)
- トランザクション・マネージャー (TM)
- リソース・マネージャー (RM)

図 62 は、このモデルと 3 つのコンポーネントの相互関係を示しています。

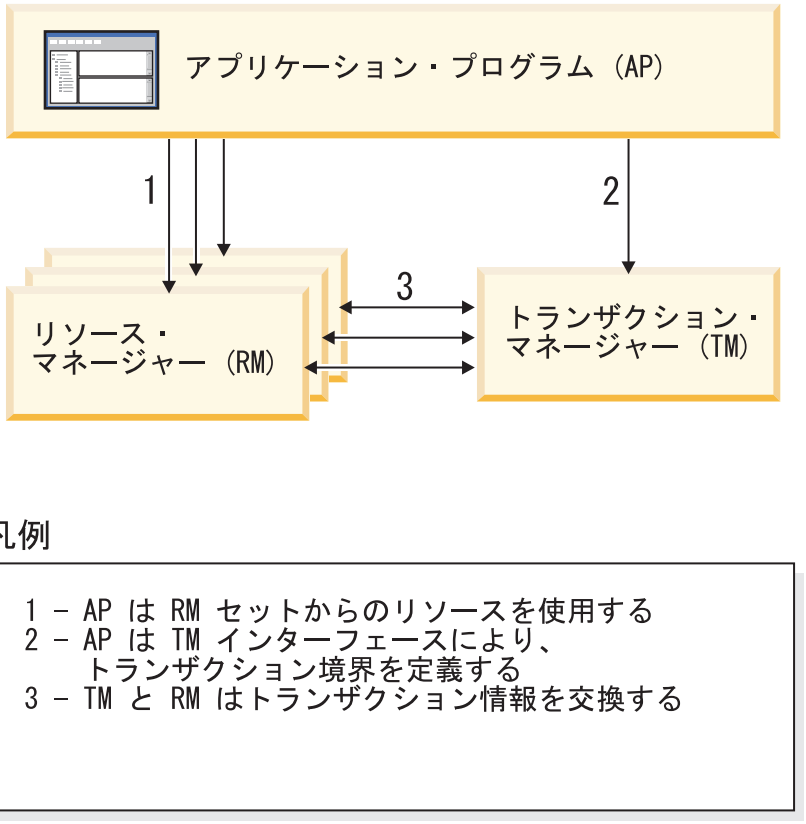

図 62. X/Open 分散トランザクション処理 (DTP) モデル

アプリケーション・プログラム (AP)

アプリケーション・プログラム (AP) は、トランザクション境界を定義し、トランザクションを構成するアプリケーション固有のアクションを定義します。

たとえば、CICS®* アプリケーション・プログラムでデータベースや CICS 一時データ・キューなどのリソース・マネージャー (RM) にアクセスし、プログラミング・ロジックを使用してデータを操作できます。それぞれのアクセス要求は、その RM に固有の関数呼び出しによって、適切なリソース・マネージャーに渡されます。 DB2 ® Universal Database (DB2 UDB) の場合、このような呼び出しは、各

SQL ステートメントごとに DB2 UDB プリコンパイラーが生成した関数呼び出し、または、プログラマーが API を使用して直接コーディングしたデータベース呼び出しとすることができます。

トランザクション・マネージャー (TM) 製品には、通常、ユーザーのアプリケーションを実行するためのトランザクション処理 (TP) モニターが含まれています。 TP モニターには、アプリケーションがトランザクションを開始および終了したり、アプリケーションのスケジューリングを実行したり、そのアプリケーションを実行する多くのユーザーの間で負荷バランス調整を実行するための API が用意されています。分散トランザクション処理 (DTP) 環境のアプリケーション・プログラムは、実際にはユーザー・アプリケーションと TP モニターの組み合わせです。

効率的なオンライン・トランザクション処理 (OLTP) 環境を容易にするため、 TP モニターは起動時に複数のサーバー・プロセスを事前に割り振り、スケジューリングを実行して、多数のユーザー・トランザクション間でこれらのプロセスを再使用します。これによって、より少ない数のサーバー・プロセスおよびそれらに対応する RM プロセスを使ってより多くの並行ユーザーをサポートすることが可能になり、システム・リソースの節約になります。これらのプロセスを再使用すれば、ユーザー・トランザクションまたはプログラムごとに、 TM と RM でのプロセスを起動する場合のオーバーヘッドを回避することもできます。 (1 つのプログラムで 1 つまたは複数のトランザクションが呼び出されます。) このことは、TM および RM にとってはこれらのサーバー・プロセスが実際の「ユーザー・プロセス」になるということも意味します。このことは、セキュリティー管理やアプリケーション・プログラミングにも関係します。

TP モニターからは、以下のタイプのトランザクションが可能です。

- XA 以外のトランザクション

  これらのトランザクションには、TM に対して定義されていない RM が関係しているため、 TM の 2 フェーズ・コミット・プロトコルの下では調整されません。アプリケーションで XA インターフェースをサポートしていない RM にアクセスする必要がある場合は、この調整が必要になります。 TP モニターは、単にアプリケーションの効率的なスケジューリングと負荷バランス調整を提供するだけです。 TM は XA 処理のために RM を明示的に「オープン」することはないため、 RM はこのアプリケーションを、非 DTP 環境で実行される他のアプリケーションと同じようにして処理します。

- グローバル・トランザクション

  これらのトランザクションは、TM に対して定義されている RM が関係しているため、 TM の 2 フェーズ・コミットによって制御されます。グローバル・トランザクションとは、1 つまたは複数の RM が関係する作業単位のことです。トランザクション・ブランチ とは、 TM と RM との間のグローバル・トランザクションをサポートする部分のことです。 TM によって調整されるアプリケーション・プロセスが複数の RM にアクセスする場合は、 1 つのグローバル・トランザクションに複数のトランザクション・ブランチが存在します。

  個々のアプリケーション・プロセスが、TM の調整下にありながら、あたかも別々のグローバル・トランザクションに属しているかのように複数の RM にアクセスする場合は、疎結合のグローバル・トランザクションが存在しています。個々

のアプリケーション・プロセスごとに、 RM 内にそれぞれ固有のトランザクション・ブランチがあります。いずれかの AP、TM、または RM によりコミットまたはロールバックが要求されると、トランザクション・ブランチはすべて完了します。分岐間でリソース・デッドロックが発生しないようにするのは、アプリケーション側の責任です。 (SYNCPOINT(TWOPHASE) オプションを指定して作成されたアプリケーションに対して DB2 UDB トランザクション・マネージャーが実行するトランザクション調整は、大まかにいってこの疎結合のグローバル・トランザクションと同等であることに注意してください。)

複数のアプリケーション・プロセスが RM 内の同じトランザクション・ブランチの下で作業を分担している場合は、密結合グローバル・トランザクションが存在しています。これら 2 つのアプリケーション・プロセスは、RM からは単一のエンティティーと見なされます。 RM では、トランザクション・ブランチの中でリソースのデッドロックが発生しないようにする必要があります。

トランザクション・マネージャー (TM)

トランザクション・マネージャー (TM) は、トランザクションに ID を割り当て、進行状況を監視し、トランザクションの完了と障害時の処理を実行します。トランザクション・ブランチ ID (XID と呼ばれるもの) は TM によって割り当てられ、グローバル・トランザクションと RM 内部の固有の分岐の両方を識別するものとなります。これは、TM のログと RM のログの間の相関トークンです。 XID は、2 フェーズ・コミットまたはロールバックを行う場合、システム始動時の*再同期化* 操作 (*resync* ともいう) を行う場合、または、必要に応じて、管理者がヒューリスティック な操作 (*手動介入* ともいう) を実行する場合に必要です。

TP モニターを始動すると、 TP モニターは一連のアプリケーション・サーバーによって定義されているすべての RM をオープンするよう TM に要請します。 TM は RM に対して **xa_open** 呼び出しを渡し、 RM が DTP 処理のために初期設定されるようにします。 TM は、この始動手続き中に再同期化を実行し、すべての*未確定トランザクション* をリカバリーします。未確定トランザクションとは、不確かな状態のままになっているグローバル・トランザクションのことです。これが発生するのは、 2 フェーズ・コミット・プロトコルの最初のフェーズ (つまり準備フェーズ) が正常完了した後に、 TM (または少なくとも 1 つの RM) が使用不能になるときです。 RM のログが再度使用可能になって TM が自身のログと RM のログとを整理調整するまで、RM はトランザクションの分岐に対してコミットとロールバックのどちらを実行すればよいのかを識別できません。再同期操作を実行するため、 TM は個々の RM に対して **xa_recover** 呼び出しを 1 回または複数回発行して、すべての未確定トランザクションを識別します。 TM は、それらの応答と自身のログ情報とを比較して、トランザクションに関して **xa_commit** と **xa_rollback** のどちらを実行するよう RM に通知するべきかを判断します。管理者のヒューリスティック操作により、 RM が未確定トランザクションの分岐をすでにコミットまたはロールバックしていた場合、 TM はその RM に対して **xa_forget** 呼び出しを発行して、再同期操作を完了します。

ユーザー・アプリケーションからコミットまたはロールバック要求を出すときは、関係するすべての RM 間のコミットまたはロールバックの調整を TM が行えるようにするため、 TP モニターまたは TM で提供されている API を使用する必要があります。たとえば、 CICS アプリケーションが CICS SYNCPOINT 要求を発行してトランザクションをコミットすると、今度は CICS XA の TM (Encina® Server

に実装されている) が、 **xa_end**、 **xa_prepare**、 **xa_commit**、または **xa_rollback** などの XA 呼び出しを発行して、トランザクションをコミットまたはロールバックするよう RM に要求します。 RM が 1 つしか関係していない場合、または分岐が読み取り専用であるという応答が RM から返ってきた場合には、 TM は 2 フェーズ・コミットではなく 1 フェーズ・コミットを使用できます。

リソース・マネージャー (RM)

リソース・マネージャー (RM) は、データベースなどの共有リソースへのアクセスを提供するものです。

DB2 UDB は、データベースのリソース・マネージャーとして、 XA 準拠の TM によって調整されているグローバル・トランザクション に参加できます。 XA インターフェースとして必要とされるものとして、db2xa_switch が用意されています。これは、データベース・マネージャーに、 XA スイッチ構造体を TM に戻すために使う *xa_switch_t* 型の外部 C 変数です。このデータ構造体には、 TM が呼び出すさまざまな XA ルーチンのアドレスと RM の操作特性とが入れられます。

RM が個々のグローバル・トランザクションへの参加を登録する方法には、*静的登録* と*動的登録* の 2 つがあります。

- 静的登録の場合、特定の RM がトランザクションで使用中かどうかに関係なく、サーバー・アプリケーションに定義されているすべての RM に対して、 TM は **xa_start**、 **xa_end**、および **xa_prepare** の一連の呼び出しを (各トランザクションごとに) 発行する必要があります。すべての RM がすべてのトランザクションに関係しているわけではない場合、これは非効率であり、定義されている RM の数に比例して、効率は低下します。
- 動的登録 (DB2 UDB で使用される) は、柔軟で効率の良いものです。 RM は、リソース要求を受信した場合に限り、 **ax_reg** を使用して TM に登録します。この方法だと、RM が 1 つしか定義されていない場合、またはすべての RM がすべてのトランザクションで使用されている場合であっても、 TM での **ax_reg** 呼び出しと **xa_start** 呼び出しのパスが類似しているため、パフォーマンス上不利な点はありません。

XA インターフェースでは、TM と RM との間の双方向通信が提供されます。これは、2 つの DTP ソフトウェア・コンポーネントの間のシステム・レベルのインターフェースであり、アプリケーション開発者がコーディングする普通のアプリケーション・プログラム・インターフェースではありません。ただし、アプリケーション開発者は、 DTP ソフトウェア・コンポーネントに関連したプログラミング上の制限事項に通じている必要があります。

XA インターフェースは一定ですが、XA 準拠の各 TM では、 RM が製品固有の方法で組み込まれている場合があります。ご使用の DB2 UDB 製品をリソース・マネージャーとして特定のトランザクション・マネージャーに組み込む方法については、該当する TM 製品の資料を参照してください。

関連概念:

- 229 ページの『XA トランザクション・マネージャーのセキュリティーに関する考慮事項』
- 231 ページの『DB2 Universal Database によってサポートされる XA 機能』

- 「アプリケーション開発ガイド クライアント・アプリケーションのプログラミング」の『X/Open XA インターフェース・プログラミングに関する考慮事項』

関連タスク:

- 199 ページの『トランザクションで複数のデータベースを更新する』

リソース・マネージャーのセットアップ

各データベースは、トランザクション・マネージャー (TM) に対して別個のリソース・マネージャー (RM) として定義されているので、 xa_open ストリングによって識別する必要があります。

データベースをリソース・マネージャーとして設定する場合、 xa_close ストリングは必要ありません。このストリングを指定しても、データベース・マネージャーによって無視されます。

データベース接続に関する考慮事項

自動クライアント・リルート (ACR)

サーバーがクラッシュすると、そのサーバーに接続している各クライアントでは通信エラーが発生して接続が終了し、アプリケーション・エラーになります。可用性が重視されるアプリケーション環境では、ユーザーは余分なセットアップをするか、サーバーをスタンバイ・ノードにフェイルオーバーすることになります。いずれにしても、DB2® Universal Database (DB2 UDB) のクライアント・コードは、元のデータベース (IP アドレスもフェイルオーバーしているフェイルオーバー・ノードで実行されている場合がある) か、または異なるサーバー上の新しいデータベースのいずれかとの接続の再確立を試みます。次にアプリケーションに対して、接続が転送され、実行中だった特定のトランザクションがロールバックされたことが SQLCODE を介して通知されます。その時点でアプリケーションは、そのトランザクションを再実行するか、それともそのまま継続するかを選択できます。

ACR を使用した場合、障害の発生した主データベースと、「フェイル先」スタンバイ・データベースの間のデータの整合性は、接続の転送先のデータベースのデータベース・ログの状態に大きく依存します。説明の便宜上、このデータベースを「スタンバイ・データベース」と呼び、そのスタンバイ・データベースの存在するサーバーを「スタンバイ・サーバー」と呼ぶことにします。スタンバイ・データベースが、障害の発生した主データベースの障害発生時点での正確なコピーであるなら、スタンバイ・データベースのデータは整合性があり、データの整合性に関して問題はありません。しかし、スタンバイ・データベースが、障害の発生した主データベースの正確なコピーでない場合には、 XA トランザクション・マネージャーによって準備されているがまだコミットされていないトランザクションのトランザクション出力が矛盾しているため、データ整合性に関して問題が発生している可能性があります。それらは未確定トランザクションと呼ばれます。 ACR 機能を使用するデータベース管理者とアプリケーション開発者は、その機能を使用する場合にデータ整合性の問題の危険性があることを認識しておく必要があります。

以下のセクションでは、さまざまな DB2 UDB データベース環境と、そのそれぞれにおけるデータ整合性の問題の危険性について説明します。

高可用性災害時リカバリー (HADR):

DB2 UDB の高可用性災害時リカバリー・フィーチャー (HADR) は、主データベースの障害の後、アプリケーションが接続を再び獲得した時点で、 1 次 (主) データベースと 2 次データベースの間のログ複写のレベルを制御するために使用できます。ログ複写のレベルを制御するデータベース構成パラメーターは、*hadr_syncmode* です。以下に、このパラメーターに可能な 3 つの値を示します。

- SYNC

  このモードでは、トランザクションの損失からの最大の保護が提供されますが、トランザクションの応答時間は 3 つのモードの中で最も長くなります。このモードの名前が示すように、SYNC は、主データベースとスタンバイ・データベースのトランザクション・ログの書き込みを同期化するために使用されます。同期は、主データベースがそのログ・ファイルを書き込んだ後、スタンバイ・データベースにもそれらのログが書き込まれたという確認通知をスタンバイ・データベースから受け取った時点で達成されます。

  DB2 UDB リソースの関係したトランザクションを調整するために XA トランザクション・マネージャーが使用されている場合には、 SYNC モードを使用することをお勧めします。 2 次データベースは主 (1 次) データベースの正確なレプリカなので、クライアントが 2 次データベースに転送される場合には、 SYNC モードを使用することにより、データの整合性とトランザクションの再同期整合性が保証されます。

- NEARSYNC

  このモードでは、 SYNC モードと比較した場合にトランザクションの応答時間が短くなる代わりに、トランザクションの損失に関する保護がやや少なくなります。主データベースでログ書き込みが成功したと見なされるのは、そのログ・ファイルにログが書き込まれ、かつスタンバイ・データベースのメイン・メモリーにログが書き込まれたことを示す確認通知を 2 次データベースから受け取った場合だけです。ログをメモリーからディスクにコピーする前にスタンバイ・データベースがクラッシュした場合、短期間のうちにスタンバイ・データベース上のログは失われます。

  データベース・ログが失われる可能性があり、スタンバイ・データベースが主データベースの正確なレプリカではないという状況では、データの整合性を犠牲にすることで妥協するという可能性があります。そのような妥協は、特定のトランザクションが未確定である時点で主データベースがクラッシュした場合に発生します。そのトランザクションの結果が COMMIT だとしましょう。 XA TM がそれ以降に XA_COMMIT 要求を発行すると、主データベースがクラッシュしているため、それは失敗します。 XA_COMMIT 要求が失敗すると、 XA TM は、XA_RECOVER 要求を発行することによって、このデータベース上でそのトランザクションをリカバリーする必要があります。スタンバイ・データベースは、そのトランザクションすべてのうち INDOUBT であるもののリストを戻すことによって応答します。『メモリー内の』 データベース・ログがディスクに書き込まれる前、かつ XA_RECOVER 要求が XA TM から発行される前にスタンバイ・データベースがクラッシュおよび再始動することになると、スタンバイ・データベースでは、そのトランザクションに関するログ情報が失われ、 XA_RECOVER 要求への応答でそれを戻すことができなくなります。その場合、XA TM はデータ

ベースがそのトランザクションをコミットしたものと仮定します。しかし、実際にはデータ操作は失われて、トランザクションがロールバックされたかのように見えることになります。その結果、そのトランザクションに関係するその他のすべてのリソースは XA TM によって COMMIT されたため、データ整合性の問題が発生することになります。

NEARSYNC を使用することは、データ整合性とトランザクションの応答時間の間での適切な妥協点といえます。というのは、主データベースとスタンバイ・データベースの両方がクラッシュする可能性は低いはずだからです。それでもデータベース管理者は、データ整合性の問題が発生する可能性について理解しておく必要があります。

- ASYNC

  このモードでは、 3 つのモードの中でトランザクションの応答時間が最短になる代わりに、主データベースで障害が発生した場合にトランザクションの損失が発生する可能性が最大になります。主データベースでログの書き込みが成功したと見なされるのは、そのログ・ファイルにログが書き込まれ、かつ主データベースのホスト・マシン上の TCP 層にログが配信された場合だけです。主データベースは、スタンバイ・データベースからの何らかの確認通知を待機することはしません。関連するトランザクションがコミットされたと主データベースが見なした時点でも、ログはまだスタンバイ・データベースへ向かう途中であるという可能性があります。

  NEARSYNC で述べたのと同じシナリオが発生した場合、トランザクション情報が失われる可能性は NEARSYNC の場合より高くなります。したがって、データ整合性の問題の発生する可能性は、 NEARSYNC の場合より、そして当然のことながら SYNC の場合より高くなります。

DB2 UDB ESE データベース、パーティション環境:

パーティション環境で ACR を使用する場合も、データ整合性の問題が発生する可能性があります。スタンバイ・データベースが同じデータベースの別のパーティションとして定義されている場合、前述の高可用性災害時リカバリーの NEARSYNC セクションで述べたシナリオで、未確定トランザクションをリカバリーすると、データ整合性の問題が発生する可能性があります。それは、パーティションはデータベース・トランザクション・ログを共有しないためです。したがって、スタンバイ・データベース (パーティション B) は、主データベース (パーティション A) に存在する未確定トランザクションを認識していません。

DB2 UDB ESE データベース、非パーティション環境:

非パーティション環境で ACR を使用する場合も、データ整合性の問題が発生する可能性があります。 IBM® AIX® High Availability Cluster Multiprocessor (HACMP)、 Microsoft® Cluster Service (MSCS)、 HP の Service Guard などのディスク・フェイルオーバー・テクノロジーを使用していないなら、主 (1 次) データベースに障害が発生した場合、主データベースには存在していたデータベース・トランザクション・ログも、 2 次データベースには存在しなくなります。したがって、前述の高可用性災害時リカバリーの NEARSYNC セクションで述べたシナリオで、未確定トランザクションをリカバリーすると、データ整合性の問題が発生する可能性があります。

パーティション・データベースにアクセスするトランザクション

パーティション・データベース環境では、ユーザー・データが複数のデータベース・パーティションにまたがって分散されることがあります。このデータベースにアクセスするアプリケーションは、データベース・パーティション (コーディネーター・ノード) のいずれかに接続し、要求を送信します。異なったアプリケーションが異なったデータベース・パーティションに接続する、また同じアプリケーションが異なった接続について異なったデータベース・パーティションを選択することができます。

パーティション・データベース環境のデータベースに対するトランザクションについては、*同一の* データベース・パーティションから、すべてのアクセスが行われなければなりません。つまり、トランザクションの開始からそのトランザクションがコミットされる時まで (この時点も含む)、同じデータベース・パーティションを使用しなければならないということです。

パーティション・データベースに対するトランザクションは、切断前にコミットされる必要があります。

関連概念:

- 212 ページの『X/Open 分散トランザクション処理のモデル』
- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『高可用性災害時リカバリーの概要』

関連資料:

- 219 ページの『xa_open ストリング形式』

xa_open ストリング形式

DB2 Universal Database™ (DB2 UDB) および DB2 Connect™ バージョン 8 フィックスパック 3 以降での xa_open ストリングの形式

以下は、xa_open ストリングの形式です。

```
parm_id1 = <parm value>,parm_id2 = <parm value>, ...
```

パラメーターは任意の順序で指定できます。 *parm_id* の有効値は、以下の表のとおりです。

表 36. parm_id の有効値

パラメーター名	値	必須か?	大文字小文字の区別	デフォルト値
DB	データベース別名	Yes	No	なし
データベースへのアクセスにアプリケーションが使用するデータベース別名。				
UID	ユーザー ID	No	Yes	なし
データベースへの接続を許可されているユーザー ID。パスワードが指定される場合に必要。				
PWD	パスワード	No	Yes	なし

表 36. parm_id の有効値 (続き)

パラメーター名	値	必須か?	大文字小文字の区別	デフォルト値
ユーザー ID に関連したパスワード。ユーザー ID が指定される場合に必要。				
TPM	トランザクション処理モニター名	No	No	なし
使用されている TP モニターの名前。サポートされている値については、次の表を参照してください。このパラメーターを指定すると、複数の TP モニターで単一の DB2 UDB インスタンスを使用できます。ここで指定した値は、データベース・マネージャー構成パラメーター *tp_mon_name* に指定された値をオーバーライドします。				
AXLIB	TP モニターの **ax_reg** 関数および **ax_unreg** 関数を含むライブラリー。	No	Yes	なし
この値は、必要な **ax_reg** 関数および **ax_unreg** 関数のアドレスを得るために DB2 UDB によって使用されます。この値を使って、TPM パラメーターに基づく仮定値をオーバーライドできます。または、TPM のリストに現れない TP モニターがこの値を使用することもできます。 AIX においてライブラリーがアーカイブ・ライブラリーの場合、ライブラリー名だけでなくアーカイブ・メンバーを指定する必要があります。たとえば、`AXLIB=/usr/mqm/lib/libmqmax_r.a(libmqmax_r.o)`.				
CHAIN_END	xa_end チェーニング・フラグ。有効な値は、`T`、`F`、または値なしです。	No	No	F
XA_END チェーニングとは、ネットワーク・フローを減らすために DB2 UDB が使用することのできる最適化です。 **xa_end** 呼び出しに続いて、ただちに同じスレッド (またはプロセス) で必ず **xa_prepare** が呼び出されるような TP モニター環境では、CHAIN_END がオンであれば、 xa_end フラグは **xa_prepare** コマンドと連結され、こうしてネットワーク・フローが 1 つ減ります。値 `T` は CHAIN_END がオンであることを示し、値 `F` は CHAIN_END がオフであることを示します。値を指定しない場合、CHAIN_END はオンになります。このパラメーターを使用して、特定の TPM 値から派生した設定をオーバーライドできます。				
SUSPEND_CURSOR	トランザクションの制御スレッドが中断されている場合にカーソルを保持するかどうかを指定します。有効な値は、`T`、`F`、または値なしです。	No	No	F

表 36. parm_id の有効値 (続き)

パラメーター名	値	必須か?	大文字小文字の区別	デフォルト値
トランザクション・ブランチを中断する TP モニターは、中断されたスレッドまたはプロセスを他のトランザクション用に再使用できます。 SUSPEND_CURSOR がオフである場合、HOLD 属性を持つカーソルを除くすべてのカーソルは閉じられます。中断されたトランザクションが再開されると、アプリケーションは再びカーソルを取得する必要があります。SUSPEND_CURSOR がオンである場合、開いたカーソルはいずれも閉じられず、再開時に、中断されたトランザクションに使用可能です。値 T 値は SUSPEND_CURSOR がオンであることを示し、値 F 値は SUSPEND_CURSOR がオフであることを示します。値を指定しない場合、SUSPEND_CURSOR はオンになります。このパラメーターを使用して、特定の TPM 値から派生した設定をオーバーライドできます。				
HOLD_CURSOR	トランザクションのコミット後に次のコミットまでカーソルを保持するかどうかを指定します。有効な値は、T、F、または値なしです。	No	No	F
通常、TP モニターは、スレッドまたはプロセスを複数のアプリケーション用に再使用します。新しくロードされたアプリケーションが、以前のアプリケーションによって開かれたカーソルを継承しないようにするために、カーソルはコミット後に閉じられます。HOLD_CURSORS がオンである場合、HOLD 属性を持つカーソルはいずれも閉じられず、トランザクション・コミット境界を介して持続します。このオプションを使用すると、グローバル・トランザクションは、同じ制御スレッドからコミットまたはロールバックされなければなりません。 HOLD_CURSOR がオフである場合、HOLD 属性を持ついずれのカーソルを開くこともリジェクトされます。値 T 値は HOLD_CURSOR がオンであることを示し、値 F 値は HOLD_CURSOR がオフであることを示します。値を指定しない場合、HOLD_CURSOR はオンになります。このパラメーターを使用して、特定の TPM 値から派生した設定をオーバーライドできます。				
TOC	DB2 UDB XA のすべての接続の結合先となるエンティティー (制御スレッド、『Thread of Control』)。有効な値は T、または P、または未設定です。	No	No	T (OS スレッド)

表 36. parm_id の有効値 (続き)

パラメーター名	値	必須か?	大文字小文字の区別	デフォルト値
TOC (Thread of Control) は、DB2 UDB XA のすべての接続の結合先となるエンティティーです。 1 つのエンティティー内で構成されるすべての DB2 UDB XA 接続はユニークでなければなりません。つまり、そのエンティティー内で同じデータベースに対して 2 つの接続が存在することは許されません。 TOC には T (OS スレッド) と P (OS プロセス) という 2 個のパラメーターがあります。 T の値を設定すると、特定の OS スレッドの下で構成される DB2 UDB XA 接続のすべては、そのスレッドに対してのみユニークになります。複数のスレッドによる DB2 UDB XA 接続の共用はできません。各 OS スレッドは、それぞれ DB2 UDB XA 接続の独自の集合を構成する必要があります。 P の値を設定すると、その OS プロセスに対してすべての DB2 UDB XA 接続がユニークになり、すべての XA 接続を OS スレッド間で共有できます。				
SREG	静的登録。有効な値は T、または F、または値なしです。	No	No	F
DB2 UDB では、グローバル・トランザクションの登録方法として 2 種類の方法がサポートされています。第 1 のものは動的登録であり、 DB2 UDB が TP の **ax_reg** 関数を呼び出すことにより、トランザクションを登録します (AXLIB を参照)。第 2 の方法は静的登録であり、 TP は XA API **xa_start** を呼び出すことにより、グローバル・トランザクションを開始します。動的登録と静的登録の両方とも相互排他的であることに注意してください。				
CREG	**xa_start** チェーニング・フラグ。有効な値は T、または F、または値なしです。	No	No	F
xa_start チェーニングとは、ネットワーク・フローを少なくするために DB2 UDB で使用される最適化の一種です。このパラメーターが有効なのは、 TP モニターで静的登録を使用している場合だけです (SREG を参照)。 TP モニター環境は、 XA API **xa_start** の呼び出しの直後に SQL ステートメントが呼び出されることを保証できるような環境です。 CREG が T に設定されているなら、 SQL ステートメントは **xa_start** 要求に対してチェーニングされ、ネットワーク・フローが 1 回分節約されます。このパラメーターを使用して、特定の TPM 値から派生した設定をオーバーライドできます。				

TPM 値および tp_mon_name 値

xa_open ストリングの TPM パラメーターとデータベース・マネージャー構成パラメーター *tp_mon_name* は、使用中の TP モニターを DB2 UDB に示すために使われます。 *tp_mon_name* 値は DB2 UDB インスタンス全体に適用されます。 TPM パラメーターは、特定の XA リソース・マネージャーにのみ適用されます。 TPM 値は *tp_mon_name* パラメーターをオーバーライドします。 TPM および *tp_mon_name* パラメーターの有効値は以下のとおりです。

表 37. TPM および tp_mon_name の有効値

TPM 値	TP モニター製品	内部設定
CICS	IBM TxSeries CICS	AXLIB=libEncServer (for Windows) =/usr/lpp/encina/lib/libEncServer (for UNIX based systems) HOLD_CURSOR=T CHAIN_END=T SUSPEND_CURSOR=F
ENCINA	IBM TxSeries Encina Monitor	AXLIB=libEncServer (for Windows) =/usr/lpp/encina/lib/libEncServer (for UNIX based systems) HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
MQ	IBM MQSeries	AXLIB=mqmax (for Windows) =/usr/mqm/lib/libmqmax_r.a (for AIX threaded applications) =/usr/mqm/lib/libmqmax.a (for AIX non-threaded applications) =/opt/mqm/lib/libmqmax.so (for Solaris) =/opt/mqm/lib/libmqmax_r.sl (for HP threaded applications) =/opt/mqm/lib/libmqmax.sl (for HP non-threaded applications) =/opt/mqm/lib/libmqmax_r.so (for Linux threaded applications) =/opt/mqm/lib/libmqmax.so (for Linux non-threaded applications) HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
CB	IBM Component Broker	AXLIB=somtrx1i (for Windows) =libsomtrx1 (for UNIX based systems) HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
SF	IBM San Francisco	AXLIB=ibmsfDB2 HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
TUXEDO	BEA Tuxedo	AXLIB=libtux HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
MTS	Microsoft Transaction Server	MTS 用に DB2 UDB を構成する必要はありません。 MTS は DB2 UDB の ODBC ドライバーによって自動的に検出されます。
JTA	Java Transaction API	IBM WebSphere などの Enterprise Java Server (EJS) 用に DB2 UDB を構成する必要はありません。 DB2 UDB の JDBC ドライバーは、この環境を自動的に検出します。したがって、この TPM 値は無視されます。

以前のバージョンの xa_open ストリング形式

以前のバージョンの DB2 UDB は、ここで説明する xa_open ストリング形式を使用します。この形式は、互換性のためにサポートされています。可能な限り、アプリケーションを新しい形式に移行してください。

各データベースは、トランザクション・マネージャー (TM) に対して別個のリソース・マネージャー (RM) として定義されているので、次の構文の xa_open ストリングによってデータベースを識別する必要があります。

```
"database_alias<,userid,password>"
```

database_alias は必須であり、データベースの別名を指定するものです。データベース作成後に明示的に別名のカタログを作成した場合を除き、この別名はデータベース名と同じになります。ユーザー名とパスワードは任意指定であり、認証方式によっては、データベースに認証情報を提供するために使用します。

例

1. Windows NT で IBM TxSeries CICS を使用しているとします。 TxSeries の資料によると、*tp_mon_name* を値 `libEncServer:C` に構成する必要があります。これは許容できる形式ですが、 DB2 UDB または DB2 Connect™ のバージョン 8 フィックスパック 3 以降では、以下のようなオプションもあります。
   - `CICS` の *tp_mon_name* を指定する (このシナリオで推奨される)。

     ```
     db2 update dbm cfg using tp_mon_name CICS
     ```

     「領域 (Region)」—> 「リソース (Resources)」—> 「製品 (Product)」—> 「XAD」—> 「リソース・マネージャー初期化ストリング (Resource manager initialization string)」で、 CICS に対して定義された各データベースごとに以下のように指定します。

     ```
     db=dbalias,uid=userid,pwd=password
     ```
   - 「領域 (Region)」—> 「リソース (Resources)」—> 「製品 (Product)」—> 「XAD」—> 「リソース・マネージャー初期化ストリング (Resource manager initialization string)」で、 CICS に対して定義された各データベースごとに以下のように指定します。

     ```
     db=dbalias,uid=userid,pwd=password,tpm=cics
     ```
2. Windows NT で IBM MQSeries を使用しているとします。 MQSeries の資料によると、 *tp_mon_name* を値 `mqmax` に構成する必要があります。これは許容できる形式ですが、 DB2 UDB または DB2 Connect のバージョン 8 フィックスパック 3 以降では、以下のようなオプションもあります。
   - `MQ` の *tp_mon_name* を指定する (このシナリオで推奨される)。

     ```
     db2 update dbm cfg using tp_mon_name MQ
     ```

     「領域 (Region)」—> 「リソース (Resources)」—> 「製品 (Product)」—> 「XAD」—> 「リソース・マネージャー初期化ストリング (Resource manager initialization string)」で、 CICS に対して定義された各データベースごとに以下のように指定します。

     ```
     uid=userid,db=dbalias,pwd=password
     ```

- 「領域 (Region)」—> 「リソース (Resources)」—> 「製品 (Product)」—> 「XAD」—> 「リソース・マネージャー初期化ストリング (Resource manager initialization string)」で、 CICS に対して定義された各データベースごとに以下のように指定します。

  ```
  uid=userid,db=dbalias,pwd=password,tpm=mq
  ```

3. Windows NT で IBM TxSeries CICS および IBM MQSeries の両方を使用しているとします。さらに、1 つの DB2 UDB インスタンスが使用されています。このシナリオでは、次のように構成します。

   a. 「領域 (Region)」—> 「リソース (Resources)」—> 「製品 (Product)」—> 「XAD」—> 「リソース・マネージャー初期化ストリング (Resource manager initialization string)」で、 CICS に対して定義された各データベースごとに以下のように指定します。

      ```
      pwd=password,uid=userid,tpm=cics,db=dbalias
      ```

   b. キュー管理プログラムのプロパティーでリソースとして定義されている各データベースごとに、XaOpenString を以下のように指定します。

      ```
      db=dbalias,uid=userid,pwd=password,tpm=mq
      ```

4. Windows NT で独自の XA 準拠トランザクション・マネージャー (XA TM) を開発していて、 DB2 UDB に対して、ライブラリー myaxlib に必要な関数 **ax_reg** および **ax_unreg** が入っていることを示したいとします。ライブラリー myaxlib は、PATH ステートメントで指定されたディレクトリーにあります。次のようなオプションがあります。

   - `myaxlib` の *tp_mon_name* を以下のように指定します。

     ```
     db2 update dbm cfg using tp_mon_name myaxlib
     ```

     その後、XA TM に定義されている各データベースごとに、 xa_open ストリングを以下のように指定します。

     ```
     db=dbalias,uid=userid,pwd=password
     ```

   - XA TM に定義されている各データベースごとに、 xa_open ストリングを以下のように指定します。

     ```
     db=dbalias,uid=userid,pwd=password,axlib=myaxlib
     ```

5. Windows NT で独自の XA 準拠トランザクション・マネージャー (XA TM) を開発していて、 DB2 UDB に対して、ライブラリー myaxlib に必要な関数 **ax_reg** および **ax_unreg** が入っていることを示したいとします。ライブラリー myaxlib は、PATH ステートメントで指定されたディレクトリーにあります。また、XA END チェーニングも使用可能にしたいとします。次のようなオプションがあります。

   - XA TM に定義されている各データベースごとに、 xa_open ストリングを以下のように指定します。

     ```
     db=dbalias,uid=userid,pwd=password,axlib=myaxlib,chain_end=T
     ```

   - XA TM に定義されている各データベースごとに、 xa_open ストリングを以下のように指定します。

     ```
     db=dbalias,uid=userid,pwd=password,axlib=myaxlib,chain_end
     ```

関連概念:

- 212 ページの『X/Open 分散トランザクション処理のモデル』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『tp_mon_name - 「トランザクション・プロセッサー・モニター名」構成パラメーター』

XA 準拠のトランザクション・マネージャーを使用したホストまたは iSeries データベース・サーバーの更新

XA トランザクション・マネージャーのアーキテクチャーによっては、ホストおよび iSeries データベース・サーバーを更新することができます。

手順:

異なるプロセスからの連続コミットをサポートするには、 DB2 Connect™ 接続コンセントレーターが使用可能でなければなりません。 DB2 Connect 接続コンセントレーターを使用可能にするには、データベース・マネージャー構成パラメーター *max_connections* を、 *maxagents* より大きな値に設定します。異なるプロセスからの連続 XA コミットを DB2 Connect 接続コンセントレーターがサポートするためには、 DB2 Universal Database™ (DB2 UDB) バージョン 7.1 クライアントが必要であることに注意してください。

DB2 UDB 同期点マネージャー (SPM) が構成されている DB2 Connect も必要です。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『maxagents - 「エージェントの最大数」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『max_connections - 「クライアント接続の最大数」構成パラメーター』

未確定トランザクションの手動での解決

XA 準拠のトランザクション・マネージャー (トランザクション処理モニター) は、DB2 Universal Database™ (DB2 UDB) トランザクション・マネージャーと同様な 2 フェーズ・コミットを使用します。これら 2 つの環境の主な違いは、 DB2 UDB トランザクション・マネージャーおよびトランザクション・マネージャー・データベースの代わりに、 TP モニターがトランザクションのロギングや制御の機能を提供することです。

DB2 UDB トランザクション・マネージャーについて起きるエラーと同様のエラーが、 XA 準拠のトランザクション・マネージャー使用中にも起きることがあります。 DB2 UDB トランザクション・マネージャーと同様、 XA 準拠のトランザクション・マネージャーは未確定トランザクションの再同期を試行します。

何らかの理由で、トランザクション・マネージャーが自動的に未確定トランザクションを解決するまで待てない場合は、未確定トランザクションを手動で解決するための措置がいくつかあります。この手動の処理は、「ヒューリスティック判定」と呼ばれることもあります。

(WITH PROMPTING オプションとともに) LIST INDOUBT TRANSACTIONS コマンドを使用して、または関連する API のセットを使用して、未確定トランザクションの照会、コミット、およびロールバックを行うことができます。さらに、ログ・レコードを削除してログ・スペースを解放することにより、ヒューリスティックな手法でコミットまたはロールバックされたトランザクションを「forget」することもできます。

制約事項:

これらのコマンド (または関連する API) は、あくまでも最後の手段として、*細心の注意* を払って使用してください。最善の方法は、トランザクション・マネージャーが再同期操作を始めるまで待つことです。ある参加データベースでは手動でトランザクションのコミットまたはロールバックを実行し、別の参加データベースでは正反対の処置を取ると、データ保全の問題が生じることがあります。データ保全の問題からリカバリーするには、アプリケーション論理を理解し、変更またはロールバックされたデータを識別して、次いでデータベースのポイント・イン・タイム・リカバリーを実行するか、または手動で変更の取り消し (またはやり直し) をする必要があります。

トランザクション・マネージャーが再同期を開始するまで待てず、かつ未確定トランザクションに結び付けられているリソースを解放しなければならない場合は、ヒューリスティックな操作が必要です。このような状況は、トランザクション・マネージャーが長時間使用できないために再同期を実行することができず、緊急に必要なリソースが未確定トランザクションによって拘束されている場合に発生する可能性があります。トランザクション・マネージャーまたはリソース・マネージャーが使用不能になる前に未確定トランザクションに関連していたリソースは、依然としてそのトランザクションに結び付けられています。データベース・マネージャーの場合、これらのリソースには、表や索引のロック、ログのスペース、およびそのトランザクションにより占有されているストレージなどが含まれます。各未確定トランザクションごとに、データベースで処理できる並行トランザクションの最大数も (1 つずつ) 減っていきます。さらに、すべての未確定トランザクションが解決されるまで、オフライン・バックアップは行うことはできません。

以下の状況では、ヒューリスティックな手法の forget 関数が必要です。

- ヒューリスティックな手法でコミットまたはロールバックされたトランザクションが原因で、ログ満杯状態が発生した場合 (LIST INDOUBT TRANSACTIONS コマンドからの出力に示される)
- オフライン・バックアップが行われる場合

ヒューリスティックな手法の forget 関数を実行すると、未確定トランザクションが占有していたログ・スペースが解放されます。つまり、トランザクション・マネージャーがこの未確定トランザクションに関して再同期操作を実行すると、このリソース・マネージャーにはトランザクションのログ・レコードがないために、他のリソース・マネージャーのコミットやロールバックを行うという間違った決定を下す危険性があります。一般に、ログ・レコードが「欠落」しているということは、リソース・マネージャーがトランザクションをロールバックしたことを暗示します。

手順:

ヒューリスティックな操作を実行する単純明快な方法というものはありませんが、一般的な指針を以下に示します。

1. すべてのトランザクションを完了しなければならないデータベースに接続する。
2. LIST INDOUBT TRANSACTIONS コマンドを使って、未確定トランザクションを表示する。このとき、*xid* はグローバル・トランザクション ID を表し、このトランザクションに参加しているトランザクション・マネージャーや他のリソース・マネージャーが使用する *xid* と同じです。
3. 各未確定トランザクションについて、アプリケーションとオペレーティング環境に関する知識を活用して、他の参加リソース・マネージャーを判別する。
4. トランザクション・マネージャーが利用可能かどうかを判別する。
   - トランザクション・マネージャーが使用可能であり、かつリソース・マネージャーが第 2 コミット・フェーズまたはそれ以前の再同期プロセスで使用可能でなかったためにリソース・マネージャー内で未確定トランザクションが発生した場合は、トランザクション・マネージャーのログを調べて、他のリソース・マネージャーに対しどのようなアクションがとられたかを判別してください。次いで、そのデータベースに対して同じ処置を取ります。つまり、LIST INDOUBT TRANSACTIONS コマンドを使って、トランザクションをヒューリスティックな手法でコミットするか、またはヒューリスティックな手法でロールバックします。
   - トランザクション・マネージャーが*利用不能* であれば、他の参加リソース・マネージャーにおけるそのトランザクションの状況を利用して、以下のように取るべき処置を判断します。
     - 他のリソース・マネージャーのうちの少なくとも 1 つがそのトランザクションをコミットしていれば、すべてのリソース・マネージャー内でそのトランザクションをヒューリスティックな手法でコミットしてください。
     - 他のリソース・マネージャーのうちの少なくとも 1 つがそのトランザクションをロールバックしていれば、そのトランザクションをヒューリスティックな手法でロールバックしてください。
     - そのトランザクションがすべての参加リソース・マネージャーで「準備済み」(未確定) 状態であれば、そのトランザクションをヒューリスティックな手法でロールバックしてください。
     - 他のリソース・マネージャーがまったく使用不可であれば、そのトランザクションをヒューリスティックな手法でロールバックしてください。

UNIX、または Windows の DB2 UDB から未確定トランザクション情報を得るには、データベースに接続し、 LIST INDOUBT TRANSACTIONS WITH PROMPTING コマンドまたはこれに対応する API を発行します。

ホストまたは iSeries データベース・サーバーに関連した未確定トランザクション情報は、以下の 2 つの方法のいずれかで取得できます。

- ホストまたは iSeries サーバーから未確定情報を直接入手する。

  DB2 for z/OS および OS/390 から未確定情報を直接取得するには、 DISPLAY THREAD TYPE(INDOUBT) コマンドを呼び出します。ヒューリスティック判定を実施するには、RECOVER コマンドを使用します。 DB2 for iSeries から未確定情報を直接取得するには、 **wrkcmtdfn** コマンドを呼び出します。

- ホストまたは iSeries データベース・サーバーへのアクセスに使用されている DB2 Connect サーバーから、未確定情報を取得する。

  DB2 Connect サーバーから未確定情報を取得するには、まず、データベース・マネージャー構成パラメーター *spm_name* の値で示される DB2 UDB インスタンスに接続することによって、 DB2 UDB 同期点マネージャーに接続します。次に、未確定トランザクションを表示してヒューリスティック判定を実施するために、 LIST DRDA INDOUBT TRANSACTIONS WITH PROMPTING コマンドを発行します。

関連概念:

- 205 ページの『2 フェーズ・コミット』

関連資料:

- 「コマンド・リファレンス」の『LIST INDOUBT TRANSACTIONS コマンド』
- 「コマンド・リファレンス」の『LIST DRDA INDOUBT TRANSACTIONS コマンド』

関連サンプル:

- 『dbxamon.c -- Show and roll back indoubt transactions.』

XA トランザクション・マネージャーのセキュリティーに関する考慮事項

TP モニターは一連のサーバー・プロセスを事前に割り振り、それらのサーバー・プロセスの ID 下で異なるユーザーからトランザクションを実行します。データベース側からすれば、各サーバー・プロセスは、そのサーバー・プロセスに関連した同じ ID で実行中の多くの作業単位を持つ 1 つの巨大なアプリケーションのように見えます。

たとえば、CICS® を使用している AIX® 環境では、TXSeries® CICS 領域が始動すると、その領域は定義されている AIX ユーザー名に関連付けられます。すべての CICS アプリケーション・サーバー・プロセスも、この TXSeries CICS の「マスター」ID (通常 ″cics″ と定義されている) で実行されます。 CICS ユーザーは DCE ログイン ID で CICS トランザクションを呼び出すことができ、 CICS にいる間は、CESN サインオン・トランザクションを使用して ID を変更することもできます。どちらの場合も、RM にはエンド・ユーザーの ID を使用できません。結果として、CICS アプリケーション・プロセスは多くのユーザーの代行としてトランザクションを実行することになりますが、 RM からは、それらが同じ ″cics″ ID の多くの作業単位を伴う単一プログラムのように見えます。オプションとして xa_open ストリングにユーザー ID とパスワードを指定すると、データベース接続時には、″cics″ ID ではなくそのユーザー ID が使用されます。

静的 SQL ステートメントの場合は、エンド・ユーザーの特権ではなく、バインド側の特権を使用してデータベースにアクセスするので、あまり影響はありません。ただし、これは、データベース・パッケージの EXECUTE 特権をエンド・ユーザー ID ではなくサーバー ID に与える必要があるというわけではありません。

ランタイムにアクセス認証を行う動的ステートメントの場合は、データベース・オブジェクトへのアクセス特権は、それらのオブジェクトの実際のユーザーではなく、サーバーの ID に付与する必要があります。データベースによって特定のユー

ザーのアクセスを制御するのではなく、 TP モニター・システムを利用して、どのユーザーがどのプログラムを実行できるかを判別する必要があります。サーバー ID には、SQL ユーザーが必要とするすべての特権を付与することが必要です。

だれがデータベース表またはビューにアクセスしたかを調べるためには、以下のステップを実行することができます。

1. SYSCAT.PACKAGEDEP カタログ・ビューから、その表またはビューに依存するすべてのパッケージのリストを入手する。
2. インストール時に使用した命名規則により、それらのパッケージに対応するサーバー・プログラム (CICS プログラムなど) の名前が何かを調べる。
3. それらのプログラムを呼び出せるクライアント・プログラム (CICS トランザクション ID など) を調べ、 TP モニターのログ (CICS ログなど) を使用して、いつだれがこれらのトランザクションまたはプログラムを実行したかを調べる。

関連概念:

- 212 ページの『X/Open 分散トランザクション処理のモデル』

XA トランザクション・マネージャーの構成に関する考慮事項

TP モニター環境を設定する場合は、次の構成パラメーターを考慮してください。

- *tp_mon_name*

  このデータベース・マネージャー構成パラメーターは、使用される TP モニター製品の名前を識別します (たとえば CICS や ENCINA)。

- *tpname*

  このデータベース・マネージャー構成パラメーターは、データベース・クライアントが APPC 通信プロトコルを使用してデータベース・サーバーに割り振り要求を出すときに、使用しなければならないリモート・トランザクション・プログラムの名前を指定します。値はサーバーの構成ファイルで設定されます。この値は SNA トランザクション・プログラムに構成されているトランザクション・プロセッサー (TP) の名前と同じでなければなりません。

- *tm_database*

  DB2® Universal Database (DB2 UDB) は XA 環境でトランザクションを調整しない ので、このデータベース・マネージャー構成パラメーターは、 XA 調整済みトランザクションには使用されません。

- *maxappls*

  このデータベース構成パラメーターには、アクティブなアプリケーションの許容最大数を指定します。このパラメーターの値は、接続されるアプリケーションの合計数に、 2 フェーズ・コミットまたはロールバックを同時に完了しようとする可能性のあるアプリケーションの数を加えた値より大きいか等しくなければなりません。さらに、任意の時点で存在する可能性のある未確定トランザクションの数を、この合計に加算してください。

TP モニター環境 (たとえば TXSeries® CICS®) の場合は、 *maxappls* パラメーターの値を大きくする必要があるかもしれません。こうすれば、すべての TP モニター・プロセスを確実に記憶できるようになります。

- *autorestart*

  このデータベース構成パラメーターには、必要に応じて RESTART DATABASE ルーチンを自動的に呼び出すかどうかを指定します。デフォルト値は YES (呼び出す) です。

  未確定トランザクションが含まれているデータベースは、データベースの再始動操作によって始動する必要があります。データベースへの最後の接続がドロップされるときに *autorestart* が使用可能でない場合、その次の接続は失敗し、再び RESTART DATABASE を明示的に呼び出す必要があります。この状態は、トランザクション・マネージャーの再同期操作によって、または手動による管理者の開始するヒューリスティックな操作によって、未確定トランザクションが除去されるまで続きます。 RESTART DATABASE コマンドが発行されるとき、データベースに未確定トランザクションが存在していれば、メッセージが戻されます。管理者は、 LIST INDOUBT TRANSACTIONS コマンドなどのコマンド行プロセッサーのコマンドを使用することによって、それらの未確定トランザクションについての情報を検索できます。

関連概念:

- 212 ページの『X/Open 分散トランザクション処理のモデル』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『autorestart - 「自動再始動使用可能」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『tpname - 「APPC トランザクション・プログラム名」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『maxappls - 「アクティブ・アプリケーションの最大数」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『tm_database - 「トランザクション・マネージャー・データベース名」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『tp_mon_name - 「トランザクション・プロセッサー・モニター名」構成パラメーター』
- 「*コマンド・リファレンス*」の『LIST INDOUBT TRANSACTIONS コマンド』
- 「*コマンド・リファレンス*」の『RESTART DATABASE コマンド』

DB2 Universal Database によってサポートされる XA 機能

DB2® Universal Database (DB2 UDB) は、 *X/Open CAE Specification Distributed Transaction Processing: The XA Specification* で定義されている XA91 仕様をサポートしますが、以下は例外です。

- 非同期サービス

  XA 仕様では、インターフェースで非同期サービスを使用することができます。このサービスを使用すると、要求の結果を後で調べることができます。データベース・マネージャーでは、要求を同期モードで呼び出す必要があります。

- 静的登録

  XA インターフェースでは、静的登録と動的登録という 2 つの RM 登録方法が可能です。 DB2 UDB は、より高機能で効率的な動的登録のみをサポートしています。

- 関連の移行

  DB2 UDB は、制御スレッド間のトランザクション移行をサポートしていません。

XA スイッチの使用法と位置

XA インターフェースとして必要とされるものとして、データベース・マネージャーには、 XA スイッチ構造体を TM に戻すために使う *xa_switch_t* 型の外部 C 変数 db2xa_switch が用意されています。さまざまな XA 関数のアドレス以外に、以下のフィールドが返されます。

フィールド	値
name	データベース・マネージャーの製品名。たとえば、DB2 for AIX®。
flags	TMREGISTER | TMNOMIGRATE DB2 UDB が動的登録を使用し、 TM は関連の移行を使用してはならないことを明示的に示します。非同期操作がサポートされないことを暗黙的に示します。
version	常に 0。

DB2 Universal Database XA スイッチの使用

XA アーキテクチャーでは、 XA トランザクション・マネージャー (TM) がリソース・マネージャー (RM) の *xa_* ルーチンにアクセスできるようにする**スイッチ**を、RM が提供しなければなりません。 RM のスイッチは、xa_switch_t と呼ばれる構造体を使用します。スイッチには、RM の名前、RM の XA 入り口点への非 NULL ポインター、フラグ、およびバージョン番号が含まれます。

UNIX ベースのシステム

DB2 UDB のスイッチは、以下の 2 つの方法のいずれかによって得られます。

- 間接的なレベルを追加して使用する。 C プログラムでは、これは次のマクロを定義することによって行うことができます。

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

  ただし、これは *db2xa_switch* を使用する前に行います。

- **db2xacic** を呼び出す。

  DB2 UDB は、 *db2xa_switch* 構造体のアドレスを戻すこの API を提供します。この関数のプロトタイプは次のとおりです。

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

いずれの場合も、`libdb2` を使用して、アプリケーションをリンクする必要があります。

Windows NT

xa_switch 構造体 *db2xa_switch* を示すポインターは DLL データとしてエクスポートされます。したがって、この構造体を使用する Windows® NT アプリケーションは、次の 3 つのいずれかの方法でこれを参照する必要があります。

- 間接的なレベルを追加して使用する。 C プログラムでは、これは次のマクロを定義することによって行うことができます。

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

  ただし、これは *db2xa_switch* を使用する前に行います。
- Microsoft® Visual C++ コンパイラーを使用する場合は、 *db2xa_switch* は次のように定義することができる。

  ```
  extern __declspec(dllimport) struct xa_switch_t db2xa_switch
  ```
- **db2xacic** を呼び出す。

  DB2 UDB は、 *db2xa_switch* 構造体のアドレスを戻すこの API を提供します。この関数のプロトタイプは次のとおりです。

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

いずれの方式でも、 `db2api.lib` を使用してアプリケーションをリンクする必要があります。

C コードの例

以下のコードは、任意の DB2 UDB プラットフォーム上の C プログラムで *db2xa_switch* にアクセスするいくつかの方法を示しています。必ずアプリケーションを適切なライブラリーとリンクしてください。

```
   #include <stdio.h>
   #include <xa.h>

   struct xa_switch_t * SQL_API_FN  db2xacic( );

   #ifdef DECLSPEC_DEFN
   extern __declspec(dllimport) struct xa_switch_t db2xa_switch;
   #else
   #define db2xa_switch (*db2xa_switch)
   extern struct xa_switch_t db2xa_switch;
   #endif

main( )
   {
      struct xa_switch_t *foo;
      printf ( "%s ¥n", db2xa_switch.name );
      foo = db2xacic();
      printf ( "%s ¥n", foo->name );
      return ;
   }
```

関連概念:

- 212 ページの『X/Open 分散トランザクション処理のモデル』

XA インターフェースの問題判別

TM からの XA 要求時にエラーが検出された場合、アプリケーション・プログラムは TM からそのエラー・コードを入手することはできません。ご使用のプログラムが異常終了したり、 TP モニターまたは TM からの暗号戻りコードを受け取ったりした場合、基本障害保守ログを調べてください。診断レベルが 3 以上であればここに XA エラー情報が報告されています。

その他に、コンソール・メッセージ、TM エラー・ファイル、またはご使用の外部トランザクション処理ソフトウェア製品固有の情報も調べてください。

データベース・マネージャーは、XA 固有のエラーを基本障害保守ログに書き込み、その際 SQLCODE -998 (トランザクション・エラーまたはヒューリスティック・エラー) と該当する理由コードを指定します。最も一般的なエラーには、以下のようなものがあります。

- xa_open ストリングの構文が無効。
- 以下のいずれかの結果としてオープン・ストリングに指定されているデータベースに接続されなかった。
  - データベースのカタログが作成されなかった。
  - データベースが始動しなかった。
  - サーバー・アプリケーションのユーザー名またはパスワードでは、データベースへの接続が許可されない。
- 通信エラー

関連概念:

- 212 ページの『X/Open 分散トランザクション処理のモデル』

関連資料:

- 219 ページの『xa_open ストリング形式』

XA トランザクション・マネージャー構成

IBM WebSphere Application Server の構成

IBM® WebSphere™ Application Server は、 Java ベースのアプリケーション・サーバーです。 DB2 JDBC ドライバーに用意されている Java Transaction API (JTA) によって、 DB2 Universal Database™ (DB2 UDB) の XA サポートを使用できるようになっています。 WebSphere Application Server で Java Transaction API を使用する方法については、 IBM WebSphere の資料を参照してください。 WebSphere Application Server の資料は、 http://www.ibm.com/software/webservers/appserv/infocenter.html にてオンラインで閲覧できます。

IBM TXSeries CICS の構成

DB2 Universal Database™ (DB2 UDB) をリソース・マネージャーとして使用するよう IBM TXSeries CICS® を構成する方法については、お手持ちの「*IBM TXSeries*

CICS 管理ガイド」を参照してください。 TXSeries の資料は、http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/ にてオンラインで閲覧できます。

ホストおよび iSeries データベース・サーバーは、 CICS 調整トランザクションに参加することができます。

IBM TXSeries Encina の構成

以下に示すさまざまな API および構成パラメーターは、 Encina モニターと DB2 Universal Database™ (DB2 UDB) サーバーの統合に必要とされるもの、および (DB2 Connect™ を使ってアクセスする場合) DB2 for z/OS and OS/390、 DB2 for iSeries、または DB2 for VSE & VM と Encina モニターとの統合に必要とされるものです。 TXSeries の資料は、http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/ にてオンラインで閲覧できます。

ホストおよび iSeries データベース・サーバーは、 Encina 調整トランザクションに参加することができます。

DB2 Universal Database の構成

DB2 Universal Database™ (DB2 UDB) を構成するには、

1. すべてのデータベース名を DB2 UDB データベース・ディレクトリーに定義する必要があります。データベースがリモート・データベースの場合は、ノード・ディレクトリーも定義する必要があります。構成を行うには、構成アシスタントまたは DB2 UDB コマンド行プロセッサー (CLP) を使用できます。たとえば、以下のとおりです。

   ```
   DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER
   DB2 CATALOG TCPIP NODE host1 REMOTE hostname1 SERVER svcname1
   ```

2. DB2 UDB クライアントは、Encina を処理していることを認識している場合は、内部処理を Encina 用に最適化できます。これを指定するには、 *tp_mon_name* データベース・マネージャー構成パラメーターを `ENCINA` に設定します。デフォルト動作は、特別な最適化なしです。 *tp_mon_name* を設定する場合、アプリケーションでは、作業単位を実行するスレッドもまた、作業完了の直後にその作業を必ずコミットしなければなりません。他の作業単位を開始してはなりません。ご使用の環境がこのようになっていない 場合は、 *tp_mon_name* 値を必ず `NONE` にしてください (または、 CLP からこの値を `NULL` に設定します)。このパラメーターはコントロール・センターまたは CLP から更新できます。 CLP コマンドを以下に示します。

   ```
   db2 update dbm cfg using tp_mon_name ENCINA
   ```

リソース・マネージャーごとの Encina の構成

Encina をリソース・マネージャー (RM) ごとに構成するには、管理者は、リソース・マネージャーをアプリケーション内のトランザクションに登録する前に、各 DB2 UDB データベースのオープン・ストリング、クローズ・ストリング、および制御の取り決め (Control Agreement) のスレッドを、リソース・マネージャーとして

定義する必要があります。 Enconcole フルスクリーン・インターフェース、または Encina コマンド行インターフェースを使用して構成を実行できます。たとえば、以下のとおりです。

```
monadmin create rm inventdb -open "db=inventdb,uid=user1,pwd=password1"
```

各 DB2 UDB データベースごとに 1 つのリソース・マネージャーがあります。各リソース・マネージャー構成には、 1 つの rm 名 (「論理 RM 名」) がなければなりません。状況を単純にするため、それをデータベース名と同じにするとよいでしょう。

xa_open ストリングには、データベースへの接続を確立するために必要な情報が入っています。このストリングの内容は、RM によって異なります。 DB2 UDB の xa_open ストリングには、開くデータベースの別名が入っており、オプションで、接続に関連させるユーザー ID とパスワードが入っています。ここで定義されるデータベース名は、すべてのデータベース・アクセスに必要な正規のデータベース・ディレクトリーにもカタログする必要があることに注意してください。

DB2 UDB は、xa_close ストリングを使用しません。

制御スレッドの取り決め (Thread of Control Agreement) は、アプリケーション・エージェント・スレッドが同時に 2 つ以上のトランザクションを扱うことができるかどうかを判別します。

DB2 for z/OS and OS/390、DB2 for iSeries、または DB2 for VSE & VM にアクセスする場合は、 DB2 Syncpoint Manager を使用する必要があります。

Encina アプリケーションから DB2 UDB データベースを参照する

Encina アプリケーションから DB2 UDB データベースを参照するには、次のようにします。

1. Encina Scheduling Policy API を使用して、単一 TP モニター・アプリケーション・プロセスから実行できるアプリケーション・エージェントの数を指定します。たとえば、以下のとおりです。

   ```
   rc = mon_SetSchedulingPolicy (MON_EXCLUSIVE)
   ```

2. Encina RM Registration API を使用して、XA スイッチと、アプリケーション・プロセスで RM を参照する時に Encina が使用する論理 RM 名を提供します。たとえば、以下のとおりです。

   ```
   rc = mon_RegisterRmi ( &db2xa_switch,   /* xa switch */
                          "inventdb",      /* logical RM name */
                          &rmiId );        /* internal RM ID */
   ```

   XA スイッチは、TM が呼び出すことのできる RM の XA ルーチンのアドレスを含んでおり、 RM が提供する機能性も指定します。 DB2 UDB の XA スイッチは、`db2xa_switch` で、これは DB2 UDB クライアント・ライブラリー (Windows オペレーティング・システムでは `db2app.dll`、 UNIX ベースのシステムでは `libdb2`) にあります。

   論理 RM 名は Encina が使用する名前で、 Encina の下で実行される SQL アプリケーションが使用する実際のデータベース名ではありません。実際のデータベース名は、Encina RM 登録 API の xa_open ストリングで指定されます。この例では、論理 RM 名がデータベース名と同じになるように設定されています。

3 番目のパラメーターは、この接続を参照するために TM が使用する内部 ID またはハンドルを戻します。

関連概念:

- 「*DB2 Connect ユーザーズ・ガイド*」の『DB2 Connect とトランザクション処理モニター』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『tp_mon_name - 「トランザクション・プロセッサー・モニター名」構成パラメーター』
- 219 ページの『xa_open ストリング形式』

BEA Tuxedo の構成

手順:

DB2 Universal Database™ (DB2 UDB) をリソース・マネージャーとして使用するよう Tuxedo を構成するには、以下のステップを実行します。

1. Tuxedo の資料で指定されているように Tuxedo をインストールする。ログ・ファイルと環境変数を含めた、Tuxedo のすべての基本構成を必ず実行してください。

   コンパイラーと DB2 UDB Application Development Client も必要です。必要ならこれらをインストールします。

2. Tuxedo サーバー ID で、 Tuxedo に使用させたいデータベースを含むインスタンスを参照するように DB2INSTANCE 環境変数を設定します。 DB2 UDB プログラム・ディレクトリーを含むように PATH 変数を設定します。 Tuxedo サーバー ID で DB2 UDB データベースに接続できることを確認します。
3. `TUXEDO` の値で *tp_mon_name* データベース・マネージャー構成パラメーターを更新します。
4. DB2 UDB の定義を Tuxedo リソース・マネージャー定義ファイルに追加します。以下の例では、`UDB_XA` は、ローカルに定義される DB2 UDB のリソース・マネージャー名で、 *db2xa_switch* は、タイプ `xa_switch_t` の構造体の DB2 定義の名前です。
   - AIX の場合: 以下の定義をファイル `${TUXDIR}/udataobj/RM` に追加します。

     ```
     # DB2 UDB
     UDB_XA:db2xa_switch:-L${DB2DIR} /lib -ldb2
     ```

     ここで、 `{TUXDIR}` は Tuxedo をインストールしたディレクトリー、 `{DB2DIR}` は DB2 UDB インスタンス・ディレクトリーです。

   - Windows NT の場合: ファイル `%TUXDIR%¥udataobj¥rm` の中に、次の定義を追加します。

     ```
     # DB2 UDB
     UDB_XA;db2xa_switch;%DB2DIR%¥lib¥db2api.lib
     ```

     ここで、`%TUXDIR%` は Tuxedo をインストールしたディレクトリー、 `%DB2DIR%` は DB2 UDB インスタンス・ディレクトリーです。
5. 次のようにして、DB2 UDB の Tuxedo トランザクション・モニター・サーバー・プログラムを構築する。

- AIX の場合:

```
${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
```

  ここで、{TUXDIR} は Tuxedo をインストールしたディレクトリーです。

- Windows NT の場合、

```
%TUXDIR%¥bin¥buildtms -r UDB_XA -o %TUXDIR%¥bin¥TMS_UDB
```

6. アプリケーション・サーバーを構築する。以下の例では、-r オプションはリソース・マネージャー名を指定し、 -f オプション (複数回使用可) はアプリケーション・サービスを含むファイルを指定し、 -s オプションはこのサーバーのアプリケーション・サービス名を指定し、 -o オプションは出力サーバー・ファイル名を指定しています。

   - AIX の場合:

```
${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
   -o UDBserver
```

     ここで、{TUXDIR} は Tuxedo をインストールしたディレクトリーです。

   - Windows NT の場合、

```
%TUXDIR%¥bin¥buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
   -o UDBserver
```

     ここで、%TUXDIR% は Tuxedo をインストールしたディレクトリーです。

7. DB2 UDB サーバーを参照するように Tuxedo 構成ファイルを設定する。UDBCONFIG ファイルの *GROUPS セクションに、次のような項目を追加します。

```
UDB_GRP   LMID=simp GRPNO=3
  TMSNAME=TMS_UDB TMSCOUNT=2
  OPENINFO="UDB_XA:db=sample,uid=db2_user,pwd=db2_user_pwd"
```

   ここで、TMSNAME パラメーターは以前に作成したトランザクション・モニター・サーバー・プログラムを指定し、 OPENINFO パラメーターはリソース・マネージャー名を指定しています。これに続けてデータベース名と DB2 UDB ユーザー ID とパスワードがありますが、これらは認証に使用されます。

   以前に構築したアプリケーション・サーバーは、 Tuxedo 構成ファイルの *SERVERS セクション内で参照されています。

8. DB2 for z/OS and OS/390、DB2 for iSeries、または DB2 for VM & VSE にあるデータにアプリケーションがアクセスする場合は、 DB2 Connect XA コンセントレーターが必要です。

9. 次のようにして Tuxedo を開始する。

```
tmboot -y
```

   コマンドが終了すると、Tuxedo メッセージはサーバーが開始されたことを示します。さらに、DB2 UDB コマンド LIST APPLICATIONS ALL を出すと、2 つの接続が表示されます。これらの接続は、(この場合は、) Tuxedo 構成ファイル UDBCONFIG によって設定された UDB グループの TMSCOUNT パラメーターで指定されます。

関連概念:

- 「*DB2 Connect* ユーザーズ・ガイド」の『DB2 Connect とトランザクション処理モニター』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『tp_mon_name - 「トランザクション・プロセッサー・モニター名」構成パラメーター』
- 「*コマンド・リファレンス*」の『LIST APPLICATIONS コマンド』

第 3 部 付録

付録 A. リリース間の非互換性

このセクションでは、 DB2 Universal Database™ (DB2 UDB) の現行リリースと DB2 Universal Database の以前のリリースとの間の非互換性について説明します。

非互換性 とは、 DB2 Universal Database のうち、DB2 Universal Database の以前のリリースとは異なる動作をする部分です。既存のアプリケーションでこれを使用すると、予期しない結果が発生したり、アプリケーションを変更する必要が生じたり、またはパフォーマンスが低下します。ここでいう「アプリケーション」とは、以下のものを指します。

- アプリケーション・プログラム・コード
- サード・パーティー製ユーティリティー
- 対話式 SQL 照会
- コマンドまたは API 呼び出し

ここでは、 DB2 Universal Database バージョン 7 とバージョン 8 の非互換性について説明します。非互換性は次のカテゴリーで分類されています。

- システム・カタログ情報
- アプリケーション・プログラミング
- SQL
- データベースのセキュリティーと調整
- ユーティリティーとツール
- 接続性と共存
- メッセージ
- 構成パラメーター

それぞれの非互換性セクションでは、非互換性の説明、非互換性の症状と影響、および可能な解決方法について説明します。また、それぞれの非互換性の説明の先頭に、非互換性の当てはまるオペレーティング・システムを示す標識が以下のように示されます。

Windows
DB2 Universal Database がサポートする Microsoft Windows® プラットフォーム

UNIX DB2 Universal Database がサポートする UNIX® 系プラットフォーム

OS/2 OS/2® (DB2 Universal Database バージョン 7 のみ)

DB2 Universal Database で計画されている非互換性

このセクションでは、将来の非互換性について説明します。 DB2 Universal Database™ (DB2 UDB) のユーザーは、新しいアプリケーションを作成する時、または既存のアプリケーションを変更する時に、これを念頭に置く必要があります。これにより、DB2 UDB の将来のバージョンに容易に移行することができます。

システム・カタログ情報

将来の DB2 Universal Database のバージョンでの PK_COLNAMES および FK_COLNAMES

WIN	UNIX

変更点: SYSCAT.REFERENCES の列 PK_COLNAMES および FK_COLNAMES は、使用できなくなります。

症状: 列が存在せず、エラーが戻されます。

説明: ツールまたはアプリケーションが、使われなくなった PK_COLNAMES および FK_COLNAMES 列を使用するようにエンコードされています。

解決方法: 代わりに SYSCAT.KEYCOLUSE ビューを使用するように、ツールまたはアプリケーションを変更します。

将来の DB2 Universal Database のバージョンで COLNAMES が使用できない

WIN	UNIX

変更点: SYSCAT.INDEXES の列 COLNAMES は使用できなくなります。

症状: 列が存在せず、エラーが戻されます。

説明: ツールまたはアプリケーションが、使われなくなった COLNAMES 列を使用するようにエンコードされています。

解決方法: 代わりに SYSCAT.INDEXCOLUSE ビューを使用するように、ツールまたはアプリケーションを変更します。

ユーティリティーとツール

type-1 索引再作成のサポートの除去

WIN	UNIX

変更点: バージョン 8 では、 type-2 索引という、新しいタイプの索引が導入されています。 type-1 索引 (バージョン 8 よりも前で作成された索引) では、表行の削除または更新の一部として、キーがリーフ・ページから物理的に削除されます。 type-2 索引では、行の削除または更新時にキーが削除済みとしてマークされますが、その削除または更新がコミットされるまで、物理的に除去されることはありません。 type-1 索引の再作成のサポートが除去されても、ご使用の索引を手動で再作成する必要はありません。 type-1 索引は引き続き正常に機能します。索引が再作成されるすべてのアクションでは、 type-1 索引が type-2 索引に自動的に変換されます。将来のバージョンでは、 type-1 索引のサポートが除去されます。

説明: type-2 索引には、 type-1 索引と比べて以下のような利点があります。

- 長さが 255 バイトを超える type-2 索引を、列に作成できます。
- Next Key ロックの使用が最小限まで減少するため、並行性が向上します。

解決方法: 長期的に既存の索引を type-2 索引に変換していく計画を立ててください。オンライン索引再編成機能は、可用性に関する障害を最小限にとどめつつ、これを行うのに役立ちます。必要に応じて、索引の表スペースのサイズを増やしてください。 LARGE 表スペースに新しい索引を作成し、既存の索引をその LARGE 表スペースに移動することを考慮してください。

バージョン 8 と以前のリリースとの非互換性

システム・カタログ情報

カタログ表の IMPLEMENTED 列

Windows	UNIX

変更点: 以前のバージョンでは、 SYSIBM.SYSFUNCTIONS および SYSCAT.SYSFUNCTIONS 内の IMPLEMENTED 列の値は Y、 M、H、および N でした。バージョン 8 では、値は Y と N です。

解決方法: 値 Y および N のみを使用するようアプリケーション・コードを変更します。

OBJCAT ビューから SYSCAT ビューへの名前変更

Windows	UNIX

変更点: 以下の OBJCAT ビューは SYSCAT ビューに名前変更されました。TRANSFORMS、INDEXEXTENSIONS、INDEXEXTENSIONMETHODS、INDEXEXTENSIONDEP、INDEXEXTENSIONPARMS、PREDICATESPECS、INDEXEXPLOITRULES。

解決方法: SYSCAT ビューを使用するようアプリケーション・コードを変更します。

読み取り専用になった SYSCAT ビュー

Windows	UNIX

変更点: バージョン 8 では、SYSCAT ビューが読み取り専用になりました。

症状: SYSCAT スキーマ内のビューに対して UPDATE または INSERT 操作を行うと失敗します。

説明: システム・カタログ情報を更新するには、 SYSSTAT ビューを使用することをお勧めします。 SYSCAT ビューでも更新可能でしたが、これは意図されない動作であり、修正されました。

解決方法: 更新可能な SYSSTAT ビューを新たに参照するようアプリケーションを変更します。

アプリケーション・プログラミング

監査コンテキスト・レコード・ステートメントのサイズの増加

Windows	UNIX

変更点: ステートメントの限界が 2 MB になりました。

症状: 監査コンテキスト・レコード・ステートメント・テキストが大きすぎて、表に入りません。

説明: 監査コンテキスト・レコードを記録するために使用される既存の表では、ステートメント・テキストとして可能なのは 32 KB だけです。新しいステートメントの限界は 2 MB です。長いステートメントを使用しない場合、このことは影響しません。

解決方法: ステートメント・テキスト列の値が CLOB(2M) である監査コンテキスト・レコードを入れる新しい表を作成します。必要なら、新しい表に以前の表のデータを入れてから、以前の表をドロップし、新しい表を使用します。新しい表の名前は、以前の表と同じ名前に変更できます。新しい表を使用するアプリケーションがあるなら、それを再バインドします。

アプリケーションはデフォルトでマルチスレッドで実行される

Windows	UNIX

変更点: バージョン 8 の場合、アプリケーションは、デフォルトでマルチスレッド・モードで実行されます。以前のバージョンの場合、デフォルトでアプリケーションは、単一スレッド・モードで実行されていました。この変更により、sqleSetTypeCtx API の呼び出しには効果がなくなります。

バージョン 8 のマルチスレッド・モードは、バージョン 8 より前のアプリケーションで SQL_CTX_MULTI_MANUAL オプションを指定した sqleSetTypeCtx API を呼び出すことと同じです。バージョン 7 クライアントは、引き続き単一スレッド・モードでアプリケーションを実行できます。

説明: バージョン 7 の場合、アプリケーションをマルチスレッド・モードで実行するためには、コンテキスト API を呼び出してコンテキストを管理する必要がありました。バージョン 8 では、それが不要になりました。 DB2 Universal Database™ (DB2 UDB) が内部でコンテキストを管理するからです。しかし、必要ならバージョン 8 でも、外部コンテキスト API を使用することによって、アプリケーションのためにコンテキストを管理することが可能です。

VERSION オプション使用時に SQL0818N エラーが戻されない

Windows	UNIX

変更点: PRECOMPILE、BIND、REBIND、および DROP PACKAGE コマンドで新しい VERSION オプションを使用すると、実行要求に対して SQL0818N エラーではなく、 SQL0805N エラーが戻されるようになりました。

症状: SQL0818N エラーに対応するアプリケーション・コードは、以前のように動作しなくなる可能性があります。

解決方法: SQL0805N と SQL0818N の両方のエラーに対応するようアプリケーション・コードを変更します。

ホスト変数が未定義の場合、 SQL0306N エラーがプリコンパイラーに戻されない

Windows	UNIX

変更点: BEGIN DECLARE セクションで宣言されないホスト変数が EXEC SQL セクションで使われた場合、プリコンパイラーによって SQL0306N が戻されなくなりました。アプリケーション内の他のどこかの場所で変数が宣言されている場合、アプリケーション・ランタイムは SQL0804N を戻します。変数がアプリケーションのどこにも宣言されていない場合、コンパイル時にコンパイラーがエラーを戻します。

症状: プリコンパイル時に SQL0306N エラーに対応するアプリケーション・コードは、以前のように動作しなくなる可能性があります。

解決方法: ホスト変数を BEGIN DECLARE セクションで宣言してください。BEGIN DECLARE 以外のセクションでホスト変数を宣言する場合、戻りコード SQL0804 を処理するようアプリケーション・コードを変更する必要があります。

両方向スクロール・カーソルとともに使用できないデータ・タイプ

Windows	UNIX

変更点: LONG VARCHAR、LONG VARGRAPHIC、DATALINK、LOB の各タイプと、これらのタイプの特殊タイプ、および構造型を使用する両方向スクロール・カーソルは、バージョン 8 ではサポートされません。バージョン 7 の両方向スクロール・カーソルでサポートされていたこれらのデータ・タイプは、すべてサポートされなくなりました。

症状: このいずれかのデータ・タイプの列が両方向スクロール・カーソルの選択リストで指定された場合、 SQL0270N 理由コード 53 が戻されます。

解決方法: 両方向スクロール・カーソルの選択リストを変更して、これらのタイプの列が含まれないようにします。

ユーロ・バージョンのコード・ページ変換表

Windows	UNIX

変更点: バージョン 8 のコード・ページ変換表はユーロ記号をサポートしますが、以前のバージョンの DB2 UDB の変換表とは少し異なっています。

解決方法: バージョン 8 より前のコード・ページ変換表を使用したい場合には、ディレクトリー `sqllib/conv/v7` にこれらの変換表があります。

LOB ロケーターと LOB 値の間の切り替え

Windows	UNIX

変更点: カーソル・ステートメントのバインド・アウト時の、ラージ・オブジェクト (LOB) ロケーターと LOB 値の間の切り替え機能が変更されました。アプリケーションが SQLRULES DB2 でバインドされるとき (デフォルトの動作)、ユーザーは LOB ロケーターと LOB 値の間で切り替えできません。

解決方法: カーソル・ステートメントのバインド・アウト時に LOB ロケーターと LOB 値の間で切り替えたい場合には、アプリケーションを SQLRULES STD でプリコンパイルしてください。

UNIX プラットフォームでコミットされない作業単位

	UNIX

変更点: バージョン 8 では、すべてのアプリケーションの終了によって、未解決の作業単位は暗黙的にロールバックされます。 Windows ベースのアプリケーションは変更されていません。 Windows ベースのアプリケーションの場合、アプリケーションが正常終了したか異常終了したかにかかわらず、これまでどおり暗黙的に ROLLBACK が実行されます。バージョン 8 より前は、コンテキストを明示的にも暗黙的にもサポートしない UNIX ベースのアプリケーションでは、 CONNECT RESET、COMMIT、あるいは ROLLBACK ステートメントを直接呼び出さずにアプリケーションが正常終了した場合、未解決の作業単位はコミットされました。 (コンテキストを暗黙的にサポートする) CLI、ODBC、および Java ベースのアプリケーションの場合、およびアプリケーション・コンテキストを明示的に作成するアプリケーションの場合には、アプリケーション終了時に未解決の作業単位は常にロールバックされました。さらに、アプリケーションが異常終了したときにも、未解決の作業単位に対して暗黙的に ROLLBACK が実行されました。

解決方法: トランザクションを確実にコミットするためには、アプリケーションが終了する前に、明示的 COMMIT または CONNECT RESET を実行する必要があります。

セーブポイントの命名に関する変更

Windows	UNIX

変更点: セーブポイントの名前の先頭に ″SYS″ を使用することはできなくなりました。

症状: 名前が ″SYS″ で始まるセーブポイントを作成しようとすると、エラー SQL0707N が発生して失敗します。

説明: 名前が ″SYS″ で始まるセーブポイントは、システムで使用するために予約されています。

解決方法: 名前が ″SYS″ で始まるすべてのセーブポイントの名前を変更して、″SYS″ 以外で始まるようにしてください。

コード・ページ変換エラーとバイト置換

Windows	UNIX

変更点: 入力ホスト変数内の文字データは、必要であれば、ホスト変数が表示される SQL ステートメント内で使用される前に、データベース・コード・ページに変換されます。コード・ページの変換時に、データ拡張が起こる場合があります。以前は、ホスト変数内のデータでコード・ページ変換を検出すると、この拡張に対応するために、ホスト変数に想定される実際の長さが増分されました。現在では、データ・タイプの長さの変更による、その他の SQL 操作に対する影響を軽減するために、この長さについての想定の増分は実行されません。

注: このいずれも、FOR BIT DATA のコンテキスト内で使用されるホスト変数には適用されません。これらのホスト変数内のデータは、ビット・データに関して使用される前には変換されません。

症状: ホスト変数が、コード・ページ変換後に、拡張された長さを保持するために十分な大きさでない場合には、エラーが戻されます (SQLSTATE 22001、SQLCODE -302)。

説明: コード・ページ変換時に拡張または収縮が行われる可能性があるため、ホスト変数内のデータの長さに依存する操作の結果が異なったり、エラー状態が発生したりする場合があります。

解決方法: 考えられる代替手段には、以下のものがあります。

- 文字ホスト変数の長さの増分でデータの長さが変わる原因となる、コード・ページ変換の可能性を扱うアプリケーションをコーディングする。
- 拡張の原因となる文字を回避するようにデータを変更する。
- コード・ページ変換が起こらないようにするために、データベース・コード・ページと一致するようにアプリケーション・コード・ページを変更する。

ホスト変数のコード・ページ変換

Windows	UNIX

変更点: コード・ページ変換が必要な場合、変換はバインド・イン段階で実行されるようになりました。

症状: 異なる結果。

説明: コード・ページ変換が必要な場合、常にホスト変数に関して変換されるようになったため、述部評価はアプリケーション・コード・ページではなく、常にデータベース・コード・ページで行われます。たとえば、

```
SELECT * FROM table WHERE :hv1 > :hv2
```

この処理は、アプリケーション・コード・ページではなく、データベース・コード・ページを使って行われます。使用される照合は、これまでどおりデータベース照合です。

解決方法: 以前のバージョンでの結果が、本当に望ましい結果であったかどうかを確認してください。望ましい結果であった場合には、述部を変更して、使用されるデータベース照合とコード・ページのもとで適切な結果が生成されるようにしてください。あるいは、アプリケーション・コード・ページまたはデータベース・コード・ページを変更してください。

ホスト変数内のデータの拡張と収縮

Windows	UNIX

変更点: 必要であれば、コード・ページ変換は、バインド操作時に実行されます。

症状: ホスト変数のデータの長さが異なります。

説明: コード・ページ変換時に拡張または収縮が行われる可能性があるため、ホスト変数内のデータの長さに依存する操作の結果が異なったり、エラー状態が発生したりする場合があります。

解決方法: データ、アプリケーション・コード・ページ、またはデータベース・コード・ページを変更して、コード・ページ変換によって変換後のデータの長さが変更されないようにしてください。あるいは、コード・ページ変換によってデータの長さが変更された場合に対処できるよう、アプリケーション・コードを変更してください。

コード・ページ変換後のホスト変数の長さ

Windows	UNIX

変更点: コード・ページ変換の際、拡張のために、ホスト変数やパラメーター・マーカーの結果の長さが増幅されることはなくなりました。

症状: データ切り捨てエラー。

説明: タイプなしパラメーター・マーカー用に決定される文字データ・タイプの長さは、もはやコード・ページ変換による拡張の可能性を考慮して増幅されません。結果の長さは、タイプなしパラメーター・マーカーの長さを使って結果の長さを決定する操作のために、より短くなります。たとえば、C1 が CHAR(10) 列とすると、

```
VALUES CONCAT (?, C1)
```

この結果は、アプリケーション・コード・ページからデータベース・コード・ページへの変換で 3 倍に拡張される可能性のあるデータベースの場合、もはや CHAR(40) のデータ・タイプおよび長さにはならず、 CHAR(20) の結果データ・タイプおよび長さになります。

解決方法: CAST を使用して、タイプなしパラメーター・マーカーに適切なタイプを設定するか、タイプなしパラメーター・マーカーのタイプを決定するオペランドを変更して、コード・ページ変換による拡張に対応できるようなデータ・タイプおよび長さにします。

DESCRIBE ステートメントの出力に関する変更

Windows	UNIX

変更点: コード・ページ変換の際、拡張のために、ホスト変数やパラメーター・マーカーの結果の長さが増幅されることはなくなりました。

症状: DESCRIBE ステートメントの出力が変更されました。

説明: コード・ページ変換による拡張可能性を考慮して結果の長さが増幅されることはないため、そのような結果の長さを記述する DESCRIBE ステートメントの出力が異なります。

解決方法: 必要であれば、DESCRIBE ステートメントから戻される新しい値を扱えるようにアプリケーションを変更してください。

ホスト変数とともに SUBSTR 関数を使用する際のエラー

Windows	UNIX

変更点: コード・ページ変換の際、拡張のために、ホスト変数やパラメーター・マーカーの結果の長さが増幅されることはなくなりました。

症状: SUBSTR からのエラー SQL0138N。

説明: コード・ページ変換による拡張の可能性を考慮して、ホスト変数用の長さが増幅されていました。これによって、たとえば `SUBSTR (:hv,19,1)` が、長さ 10 のホスト変数に関して正常に機能しました。今後は、これが機能しません。

解決方法: 変換後のデータの長さを考慮に入れるようホスト変数の長さを増幅するか、 SUBSTR 呼び出しを変更して、ホスト変数の長さ以内の位置を指定します。

非スレッド・セーフのライブラリーはもはや Solaris でサポートされない

	UNIX

変更点: 非スレッド・セーフ・ライブラリー `libdb2_noth.so` はもはや使用できません。

症状: `libdb2_noth.so` を必要とするツールやアプリケーションは機能しません。

説明: 非スレッド・セーフ・ライブラリーを引き続きサポートする必要がないため、 `libdb2_noth.so` ライブラリーは DB2 UDB (Solaris オペレーティング環境 (Solaris Operating EnviroNment™ 版) に付属していません。

解決方法: 新たにスレッド・セーフの `libdb2.so` ライブラリーを使用するよう、ツールまたはアプリケーションを変更してください。 -mt パラメーターを使ってアプリケーションを再リンクしてください。

Unicode データベースへの接続時の DBCLOB のインポートまたはエクスポート

Windows	UNIX

変更点: バージョン 8 以前は、 Unicode データベース (UTF-8) から DBCLOB を含むデータをエクスポートしたときに、 LOBSINFILE ファイル・タイプ修飾子を使用した場合、 DBCLOB はコード・ページ 1200 (Unicode GRAPHIC コード・ページ) でエクスポートされました。 DBCLOB を含むデータをインポートしたときに、LOBSINFILE ファイル・タイプ修飾子を使用した場合、 DBCLOB はコード・ページ 1200 (Unicode GRAPHIC コード・ページ) でインポートされました。この動作は、DB2GRAPHICUNICODESERVER レジストリー変数を ON に設定している場合に、バージョン 8 でも維持されます。

バージョン 8 では、DB2GRAPHICUNICODESERVER レジストリー変数のデフォルト設定は OFF です。 DBCLOB を含むデータをエクスポートするときに、LOBSINFILE ファイル・タイプ修飾子を使用している場合、 DBCLOB はアプリケーションの GRAPHIC コード・ページでエクスポートされます。 DBCLOB を含むデータをインポートするときに、LOBSINFILE ファイル・タイプ修飾子を使用している場合、 DBCLOB はアプリケーションの GRAPHIC コード・ページでインポートされます。ご使用のアプリケーション・コードが IBM-eucJP (954) または IBM-eucTW (964) で、 DBCLOB を含むデータをエクスポートするときに、LOBSINFILE ファイル・タイプ修飾子を使用している場合、 DBCLOB はアプリケーションの文字コード・ページでエクスポートされます。 DBCLOB を含むデータをインポートするときに、LOBSINFILE ファイル・タイプ修飾子を使用している場合、 DBCLOB はアプリケーションの文字コード・ページでインポートされます。

症状: LOBSINFILE ファイル・タイプ修飾子を持つデータを、 Unicode データベースにインポートしている場合、文字データは正しく変換されますが、DBCLOB データは壊れます。

解決方法: バージョン 8 データベースと古いデータベース間でデータを移動している場合は、 DB2GRAPHICUNICODESERVER レジストリー変数を ON に設定して、以前の動作を保存します。

SQL

関数とプロシージャーの同一の特定名は許可されない

Windows	UNIX

変更点: SPECIFICNAME 用のネーム・スペースが統一されました。以前のバージョンの DB2 UDB では、関数とプロシージャーの特定名を同じにすることができましたが、バージョン 8 ではこれは許可されません。

症状: データベースをバージョン 8 に移行する場合、 db2ckmig ユーティリティーが関数とプロシージャーの同じ特定名をチェックします。移行の際に重複名が検出されると、移行は失敗します。

解決方法: プロシージャーをドロップし、別の特定名を使って再作成してください。

関数およびプロシージャーの EXECUTE 特権

Windows	UNIX

変更点: これまでは、他のユーザーがルーチンを使用できるようにするには、単にルーチンを作成する必要があるだけでした。今後は、ルーチンを作成した後、それに対してまず GRANT EXECUTE を実行すれば、他のユーザーはそれを使用できるようになります。

以前のバージョンではプロシージャーに関する許可検査がありませんでしたが、呼び出し側では、プロシージャーから呼び出されるすべてのパッケージに関する EXECUTE 特権が必要でした。バージョン 8 では、 CALL_RESOLUTION IMMEDIATE を使ってプリコンパイルされる組み込みアプリケーションの場合、および CLI カタログ式プロシージャーの場合には、呼び出し側にプロシージャーに関する EXECUTE 特権が必要です。すべてのパッケージに関する EXECUTE 特権が必要なのは、プロシージャー定義者のみです。

症状:

1. アプリケーションが正しく動作しない可能性があります。
2. 複数のパッケージからなる既存のプロシージャーで、定義者がすべてのパッケージへのアクセスを許可されているわけではない場合には、プロシージャーが正しく動作しません。

解決方法:

1. 必要な GRANT EXECUTE ステートメントを発行します。すべてのルーチンが単一のスキーマに入っている場合、それぞれのタイプのルーチンごとに以下のような 1 つのステートメントを使用して、特権を付与します。

   ```
   GRANT EXECUTE ON FUNCTION schema1.* TO PUBLIC
   ```

2. 1 つのパッケージはすべてのユーザーが使用でき、別のパッケージは少数の特権ユーザーにのみ制限されている場合、両方のパッケージを使用するストアード・プロシージャーは、 2 番目のパッケージにアクセスするとき、権限エラーを監視します。権限エラーが検出された場合には、そのユーザーが特権ユーザーではないことが判明したため、プロシージャーの論理の一部がう回されます。

   これを解決するには、以下のいくつかの方法があります。

   a. プログラムをプリコンパイルするとき、 CALL_RESOLUTION DEFERRED を設定します。こうすれば、プリコンパイラーが CALL ステートメントでプロシージャーの解決に失敗した場合、使用すべきでない sqleproc() API 呼び出しとしてプログラムが実行されます。
   b. プロシージャーの呼び出し方法を制御するために、 CLI キーワード UseOldStpCall を db2cli.ini ファイルに追加することができます。この値は 2

つあり、値 *0* の場合は古い呼び出しメソッドを使ってプロシージャーを呼び出しません。値 *1* の場合、古い呼び出しメソッドを使ってプロシージャーを呼び出します。

c. パッケージを実行するすべてのユーザーに EXECUTE 特権を付与します。

表への外部キー制約の追加

Windows	UNIX

変更点: 以前のバージョンでは、チェック・ペンディング状態の表を参照する外部キー制約を作成した場合、従属表もまたチェック・ペンディング状態になりました。バージョン 8 では、チェック・ペンディング状態の表を参照する外部キー制約を作成した場合、結果は 2 通りの可能性があります。

1. 従属表の作成時に外部キー制約が追加された場合、表は空の状態で作成され、制約に違反する行が存在しないため、表の作成と制約の追加はどちらも正常に行われます。
2. 既存の表に外部キーが追加された場合、エラー SQL0668N が発生します。

解決方法: 表を参照する外部キーを追加する前に、 SET INTEGRITY ... IMMEDIATE CHECKED ステートメントを使用して、チェック・ペンディング状態にある表の保全性チェックをオンにします。

SET INTEGRITY ... IMMEDIATE CHECKED の変更

Windows	UNIX

変更点: 前のリリースでは、 SET INTEGRITY ... UNCHECKED ステートメントが発行された表 (つまり、 SYSCAT.TABLES の const_checked 列に 'U' バイトが含まれる表) は、デフォルトで、次の SET INTEGRITY ... IMMEDIATE CHECKED ステートメントで完全に処理されました (つまり、すべてのレコードの制約違反がチェックされました)。全体が処理されないようにするには、明示的に INCREMENTAL を指定する必要がありました。

バージョン 8 では、 SET INTEGRITY ... IMMEDIATE CHECKED ステートメントが発行された場合、デフォルトでは増分処理だけを行い、未チェック・データをそのままにします (つまり 'U' バイトを保持します)。 (古いデータが検証されない旨の警告が戻されます。)

説明: このように変更された理由は、デフォルト動作で、 (多くのリソースを消費する) すべてのレコードの制約チェックを行わないようにするためです。

解決方法: 全体の処理を強制するためには、明示的に NOT INCREMENTAL を指定する必要があります。

CHAR 関数の小数点

Windows	UNIX

変更点: 小数点としてコンマを使用するロケールがサーバーに設定されている場合、そこで実行される動的アプリケーションに、引き数タイプ REAL または DOUBLE を使った CHAR 関数の非修飾呼び出しが含まれていれば、CHAR(double) 関数の結果としてピリオドが区切り文字として戻されます。さらに、ビューやトリガーのようなオブジェクトをバージョン 8 で再作成した場合や、静的パッケージを明示的に再バインドした場合にも、こうした非互換性が見られます。

説明: これは、SYSFUN.CHAR(double) 関数シグニチャーではなく、新しい SYSIBM.CHAR(double) 関数シグニチャーに解決されたためです。

解決方法: 以前のバージョンの DB2 UDB の動作を保持するには、関数解決で SYSIBM.CHAR シグニチャーが選択されるのを防ぐために、アプリケーションで明示的に関数 SYSFUN.CHAR を呼び出す必要があります。

CALL ステートメントの変更

Windows	UNIX

変更点: バージョン 8 では、 CALL_RESOLUTION IMMEDIATE および CLI カタログ式プロシージャーを使ってプリコンパイルされたアプリケーションには、以前のバージョンと比べて以下のような大きな相違点があります。

- ホスト変数サポートに代わって、動的 CALL がサポートされます。
- アンカタログ・ストアード・プロシージャーを呼び出すアプリケーションのコンパイルは、サポートされなくなりました。アンカタログ・ストアード・プロシージャーは、将来の DB2 UDB バージョンでは完全にサポートされなくなる予定です。
- 変数引き数リスト・ストアード・プロシージャーはサポートされなくなりました。
- ストアード・プロシージャー・ライブラリーのロードに関する規則が異なります。

解決方法: バージョン 8 より前にサポートされていた CALL ステートメントを引き続き使用できます。 PRECOMPILE PROGRAM コマンドで CALL_RESOLUTION DEFERRED オプションを使用して、これを利用することができます。

(バージョン 8 より前に作成された) 既存のアプリケーションは、引き続き動作します。 CALL_RESOLUTION DEFERRED オプションを使わずにアプリケーションが再プリコンパイルされる場合、ソース・コードを変更する必要があるかもしれません。

CALL_RESOLUTION DEFERRED ステートメントは、将来のバージョンではサポートされなくなる予定です。

固定長ストリングを戻す UDF からの出力

Windows	UNIX

変更点: UDF (スカラー関数または表関数) が固定長ストリング (CHAR(n) または GRAPHIC(n)) を戻すよう定義することができます。以前のバージョンでは、戻り値

に組み込み NULL 文字が含まれる場合、 NULL 文字および NULL 文字の右側のすべてのバイトを含めて、結果は単に n バイト (GRAPHIC データ・タイプの場合は 2n バイト) になりました。バージョン 8 では DB2 UDB は NULL 文字を検索し、その地点 (NULL 文字) 以降、値の末尾までブランクを戻します。

解決方法: バージョン 8 より前の動作を保持したい場合には、戻り値の定義を CHAR(n) から CHAR(n) FOR BIT DATA に変更してください。 GRAPHIC データに関しては、バージョン 8 より前の動作を保持する方法はありません。

データベース接続動作の変更点

Windows	UNIX

変更点: バージョン 7 では、組み込み SQL を使ってデータベースに接続した後、存在しないデータベースへの接続を試行した場合、 SQL1013N が発生してその試行は失敗しますが、最初のデータベースへの接続は存続しました。バージョン 8 では、存在しないデータベースへの接続を試行すると、最初のデータベースへの接続が切断されます。これによって、アプリケーションはどこにも接続していないままの状態になります。

解決方法: 別のデータベースへの接続試行が失敗した後、最初のデータベースに再接続するよう、組み込み SQL をコーディングしてください。

パッケージに関する CONTROL の取り消し

Windows	UNIX

変更点: ユーザーは CONTROL 特権を使用して、パッケージに関する特権を付与できます。 DB2 UDB バージョン 8 では、あるユーザーがパッケージ使用特権を他のユーザーに GRANT できるかどうかの権限を決定するために、 WITH GRANT OPTION を使用できます。このメカニズムは、ユーザーが他のユーザーに特権を付与できるかどうかを決定するうえで、 CONTROL に代わるものです。 CONTROL が取り消されても、ユーザーは引き続き他のユーザーに特権を付与できます。

症状: CONTROL 特権が取り消された後でも、ユーザーは引き続きパッケージに関する特権を付与できます。

解決方法: パッケージに関する許可を今後あるユーザーに与えないようにするには、パッケージに関するすべての特権を取り消して、必要な特権のみを付与してください。

FOR BIT DATA 文字ストリングを CLOB にキャストする際のエラー

Windows	UNIX

変更点: CAST 指定または CLOB 関数を使用して、 FOR BIT DATA として定義されている文字ストリングを CLOB にキャストすると、エラー (SQLSTATE 42846) が戻されるようになりました。

症状: CLOB にキャストすると、これまでとは異なり、エラーが戻されます。

説明: FOR BIT DATA は CLOB データ・タイプに関してサポートされていません。 FOR BIT DATA ストリングを引き数とする CAST 指定または CLOB 関数の使用結果は、定義されていません。この状況は、エラーとして検知されるようになりました。

解決方法: CAST 指定または CLOB 関数への引き数を変更して、 FOR BIT DATA ストリング以外にします。そうするには、CAST 指定を使用し、 FOR BIT DATA ストリングを FOR SBCS DATA ストリングまたは FOR MIXED DATA ストリングにキャストします。たとえば、C1FBD が FOR BIT DATA と宣言された VARCHAR(20) 列であれば、非 DBCS データベースでは、以下のような引き数が CLOB 関数への引き数として有効です。

```
CAST (C1FBD AS VARCHAR(20) FOR SBCS DATA)
```

CHR 関数からの出力

Windows	UNIX

変更点: CHR(0) は、コード・ポイント X'00' の文字の代わりに、ブランク (X'20') を返します。

症状: X'00' を引き数とした CHR 関数からの出力が、異なる結果を返します。

説明: ユーザー定義関数から呼び出され、返されるときのストリング処理が、X'00' をストリングの終わりと解釈します。

解決方法: 新しい出力値を処理するように、アプリケーション・コードを変更してください。または、SYSFUN CHR 関数にある CHAR(1) FOR BIT DATA を返すユーザー定義関数を定義して、この関数を SQL パス上で SYSFUN の前に置き換えます。

TABLE_NAME および TABLE_SCHEMA 関数は生成された列またはチェック制約で使用できません

Windows	UNIX

変更点: TABLE_NAME および TABLE_SCHEMA 関数の定義が訂正され、生成された列またはチェック制約では使用できなくなります。

症状: バインドは SQLCODE -548/SQLSTATE 42621 で、 TABLE_NAME または TABLE_SCHEMA がチェック制約のコンテキストで無効であると述べられ、失敗します。

説明: TABLE_NAME および TABLE_SCHEMA 関数はカタログ・ビューからデータを検索します。これらはクラス READS SQL DATA に属します。クラス READS SQL DATA の機能は GENERATED COLUMN 式およびチェック制約では許可されません。これは DB2 UDB が長期にわたる制約の正確さを保証できないためです。

解決方法: 生成列およびチェック制約が含まれる任意の列を更新して、TABLE_NAME と TABLE_SCHEMA の使用を除去してください。生成列を変更す

るには、 ALTER TABLE ステートメントを使用して、新しい式を設定します。チェック制約を除去するには、 DROP CONSTRAINT 文節を指定して ALTER TABLE ステートメントを使用してください。これによって変更を反映した列が含まれる表のバインドとアクセスの続行が可能になります。

データベースのセキュリティーと調整

CREATE FUNCTION、CREATE METHOD、および CREATE PROCEDURE ステートメントに関する権限

Windows	UNIX

変更点: CREATE_EXTERNAL_ROUTINE 権限がバージョン 8 で新たに導入されました。

症状: EXTERNAL オプションを使用する CREATE FUNCTION、CREATE METHOD、および CREATE PROCEDURE ステートメントは失敗します。

解決方法: EXTERNAL オプションを使用する CREATE FUNCTION、CREATE METHOD、および CREATE PROCEDURE ステートメントを発行するユーザーに対して、 CREATE_EXTERNAL_ROUTINE 権限を付与してください。

ユーティリティーとツール

コントロール・センターを使用してパフォーマンスをモニターする際の変更

Windows	UNIX

症状: コントロール・センター内に、パフォーマンス・モニターへの参照がありません。

説明: コントロール・センターのパフォーマンス・モニター機能は除去されました。

解決方法: DB2 Universal Database™ (DB2 UDB) for Windows® で作業している場合、パフォーマンスをモニターするために使用できるいくつかのツールがあります。

- **DB2 Performance Expert**

  別売の DB2 Performance Expert (マルチプラットフォーム版)、バージョン 1.1 は、 DB2 UDB のパフォーマンス関連情報に基づいて自動管理およびソース・チューニングの変更を統合、報告、分析、および推奨します。

- **DB2 UDB ヘルス・センター**

  ヘルス・センターの機能により、パフォーマンス関連情報を処理するためのさまざまな手法が可能になります。それらの機能は、コントロール・センターのパフォーマンス・モニター機能にある程度置き換わるものです。

- **Windows パフォーマンス・モニター**

Windows パフォーマンス・モニターを使用すると、データベースとシステムのパフォーマンスをモニターし、システムに登録されているパフォーマンス・データ提供元から情報を取り出すことができます。さらに Windows は、次のようなマシン操作のあらゆる面に関するパフォーマンス情報データを提供します。

- CPU 使用率
- メモリーの使用率
- ディスク・アクティビティー
- ネットワークの活動

さまざまなオンライン・ユーティリティーの同時実行

Windows	UNIX

症状: 複数のオンライン・ユーティリティーを同時に使用すると、その実行完了のための時間が長くかかる場合があります。

説明: 1 つのユーティリティーで必要なロックが、同時に実行されている他のユーティリティーの実行に影響します。

解決方法: 同時に実行されているさまざまなユーティリティーの間で、ロッキングの要件に競合がある場合、ユーティリティー実行のスケジュールを変更することを検討してください。表スペースのオンライン・バックアップ、表のロード、表のインプレース再編成などのユーティリティーでは、ユーティリティー相互の競合を避けるためにロッキング・メカニズムを使用しています。それらのユーティリティーでは、データベースで実行する処理の内容を制御するために、さまざまなタイミングで表ロック、表スペース・ロック、および表スペース状態が使用されます。あるユーティリティーがロックをかけると、それと同様のロックやそれに関連したロックを要求する他のユーティリティーは、それらのロックが解除されるまで待機する必要があります。

たとえば、インプレース表再編成の最終フェーズは、再編成の対象となる表を含むオンライン・バックアップの実行中には開始できません。バックアップを完了することが必要なら、再編成要求を一時停止することができます。

別の例として、オンライン・ロード・ユーティリティーは、同じ表に対する別のオンライン・ロード要求を処理しません。異なる表がロードされている場合には、ロード要求は相互にブロックしません。

db2move サマリー出力に対する変更

Windows	UNIX

変更点: バージョン 8.2 において、 **db2move** によって生成されるサマリー出力は、改善されて記述性が高くなりました。しかし、サマリー出力が変更されたため、以前の出力を分析するために作成されたスクリプトが失敗する可能性があります。

症状: **db2move** によって生成される旧式出力を分析するために作成されたスクリプトが失敗します。

説明: **db2move** によって生成されるサマリー出力が改善されました。

db2move に “IMPORT” オプションが指定されて実行された場合、以前は次のように出力されていました。

```
IMPORT: -Rows read:     5; -Rows committed:     5;   Table "DSCIARA2"."T20"
```

新しい出力は、次のようになります。

```
* IMPORT:  table "DSCIARA2"."T20"
  -Rows read:          5
  -Inserted:           4
  -Rejected:           1
  -Committed:          5
```

db2move に “LOAD” オプションが指定されて実行された場合、以前は次のように出力されていました。

```
* LOAD:  table "DSCIARA2"."T20"
  -Rows read:    5; -Loaded:    4;  -Rejected   1  -Deleted   0  -Committed   5
```

新しい出力は、次のようになります。

```
* IMPORT:  table "DSCIARA2"."T20"
  -Rows read:          5
  -Loaded:             4
  -Rejected:           1
  -Deleted:            0
  -Committed:          5
```

解決方法: **db2move** の出力を分析するためのスクリプトを、レイアウトと内容の変更を反映するよう変更する必要があります。

Explain 機能表に対する変更

Windows	UNIX

変更点: バージョン 8 では、既存の Explain 機能表にいくつかの変更が加えられ、 ADVISE_MQT および ADVISE_PARTITION という 2 個の新しい表があります。

症状: DB2 Design Advisor に、マテリアライズ照会表 (MQT) またはデータベース・パーティションのための推奨事項作成が依頼されると、 Explain 表が作成されていない場合にエラー・メッセージが戻されます。

説明: 新しい表 ADVISE_MQT および ADVISE_PARTITION が作成されていません。

解決方法: **db2exmig** コマンドを使用することによって、バージョン 7 とバージョン 8.1 の Explain 表をバージョン 8.2 に移行してください。このコマンドには、必要な Explain 機能表をすべて作成するために必要な EXPLAIN DLL が用意されています。

db2diag.log メッセージ形式に対する変更

Windows	UNIX

変更点: バージョン 8 では、db2diag.log のメッセージ形式が変更されています。

症状: db2diag.log メッセージの形式が変更されています。たとえば、各メッセージには診断ログ・レコード・ヘッダーがあり、レコード・フィールドの前にフィールドの名前と列があり、またロギング・レコードのメッセージとデータの部分がはっきりとマークされている、などの変更点があります。形式の変更点は、いずれもロギング・レコードを使いやすく理解しやすいものにするためのものです。

説明: DB2 UDB 診断ログが変更されました。 db2diag.log ファイルが構文解析可能になります。

下位レベルの CREATE DATABASE および DROP DATABASE はサポートされない

Windows	UNIX

変更点: バージョン 8 では、下位クライアントからの、または下位サーバーへの CREATE DATABASE コマンドと DROP DATABASE コマンドはサポートされません。

症状: これらのいずれかのコマンドを発行した場合、エラー SQL0901N が発生します。

説明: CREATE DATABASE コマンドおよび DROP DATABASE コマンドは、バージョン 8 のクライアントとサーバーの間でのみサポートされています。これらのコマンドを、バージョン 6 または 7 のクライアントからバージョン 8 サーバーに発行することはできません。これらのコマンドを、バージョン 8 クライアントからバージョン 7 サーバーに発行することはできません。

解決方法: バージョン 8 クライアントからは、バージョン 8 のデータベースを作成またはドロップしてください。バージョン 7 データベースの作成またはドロップは、バージョン 6 または 7 のクライアントから行ってください。

ロード後の表のモード変更

Windows	UNIX

変更点: 以前のバージョンでは、INSERT オプションを使ってロードされ、即時マテリアライズ照会表 (サマリー表ともいう) を含んでいる表は、 SET INTEGRITY IMMEDIATE CHECKED ステートメントの実行後に標準 (全アクセス権限) 状態になりました。バージョン 8 では、 SET INTEGRITY IMMEDIATE CHECKED ステートメント実行後に表は非データ移動モードになります。

説明: 非データ移動モードの表へのアクセスは標準 (全アクセス権限) モードの表へのアクセスとほぼ同様ですが、その表内部でのデータ移動を行う一部のステートメントやユーティリティーは使用できません。

解決方法: 従属即時サマリー表を含んでいるロード済みの基本表が、非データ移動モードをう回して、全アクセス権限モードに直接移行するように強制できます。そうするには、基本表に対して SET INTEGRITY ... IMMEDIATE CHECKED FULL

ACCESS ステートメントを発行します。ただし、これによって従属即時マテリアライズ照会表 (サマリー表ともいう) のフル・リフレッシュが強制されるため、このオプションの使用は推奨されていません。

挿入モードまたは置換モードのロード・ユーティリティー

Windows	UNIX

変更点: 以前のバージョンでは、ロード・ユーティリティーを挿入モードまたは置換モードで使用するとき、保全性検査がオフの場合のデフォルト・オプションは CASCADE IMMEDIATE でした。表がチェック・ペンディング状態になると、そのすべての従属外部キー表および従属マテリアライズ照会表 (サマリー表ともいう) もまた、ただちにチェック・ペンディング状態になりました。

バージョン 8 の場合、ロード・ユーティリティーを挿入モードまたは置換モードで使用するとき、保全性検査がオフの場合のデフォルトは CASCADE DEFERRED です。

解決方法: 従属外部キー表および従属マテリアライズ照会表を親表とともにチェック・ペンディング状態にするには、 LOAD コマンドの CHECK PENDING CASCADE IMMEDIATE オプションを使用することができます。

DB2_LIKE_VARCHAR がサブエレメント統計の収集を制御しない

Windows	UNIX

変更点: バージョン 7 では、DB2_LIKE_VARCHAR レジストリー変数はサブエレメント統計の収集を、これらの統計の使用と同様に制御していました。バージョン 8 では、DB2_LIKE_VARCHAR はサブエレメント統計の収集を制御しません。その代わりに、サブエレメント統計の収集は、 RUNSTATS コマンドの LIKE STATISTICS オプションまたは、 db2Runstats API の iColumnflags パラメーターの DB2RUNSTATS_COLUMN_LIKE_STATS 値によって制御されます。

症状: RUNSTATS コマンドの呼び出し、または db2Runstats API の呼び出しの後で、サブエレメント統計はシステム・カタログ内で -1 (デフォルト) に設定されます。これは以下のような照会を実行することで確認できます。

```
SELECT SUBSTR(TABSCHEMA,1,18), SUBSTR(TABNAME,1,18),
   SUBSTR(COLNAME,1,18), COLCARD, AVGCOLLEN, SUB_COUNT, SUB_DELIM_LENGTH
   FROM SYSSTAT.COLUMNS
   WHERE COLNAME IN ('P_TYPE', 'P_NAME')
   ORDER BY 1,2,3
```

(適当な列名で P_TYPE および P_NAME を置換)

列の結果で、COLCARD および AVGCOLLEN に負でない値があり、 SUB_COUNT および SUB_DELIM_LENGTH の値が -1 である場合、基本統計は列から収集されますが、サブエレメント統計は収集されないことを示します。

解決方法: バージョン 7 で DB2_LIKE_VARCHAR=?,Y (? は任意の値) を指定した場合、該当する列に対するこれらの統計を収集するためには、 RUNSTATS コマ

ンドで LIKE STATISTICS オプションを指定するか、 db2Runstats API で DB2RUNSTATS_COLUMN_LIKE_STATS オプションを指定する必要があります。

接続性と共存

下位レベル・サーバーのサポート

Windows	UNIX

変更点: バージョン 7 環境からバージョン 8 環境に移行するとき、すべてのサーバーをバージョン 8 に移行する前に、クライアント・マシンを 8 に移行する場合には、いくつかの制約事項および制限があります。これらの制約事項と制限は、DB2 Connect、zSeries、OS/390、 iSeries データベース・サーバーのいずれにも関連していません。

解決方法: バージョン 8 クライアントがバージョン 7 サーバーと連動する場合、バージョン 7 サーバー上で DRDA Application Server 機能を構成して使用可能にする必要があります。これを行う方法は、バージョン 7 の*インストールおよび構成補足* を参照してください。

既知の制約事項と制限を回避するためには、いずれかのクライアント・マシンをバージョン 8 に移行する前に、すべてのサーバーをバージョン 8 に移行する必要があります。それが不可能な場合は、バージョン 8 クライアントからバージョン 7 サーバーにアクセスするとき、以下のものがサポートされないことに注意してください。

- 以下のデータ・タイプ:
  - ラージ・オブジェクト (LOB) データ・タイプ
  - ユーザー定義特殊タイプ (UDT)
  - DATALINK データ・タイプ。
- 以下のセキュリティー機能:
  - 認証タイプ SERVER_ENCRYPT。
  - パスワードの変更。 DB2 UDB バージョン 7 サーバー上のパスワードを DB2 UDB バージョン 8 クライアントから変更することはできません。
- 以下の接続および通信プロトコル:
  - 接続ではなく ATTACH を必要とするインスタンス要求。
  - DB2 UDB バージョン 8 クライアントから DB2 UDB バージョン 7 サーバーへの ATTACH ステートメントはサポートされていません。
  - サポートされている唯一のネットワーク・プロトコルは TCP/IP です。
  - その他のネットワーク・プロトコル (SNA、NetBIOS、IPX/SPX など) はサポートされません。
- 以下のアプリケーション機能およびタスク:
  - DESCRIBE INPUT ステートメントはサポートされていません。ただし、ODBC/JDBC アプリケーションの場合は例外です。 ODBC/JDBC アプリケーションを実行する DB2 UDB バージョン 8 クライアントからの DB2 UDB バージョン 7 サーバーへのアクセスをサポートするには、その種のアクセスが必要なすべての DB2 UDB バージョン 7 サーバーに対して、 DESCRIBE

INPUT サポート用の修正を適用する必要があります。この修正は APAR IY30655 関連の修正ですが、 DB2 UDB バージョン 8 が一般の使用に供される前に入手することができます。 DB2 UDB の資料中の 『IBM と連絡をとる』 の項を参考に、 APAR IY30655 関連の修正を入手する方法を確認してください。 DESCRIBE INPUT ステートメントを使用すると、アプリケーション要求元は準備済みステートメントにおける入力パラメーター・マーカーについての記述を入手でき、パフォーマンスおよびユーザビリティーが向上します。CALL ステートメントの場合、これには、ストアード・プロシージャーの IN および INOUT パラメーターに関連したパラメーター・マーカーが含まれます。

– Result.getObject(1) を使用すると、 JDBC 仕様が必要とする Java Long データ・タイプの代わりに、BigDecimal が戻されます。 DB2 UDB バージョン 7 DRDA サーバーは、 DESCRIBE INPUT 要求に応答するとき、およびデータを取得するときに、 BIGINT を DEC(19,0) にマップします。これは、BIGINT が定義されていない DRDA レベルで DB2 UDB バージョン 7 サーバーが稼働するためです。
– 照会割り込みはサポートされていません。これは、割り込み API だけでなく、CLI/ODBC SQL_QUERY_TIMEOUT 接続属性にも影響します。
– 2 フェーズ・コミット。 DB2 UDB バージョン 8 クライアントが関与する整合トランザクションの使用時には、トランザクション・マネージャー・データベースとして DB2 UDB バージョン 7 サーバーを使用することはできません。 DB2 UDB バージョン 8 サーバーがトランザクション・マネージャー・データベースとなる可能性のある整合トランザクションにも、 DB2 UDB バージョン 7 サーバーは関与することはできません。
– XA 準拠のトランザクション・マネージャー。 DB2 UDB バージョン 8 クライアントを使用するアプリケーションは、 XA リソースとして DB2 UDB バージョン 7 サーバーを使用できません。これには、トランザクション管理の一部となっている WebSphere、 Microsoft COM+/MTS、BEA WebLogic などが含まれます。
– モニター。モニター関数は、DB2 UDB バージョン 8 クライアントから DB2 UDB バージョン 7 サーバーにはサポートされていません。
– ユーティリティー。以下の場合、クライアントからサーバーに向けて開始されるユーティリティーはサポートされません。
  1. クライアントが DB2 UDB バージョン 8 であり、サーバーが DB2 UDB バージョン 7 である。
  2. サイズが 32 KB を超える SQL ステートメント
– 照会割り込みはサポートされていません。これは、割り込み API だけでなく、CLI/ODBC SQL_QUERY_TIMEOUT 接続属性にも影響します。

DB2 UDB バージョン 7 サーバーと一緒に稼働する DB2 UDB バージョン 8 クライアントに対する制限事項と制約に加え、 DB2 UDB バージョン 7 サーバーと一緒に稼働する DB2 UDB バージョン 8 ツールにもそれに似た制約事項と制約があります。以下の DB2 UDB バージョン 8 ツールは、DB2 UDB バージョン 8 サーバーのみをサポートします。

- コントロール・センター
- タスク・センター

- ジャーナル
- サテライト管理センター
- インフォメーション・カタログ・センター (Web バージョンのセンターを含む)
- ヘルス・センター (Web バージョンのセンターを含む)
- ライセンス・センター
- Spatial Extender
- ツール設定
- デベロップメント・センター。ストアード・プロシージャー・ビルダーを使用して、バージョン 8 以前のサーバー上でサーバー・オブジェクトを開発する必要があります。

以下の DB2 UDB バージョン 8 ツールは、 DB2 UDB バージョン 7 サーバー (制約事項あり) と DB2 UDB バージョン 8 サーバーの両方をサポートします。

- 構成アシスタント

  DB2 UDB バージョン 7 サーバーを発見し、カタログすることができます。ただし、カタログされるとしても、 DB2 UDB バージョン 7 サーバーへのアクセスを試みる場合、何も動作しません。また、DB2 UDB バージョン 7 プロファイルを DB2 UDB バージョン 8 サーバーにインポートするか、または DB2 UDB バージョン 8 プロファイルを DB2 UDB バージョン 7 サーバーにインポートすることができます。ただし、その他すべての構成アシスタント機能は、 DB2 UDB バージョン 7 サーバーでは作動しません。
- データウェアハウス・センター
- レプリケーション・センター
- コマンド・エディター (コマンド・センターに替わるもの、コマンド・センターの Web バージョンを含む)

  DB2 UDB バージョン 7 サーバー間でのスクリプトのインポートおよび保管はできません。 ATTACH が必要なユーティリティーは動作しません。
- SQL Assist
- Visual Explain

一般的に言って、コントロール・センターのナビゲーション・ツリーからのみ立ち上げられる DB2 UDB バージョン 8 ツール、およびこれらのツールに基づく詳細ビューは、 DB2 UDB バージョン 7 以前のサーバーからアクセスおよび利用できません。 DB2 UDB バージョン 7 以前のサーバーで作業する場合は、 DB2 UDB バージョン 7 のツールを使用することを検討してみる必要があります。

両方向スクロール・カーソル・サポート

Windows	UNIX

変更点: バージョン 8 では、 DB2 UDB for Unix および for Windows バージョン 8 クライアントから DB2 UDB for Unix および for Windows バージョン 7 サーバーへの両方向スクロール・カーソル機能はサポートされません。両方向スクロール・カーソル機能がサポートされるのは、 DB2 UDB for Unix および for

Windows バージョン 8 クライアントから DB2 UDB for Unix および for Windows バージョン 8 サーバーへ、または DB2 UDB for OS/390 and z/OS サーバー 7 へ、という場合のみです。 DB2 UDB for Unix および for Windows バージョン 7 クライアントは、 DB2 UDB for Unix および for Windows バージョン 8 サーバーへの既存の両方向スクロール・カーソル機能を引き続きサポートします。

解決方法: サーバーをバージョン 8 にアップグレードします。

DB2 Connect バージョン 8 サーバーを介したバージョン 7 サーバー・アクセス

Windows	UNIX

変更点: バージョン 8 では、 DB2 UDB for Unix および for Windows クライアントからバージョン 7 DB2 UDB サーバーへのアクセスは、バージョン 8 サーバーを介してはサポートされません。バージョン 8 サーバーの機能は、 DB2 Connect Enterprise Edition バージョン 8 または DB2 UDB Enterprise Server Edition バージョン 8 によって提供されます。

解決方法: サーバーをバージョン 8 にアップグレードします。

CLP または組み込み SQL を使用したタイプ 1 接続

Windows	UNIX

変更点: DB2 UDB の以前のバージョンでは、コマンド行プロセッサー (CLP) または組み込み SQL を使用してタイプ 1 接続でデータベースに接続する場合、作業単位中に他のデータベースに接続しようとすると、 SQL0752N エラーで失敗しました。バージョン 8 では、作業単位がコミットされると接続はリセットされ、2 番目のデータベースへの接続が許可されます。作業単位はコミットされ、AUTOCOMMIT がオフの場合でも接続はリセットされます。

メッセージ

DB2 UDB メッセージの代わりに戻される DB2 Connect メッセージ

Windows	UNIX

変更点: バージョン 8 では、以前のリリースで DB2 UDB メッセージが戻されるような状況で、新たに DB2 Connect メッセージが戻される場合があります。

この変更の影響を受けるメッセージは、バインド、接続、またはセキュリティーのエラーに関連するものです。照会やその他の SQL 要求の SQL エラーには、この変更による影響はありません。

例:

- SQLCODE -1224 に代わって SQLCODE -30081 が戻されます
- SQLCODE -1403 に代わって SQLCODE -30082 が戻されます

- SQLCODE -4930 に代わって SQLCODE -30104 が戻されます

症状: DB2 UDB メッセージに対応するアプリケーション・コードは、以前のように動作しなくなる可能性があります。

構成パラメーター

使用されなくなったデータベース・マネージャー構成パラメーター

Windows	UNIX

変更点: 以下のデータベース・マネージャー構成パラメーターは、使用されなくなりました。

- *backbufsz*: 以前のバージョンでは、デフォルト・バッファー・サイズを使ってバックアップ操作を実行でき、 *backbufsz* の値がデフォルトと見なされました。バージョン 8 では、バックアップ・ユーティリティーの使用時に、バックアップ・バッファー・サイズを明示的に指定する必要があります。
- *dft_client_adpt*: DCE ディレクトリー・サービスはもはやサポートされません。
- *dft_client_comm*: DCE ディレクトリー・サービスはもはやサポートされません。
- *dir_obj_name*: DCE ディレクトリー・サービスはもはやサポートされません。
- *dir_path_name*: DCE ディレクトリー・サービスはもはやサポートされません。
- *dir_type*: DCE ディレクトリー・サービスはもはやサポートされません。
- *dos_rqrioblk*
- *drda_heap_sz*
- *fcm_num_anchors*、 *fcm_num_connect*、および *fcm_num_rqb*: DB2 UDB は新たにメッセージ・アンカー、接続項目、および要求ブロックを動的かつ自動的に調整するようになりました。これらのパラメーターを自分で調整する必要はありません。
- *fileserver*: IPX/SPX はもはやサポートされません。
- *initdari_jvm*: Java ストアード・プロシージャーは、デフォルトではマルチスレッドで実行されるようになりました。さらに、他の言語のルーチンとは別個のプロセスで実行されるようになったため、このパラメーターはもはやサポートされません。
- *ipx_socket*: IPX/SPX はもはやサポートされません。
- *jdk11_path*: これは、データベース・マネージャー構成パラメーター *jdk_path* に置き換わりました。
- *keepdari*: これは、データベース・マネージャー構成パラメーター *keepfenced* に置き換わりました。
- *max_logicagents*: これは、データベース・マネージャー構成パラメーター *max_connections* に置き換わりました。
- *maxdari*: これは、データベース・マネージャー構成パラメーター *fenced_pool* に置き換わりました。
- *num_initdaris*: これは、データベース・マネージャー構成パラメーター *num_initfenced* に置き換わりました。

- *objectname*: IPX/SPX はもはやサポートされません。
- *restbufsz*: 以前のバージョンでは、デフォルト・バッファー・サイズを使ってリストア操作を実行でき、 *restbufsz* の値がデフォルトと見なされました。バージョン 8 では、リストア・ユーティリティーの使用時に、リストア・バッファー・サイズを明示的に指定する必要があります。
- *route_obj_name*: DCE ディレクトリー・サービスはもはやサポートされません。
- *ss_logon*: これは OS/2 パラメーターであり、 OS/2 はもはやサポートされません。
- *udf_mem_sz*: UDF はもはや共有メモリーにデータを渡さないので、このパラメーターはサポートされません。

解決方法: アプリケーションから、これらのパラメーターの参照をすべて除去します。

使用されなくなったデータベース構成パラメーター

Windows	UNIX

変更点: 以下のデータベース構成パラメーターは、使用されなくなりました。

- *buffpage*: 以前のバージョンでは、デフォルト・サイズを使ってバッファー・プールを作成または変更でき、 *buffpage* の値がデフォルトと見なされました。バージョン 8 では、 ALTER BUFFERPOOL ステートメントまたは CREATE BUFFERPOOL ステートメントで SIZE キーワードを使用することによって、バッファー・プール・サイズを明示的に指定する必要があります。
- *copyprotect*
- *indexsort*

解決方法: アプリケーションから、これらのパラメーターの参照をすべて除去します。

バージョン 7 と以前のリリースとの非互換性

アプリケーション・プログラミング

Query Patroller Universal Client

WIN	UNIX	OS/2

変更点: このクライアント・アプリケーション・イネーブラー (CAE) の新しいバージョンは、新しいストアード・プロシージャーを含んでいるために、 Query Patroller Server バージョン 7 でのみ動作します。 CAE は DB2 Universal Database™ (DB2 UDB) へのアプリケーション・インターフェースであり、すべてのアプリケーションは最終的には CAE を通してデータベースにアクセスします。

症状: この CAE がバックレベルのサーバーに対して実行されると、メッセージ SQL29001 が戻されます。

オブジェクト変換関数および構造型

WIN	UNIX	OS/2

変更点: SQLDA が変更されたことが原因で、バージョン 7 以前のクライアントとバージョン 7 のサーバーとの間には、ごくわずかな非互換性がリモートに発生する可能性があります。 2 番目の SQLVAR の 8 番目のバイトには、 (値 X'00' および X'01' に加えて) 値 X'12' が使用可能になりました。新しい値を予期しないアプリケーションは、この拡張によって影響されるかもしれません。

解決方法: 将来のリリースでは、この分野で他にも拡張される点が出てくる可能性があるため、開発者は明示的に定義された値のみをテストするようお勧めします。

JVM の使用するクラスおよび jar ファイルのバージョン

WIN	UNIX	OS/2

変更点: これまでは、いったん Java ストアード・プロシージャーまたはユーザー定義関数 (UDF) が開始されると、 Java 仮想マシン (JVM) によって、CLASSPATH で指定されたすべてのファイル (`sqllib/function` の中のファイルを含む) がロックされました。 JVM は、データベース・マネージャーが停止するまでこのファイルを使用しました。ストアード・プロシージャーや UDF を実行する環境 (つまり、データベース・マネージャー構成パラメーター *keepdari* の値、およびストアード・プロシージャーが fenced されているかどうか) によっては、クラスをリフレッシュすることにより、データベース・マネージャーを停止せずにクラスおよび jar ファイルを置換できます。この点が以前の動作と異なります。

インストール、置換、および除去 jar コマンドの機能の変更

WIN	UNIX	OS/2

変更点: これまでは、jar のインストールの際、すべての DARI (データベース・アプリケーション・リモート・インターフェース) プロセスがフラッシュされていました。このようにして、新しいストアード・プロシージャー・クラスが次の呼び出しで使用されることが保証されていました。今後は、jar コマンドによって DARI プロセスがフラッシュされることはありません。新しくインストールまたは置換された jar からのクラスを使用するには、 SQLEJ.REFRESH_CLASSES コマンドを明示的に発行する必要があります。

DARI プロセスをフラッシュしないことによる別の非互換性として、 fenced されたストアード・プロシージャーの場合、データベース・マネージャー構成パラメーター *keepdari* の値が "YES" に設定されていれば、クライアントは異なるバージョンの jar ファイルを受け取る可能性があります。次のシナリオを考えてください。

1. ユーザー A は jar を置換しますが、クラスをリフレッシュしません。
2. 次にユーザー A は jar からストアード・プロシージャーを呼び出します。この呼び出しで同じ DARI プロセスが使われるとすれば、ユーザー A は jar ファイルの以前のバージョンを受け取ります。

3. ユーザー B は、同じストアード・プロシージャーを呼び出します。この呼び出しは、新しい DARI を使用します。つまり、新しく作成されたクラス・ローダーは jar ファイルの新しいバージョンを使用します。

言い換えると、jar 操作の後でクラスがリフレッシュされない場合は、使用される DARI プロセスに応じて、異なるバージョンの jar からのストアード・プロシージャーが呼び出される可能性があります。この点が、(DARI プロセスをフラッシュすることによって) 常に新しいクラスが使用されていた以前の動作と異なります。

32 ビット・アプリケーションの非互換性

	UNIX	

変更点: 32 ビット実行可能プログラム (DB2 UDB アプリケーション) は、新しい 64 ビット・データベース・エンジンに対しては実行されません。

症状: アプリケーションはリンクに失敗します。 32 ビット・オブジェクトを 64 ビット DB2 UDB アプリケーション・ライブラリーにリンクしようとすると、オペレーティング・システムのリンカー・エラー・メッセージが表示されます。

解決方法: アプリケーションを 64 ビット実行可能プログラムとして再コンパイルし、新しい 64 ビット DB2 UDB ライブラリーにリンクし直す必要があります。

スクラッチパッドの長さフィールドの変更

WIN	UNIX	OS/2

変更点: ユーザー定義関数 (UDF) で、渡されるスクラッチパッドの長さフィールドを変更すると、SQLCODE -450 が戻されるようになりました。

症状: スクラッチパッドの長さフィールドを変更する UDF は失敗します。呼び出しステートメントは SQLCODE -450 を受け取り、ここにスキーマおよび関数の固有の名前が示されます。

解決方法: スクラッチパッドの長さフィールドを変更しないように UDF 本文を書き直します。

SQL

スキーマ SESSION によって修飾される通常表を使用するアプリケーション

WIN	UNIX	OS/2

変更点: スキーマ SESSION は、一時表に使用できる唯一のスキーマです。 DB2 UDB はこれを使用して、SESSION 修飾表が一時表を参照する可能性があることを示すようになりました。ただし、SESSION は一時表のために予約されたキーワードではないため、正規基本表用のスキーマとして使用できます。このため、アプリケーションで、実表 SESSION.T1 と宣言済み一時表 SESSION.T1 とが同時に存在することが検出される可能性があります。パッケージのバインド時に、 (明示的また

は暗黙的に) ″SESSION″ で修飾されている表参照を含む静的ステートメントが見つかると、このステートメントのセクションや従属関係はカタログに保管されません。代わりに、このセクションをランタイムに増分的にバインドする必要があります。これによって、このセクションのコピーが動的 SQL のキャッシュに入ります。キャッシュに入れられたコピーは、アプリケーション・ユニークなインスタンス専用となります。ランタイムに、表名が一致する宣言済み一時表が存在する場合、たとえ同じ名前の永続基本表が存在しても、宣言済み一時表が使用されます。

症状: バージョン 6 以前では、 SESSION によって修飾される表を含む静的ステートメントのパッケージは、常に永続基本表を参照していました。パッケージをバインドする際には、セクションおよびそのステートメントに関連した従属関係レコードがカタログに保管されました。バージョン 7 では、これらのステートメントはバインド時にはバインドされず、ランタイムに同じ名前の宣言済み一時表に解決されます。したがって、次のような状況が生じる可能性があります。

- バージョン 5 からの移行。バージョン 5 でこのようなパッケージが存在する場合、これはバージョン 6 で再びバインドされ、静的ステートメントは増分的にバインドされます。このことは、パフォーマンスに影響を与えます。これは、増分的にバインドされたセクションが、キャッシュに入った動的 SQL のように動作するためです。ただし、キャッシュに入った動的セクションは、他のアプリケーション (同じアプリケーション実行可能プログラムの異なるインスタンスも含む) との間では共有できません。
- バージョン 6 からバージョン 7 への移行。バージョン 6 でこのようなパッケージが存在する場合、バージョン 7 で再びバインドする必要はありません。その代わり、ステートメントは依然として正規の静的 SQL として実行され、最初のバインド時にカタログに保管されたセクションを使用します。しかし、このパッケージが (明示的または暗黙的に) 再バインドされる場合、 SESSION 修飾表の参照を含むパッケージ内のステートメントはもはや保管されず、増分的バインドが必要となります。これにより、パフォーマンスが低下する可能性があります。

要約すると、バージョン 7 でバインドされたパッケージのうち、 SESSION 修飾表を参照する静的ステートメントを持つものは、増分的バインドが必要であるため、もはや静的 SQL のようには動作しません。実際、既存の SESSION 修飾表、ビュー、または別名と同じ名前を持つ表に関してアプリケーション・プロセスが DECLARE GLOBAL TEMPORARY TABLE ステートメントを発行すると、それらのオブジェクトへの参照は、すべて宣言済み一時表を参照するものと見なされます。

解決方法: 可能であれば、永続表のスキーマ名を ″SESSION″ 以外に変更します。それができなければ、パフォーマンスに与える影響について、および宣言済み一時表との競合について、意識しておく以外に方法はありません。

以下の照会を使用すれば、アプリケーションが一時表を使用するときに影響を受ける可能性のある表、ビュー、および別名を識別することができます。

```
select tabschema, tabname from SYSCAT.TABLES where tabschema = 'SESSION'
```

以下の照会を使用すれば、バージョン 7 でバウンドされたパッケージのうち、静的セクションがカタログに保管されており、パッケージの再バインド時に動作が変わる可能性のあるものを識別できます (これは、バージョン 6 からバージョン 7 に移行する場合にのみ関係があります)。

```
select pkgschema, pkgname, bschema, bname from syscat.packagedep
   where bschema = 'SESSION' and btype in ('T', 'V', 'I')
```

ユーティリティーとツール

AIX および Solaris での db2set

	UNIX	

変更点: コマンド db2set -ul (ユーザー・レベル) およびこれに関連する関数は、AIX および Solaris には移植されていません。

接続性と共存

32 ビット・クライアントの非互換性

WIN	UNIX	OS/2

変更点: 32 ビット・クライアントは、 64 ビット・サーバー上のインスタンスやデータベースには接続できません。

症状: クライアントおよびサーバーの両方がバージョン 7 のコードを実行している場合は、 SQL1434N が戻されます。それ以外の場合は、接続が SQLCODE -30081 を出して失敗します。

解決方法: 64 ビット・クライアントを使用します。

付録 B. 各国語サポート (NLS)

ここでは、DB2 Universal Database™ (DB2 UDB) で提供される各国語サポート (NLS) に関する情報を記載します。その中には、サポートされているテリトリー、言語、およびコード・ページ (コード・セット) に関する情報や、実際のデータベースおよびアプリケーションで DB2 UDB NLS 機能を構成および使用する方法などが含まれています。

各国語バージョン

DB2 Universal Database™ (DB2 UDB) バージョン 8 は、中国語 (簡体字)、中国語 (繁体字)、チェコ語、デンマーク語、英語、フィンランド語、フランス語、ドイツ語、イタリア語、日本語、韓国語、ノルウェー語、ポーランド語、ブラジル・ポルトガル語、ロシア語、スペイン語、およびスウェーデン語で入手できます。

DB2 UDB Run-Time Client は、さらにアラビア語、ブルガリア語、クロアチア語、オランダ語、ギリシャ語、ヘブライ語、ハンガリー語、ポルトガル語、ルーマニア語、スロバキア語、スロベニア語、およびトルコ語で利用可能です。

関連資料:

- 273 ページの『サポートされているテリトリー・コードおよびコード・ページ』

サポートされているテリトリー・コードおよびコード・ページ

以下の表は、データベース・サーバーがサポートする言語とコード・セット、およびこれらの値がデータベース・マネージャーで使用されるテリトリー・コードとコード・ページ値にどのようにマップされるかを示したものです。

次に、この表の列を説明します。

- **コード・ページ**は、オペレーティング・システムのコード・セットからマップされる IBM 定義のコード・ページを示します。
- **グループ**は、コード・ページがシングルバイト (″S″)、ダブルバイト (″D″)、またはニュートラル (″N″) のいずれかを示します。「-n」は、文字番号の組み合わせを作成するのに使われる番号です。一致する組み合わせは、 DB2 Universal Database™ (DB2 UDB) で接続と変換が可能なところを示しています。たとえば、「S-1」グループすべては、一緒に動作できます。ただし、グループがニュートラルの場合、以下にリストするほかのどのようなコード・ページを使用した接続および変換も可能です。
- **コード・セット**は、サポートされる言語に関連するコード・セットを示します。このコード・セットは、DB2 UDB コード・ページにマップされます。
- **テリトリー・コード**は、データベース・マネージャーが地域固有のサポートを提供するために内部で使用するものです。
- **ロケール**は、データベース・マネージャーがサポートするロケール値を示します。

- **オペレーティング・システム** は、言語およびコード・セットをサポートするオペレーティング・システムを示します。

表 38. Unicode

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1200	N-1	16 ビット Unicode	任意	任意	任意
1208	N-1	Unicode の UTF-8 エンコード	任意	任意	任意

表 39. アルバニア、テリトリー ID: AL

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	355	sq_AL	AIX
850	S-1	IBM-850	355	-	AIX
923	S-1	ISO8859-15	355	sq_AL.8859-15	AIX
1208	N-1	UTF-8	355	SQ_AL	AIX
37	S-1	IBM-37	355	-	ホスト
1140	S-1	IBM-1140	355	-	ホスト
819	S-1	iso88591	355	-	HP-UX
923	S-1	iso885915	355	-	HP-UX
1051	S-1	roman8	355	-	HP-UX
437	S-1	IBM-437	355	-	OS/2
850	S-1	IBM-850	355	-	OS/2
819	S-1	ISO8859-1	355	-	Solaris
923	S-1	ISO8859-15	355	-	Solaris
1252	S-1	1252	355	-	Windows

表 40. アラブ諸国/地域、テリトリー ID: AA

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1046	S-6	IBM-1046	785	Ar_AA	AIX
1089	S-6	ISO8859-6	785	ar_AA	AIX
1208	N-1	UTF-8	785	AR_AA	AIX
420	S-6	IBM-420	785	-	ホスト
425	S-6	IBM-425	785	-	ホスト
1089	S-6	iso88596	785	ar_SA.iso88596	HP-UX
864	S-6	IBM-864	785	-	OS/2
1256	S-6	1256	785	-	Windows

表 41. オーストラリア、テリトリー ID: AU

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	61	en_AU	AIX

表 41. オーストラリア、テリトリー ID: AU (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
850	S-1	IBM-850	61	-	AIX
923	S-1	ISO8859-15	61	en_AU.8859-15	AIX
1208	N-1	UTF-8	61	EN_AU	AIX
37	S-1	IBM-37	61	-	ホスト
1140	S-1	IBM-1140	61	-	ホスト
819	S-1	iso88591	61	-	HP-UX
923	S-1	iso885915	61	-	HP-UX
1051	S-1	roman8	61	-	HP-UX
437	S-1	IBM-437	61	-	OS/2
850	S-1	IBM-850	61	-	OS/2
819	S-1	ISO8859-1	61	en_AU	SCO
819	S-1	ISO8859-1	61	en_AU	Solaris
923	S-1	ISO8859-15	61	-	Solaris
1252	S-1	1252	61	-	Windows

表 42. オーストリア、テリトリー ID: AT

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	43	-	AIX
850	S-1	IBM-850	43	-	AIX
923	S-1	ISO8859-15	43	-	AIX
1208	N-1	UTF-8	43	-	AIX
37	S-1	IBM-37	43	-	ホスト
1140	S-1	IBM-1140	43	-	ホスト
819	S-1	iso88591	43	-	HP-UX
923	S-1	iso885915	43	-	HP-UX
1051	S-1	roman8	43	-	HP-UX
819	S-1	ISO-8859-1	43	de_AT	Linux
923	S-1	ISO-8859-15	43	de_AT@euro	Linux
437	S-1	IBM-437	43	-	OS/2
850	S-1	IBM-850	43	-	OS/2
819	S-1	ISO8859-1	43	de_AT	SCO
819	S-1	ISO8859-1	43	de_AT	Solaris
923	S-1	ISO8859-15	43	de_AT.ISO8859-15	Solaris
1252	S-1	1252	43	-	Windows

表 43. ベラルーシ、テリトリー ID: BY

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1167	S-5	KOI8-RU	375	-	-
915	S-5	ISO8859-5	375	be_BY	AIX
1208	N-1	UTF-8	375	BE_BY	AIX
1025	S-5	IBM-1025	375	-	ホスト
1154	S-5	IBM-1154	375	-	ホスト
915	S-5	ISO8859-5	375	-	OS/2
1131	S-5	IBM-1131	375	-	OS/2

表 43. ベラルーシ、テリトリー ID: BY (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1251	S-5	1251	375	-	Windows

表 44. ベルギー、テリトリー ID: BE

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	32	fr_BE	AIX
819	S-1	ISO8859-1	32	nl_BE	AIX
850	S-1	IBM-850	32	Fr_BE	AIX
850	S-1	IBM-850	32	Nl_BE	AIX
923	S-1	ISO8859-15	32	fr_BE.8859-15	AIX
923	S-1	ISO8859-15	32	nl_BE.8859-15	AIX
1208	N-1	UTF-8	32	FR_BE	AIX
1208	N-1	UTF-8	32	NL_BE	AIX
274	S-1	IBM-274	32	-	ホスト
500	S-1	IBM-500	32	-	ホスト
1148	S-1	IBM-1148	32	-	ホスト
819	S-1	iso88591	32	-	HP-UX
923	S-1	iso885915	32	-	HP-UX
819	S-1	ISO-8859-1	32	fr_BE	Linux
819	S-1	ISO-8859-1	32	nl_BE	Linux
923	S-1	ISO-8859-15	32	fr_BE@euro	Linux
923	S-1	ISO-8859-15	32	nl_BE@euro	Linux
437	S-1	IBM-437	32	-	OS/2
850	S-1	IBM-850	32	-	OS/2
819	S-1	ISO8859-1	32	fr_BE	SCO
819	S-1	ISO8859-1	32	nl_BE	SCO
819	S-1	ISO8859-1	32	fr_BE	Solaris
819	S-1	ISO8859-1	32	nl_BE	Solaris
923	S-1	ISO8859-15	32	fr_BE.ISO8859-15	Solaris
923	S-1	ISO8859-15	32	nl_BE.ISO8859-15	Solaris
1252	S-1	1252	32	-	Windows

表 45. ブルガリア、テリトリー ID: BG

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
915	S-5	ISO8859-5	359	bg_BG	AIX
1208	N-1	UTF-8	359	BG_BG	AIX
1025	S-5	IBM-1025	359	-	ホスト
1154	S-5	IBM-1154	359	-	ホスト
915	S-5	iso88595	359	bg_BG.iso88595	HP-UX
855	S-5	IBM-855	359	-	OS/2
915	S-5	ISO8859-5	359	-	OS/2
1251	S-5	1251	359	-	Windows

表 46. ブラジル、テリトリー ID: BR

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	55	pt_BR	AIX
850	S-1	IBM-850	55	-	AIX
923	S-1	ISO8859-15	55	pt_BR.8859-15	AIX
1208	N-1	UTF-8	55	PT_BR	AIX
37	S-1	IBM-37	55	-	ホスト
1140	S-1	IBM-1140	55	-	ホスト
819	S-1	ISO8859-1	55	-	HP-UX
923	S-1	ISO8859-15	55	-	HP-UX
819	S-1	ISO-8859-1	55	pt_BR	Linux
923	S-1	ISO-8859-15	55	-	Linux
850	S-1	IBM-850	55	-	OS/2
819	S-1	ISO8859-1	55	pt_BR	SCO
819	S-1	ISO8859-1	55	pt_BR	Solaris
923	S-1	ISO8859-15	55	-	Solaris
1252	S-1	1252	55	-	Windows

表 47. カナダ、テリトリー ID: CA

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	1	fr_CA	AIX
850	S-1	IBM-850	1	Fr_CA	AIX
923	S-1	ISO8859-15	1	fr_CA.8859-15	AIX
1208	N-1	UTF-8	1	FR_CA	AIX
37	S-1	IBM-37	1	-	ホスト
1140	S-1	IBM-1140	1	-	ホスト
819	S-1	iso88591	1	fr_CA.iso88591	HP-UX
923	S-1	iso885915	1	-	HP-UX
1051	S-1	roman8	1	fr_CA.roman8	HP-UX
819	S-1	ISO-8859-1	1	en_CA	Linux
923	S-1	ISO-8859-15	1	-	Linux
850	S-1	IBM-850	1	-	OS/2
819	S-1	ISO8859-1	1	en_CA	SCO
819	S-1	ISO8859-1	1	fr_CA	SCO
819	S-1	ISO8859-1	1	en_CA	Solaris
923	S-1	ISO8859-15	1	-	Solaris
1252	S-1	1252	1	-	Windows

表 48. カナダ (フランス語)、テリトリー ID: CA

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
863	S-1	IBM-863	2	-	OS/2

表 49. 中国 (PRC)、テリトリー ID: CN

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1383	D-4	IBM-eucCN	86	zh_CN	AIX
1386	D-4	GBK	86	Zh_CN.GBK	AIX
1208	N-1	UTF-8	86	ZH_CN	AIX
935	D-4	IBM-935	86	-	ホスト
1388	D-4	IBM-1388	86	-	ホスト
1383	D-4	hp15CN	86	zh_CN.hp15CN	HP-UX
1383	D-4	GBK	86	zh_CN.GBK	Linux
1381	D-4	IBM-1381	86	-	OS/2
1386	D-4	GBK	86	-	OS/2
1383	D-4	eucCN	86	zh_CN	SCO
1383	D-4	eucCN	86	zh_CN.eucCN	SCO
1383	D-4	gb2312	86	zh	Solaris
1208	N-1	UTF-8	86	zh.UTF-8	Solaris
1381	D-4	IBM-1381	86	-	Windows
1386	D-4	GBK	86	-	Windows
1392/5488	D-4		86	-	

1 (294 ページ) を参照してください。

表 50. クロアチア、テリトリー ID: HR

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
912	S-2	ISO8859-2	385	hr_HR	AIX
1208	N-1	UTF-8	385	HR_HR	AIX
870	S-2	IBM-870	385	-	ホスト
1153	S-2	IBM-1153	385	-	ホスト
912	S-2	iso88592	385	hr_HR.iso88592	HP-UX
912	S-2	ISO-8859-2	385	hr_HR	Linux
852	S-2	IBM-852	385	-	OS/2
912	S-2	ISO8859-2	385	hr_HR.ISO8859-2	SCO
1250	S-2	1250	385	-	Windows

表 51. チェコ共和国、テリトリー ID: CZ

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
912	S-2	ISO8859-2	421	cs_CZ	AIX
1208	N-1	UTF-8	421	CS_CZ	AIX
870	S-2	IBM-870	421	-	ホスト
1153	S-2	IBM-1153	421	-	ホスト
912	S-2	iso88592	421	cs_CZ.iso88592	HP-UX
912	S-2	ISO-8859-2	421	cs_CZ	Linux
852	S-2	IBM-852	421	-	OS/2
912	S-2	ISO8859-2	421	cs_CZ.ISO8859-2	SCO
1250	S-2	1250	421	-	Windows

表 52. デンマーク、テリトリー ID: DK

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	45	da_DK	AIX
850	S-1	IBM-850	45	Da_DK	AIX
923	S-1	ISO8859-15	45	da_DK.8859-15	AIX
1208	N-1	UTF-8	45	DA_DK	AIX
277	S-1	IBM-277	45	-	ホスト
1142	S-1	IBM-1142	45	-	ホスト
819	S-1	iso88591	45	da_DK.iso88591	HP-UX
923	S-1	iso885915	45	_	HP-UX
1051	S-1	roman8	45	da_DK.roman8	HP-UX
819	S-1	ISO-8859-1	45	da_DK	Linux
923	S-1	ISO-8859-15	45	-	Linux
850	S-1	IBM-850	45	-	OS/2
819	S-1	ISO8859-1	45	da	SCO
819	S-1	ISO8859-1	45	da_DA	SCO
819	S-1	ISO8859-1	45	da_DK	SCO
819	S-1	ISO8859-1	45	da	Solaris
923	S-1	ISO8859-15	45	da.ISO8859-15	Solaris
1252	S-1	1252	45	-	Windows

表 53. エストニア、テリトリー ID: EE

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
922	S-10	IBM-922	372	Et_EE	AIX
1208	N-1	UTF-8	372	ET_EE	AIX
1122	S-10	IBM-1122	372	-	ホスト
1157	S-10	IBM-1157	372	-	ホスト
922	S-10	IBM-922	372	-	OS/2
1257	S-10	1257	372	-	Windows

表 54. フィンランド、テリトリー ID: FI

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	358	fi_FI	AIX
850	S-1	IBM-850	358	Fi_FI	AIX
923	S-1	ISO8859-15	358	fi_FI.8859-15	AIX
1208	N-1	UTF-8	358	FI_FI	AIX
278	S-1	IBM-278	358	-	ホスト
1143	S-1	IBM-1143	358	-	ホスト
819	S-1	iso88591	358	fi_FI.iso88591	HP-UX
923	S-1	iso885915	358	-	HP-UX
1051	S-1	roman8	358	fi-FI.roman8	HP-UX
819	S-1	ISO-8859-1	358	fi_FI	Linux
923	S-1	ISO-8859-15	358	fi_FI@euro	Linux
437	S-1	IBM-437	358	-	OS/2
850	S-1	IBM-850	358	-	OS/2
819	S-1	ISO8859-1	358		SCO

表 54. フィンランド、テリトリー ID: FI (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	358	fi_FI	SCO
819	S-1	ISO8859-1	358	sv_FI	SCO
819	S-1	ISO8859-1	358	-	Solaris
923	S-1	ISO8859-15	358	fi.ISO8859-15	Solaris
1252	S-1	1252	358	-	Windows

表 55. FYR マケドニア、テリトリー ID: MK

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
915	S-5	ISO8859-5	389	mk_MK	AIX
1208	N-1	UTF-8	389	MK_MK	AIX
1025	S-5	IBM-1025	389	-	ホスト
1154	S-5	IBM-1154	389	-	ホスト
915	S-5	iso88595	389	-	HP-UX
855	S-5	IBM-855	389	-	OS/2
915	S-5	ISO8859-5	389	-	OS/2
1251	S-5	1251	389	-	Windows

表 56. フランス、テリトリー ID: FR

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	33	fr_FR	AIX
850	S-1	IBM-850	33	Fr_FR	AIX
923	S-1	ISO8859-15	33	fr_FR.8859-15	AIX
1208	N-1	UTF-8	33	FR_FR	AIX
297	S-1	IBM-297	33	-	ホスト
1147	S-1	IBM-1147	33	-	ホスト
819	S-1	iso88591	33	fr_FR.iso88591	HP-UX
923	S-1	iso885915	33	-	HP-UX
1051	S-1	roman8	33	fr_FR.roman8	HP-UX
819	S-1	ISO-8859-1	33	fr_FR	Linux
923	S-1	ISO-8859-15	33	fr_FR@euro	Linux
437	S-1	IBM-437	33	-	OS/2
850	S-1	IBM-850	33	-	OS/2
819	S-1	ISO8859-1	33		SCO
819	S-1	ISO8859-1	33	fr_FR	SCO
819	S-1	ISO8859-1	33		Solaris
923	S-1	ISO8859-15	33	fr.ISO8859-15	Solaris
1208	N-1	UTF-8	33	fr.UTF-8	Solaris
1252	S-1	1252	33	-	Windows

表 57. ドイツ、テリトリー ID: DE

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	49	de_DE	AIX
850	S-1	IBM-850	49	De_DE	AIX
923	S-1	ISO8859-15	49	de_DE.8859-15	AIX
1208	N-1	UTF-8	49	DE_DE	AIX
273	S-1	IBM-273	49	-	ホスト
1141	S-1	IBM-1141	49	-	ホスト
819	S-1	iso88591	49	de_DE.iso88591	HP-UX
923	S-1	iso885915	49	_	HP-UX
1051	S-1	roman8	49	de_DE.roman8	HP-UX
819	S-1	ISO-8859-1	49	de_DE	Linux
923	S-1	ISO-8859-15	49	de_DE@euro	Linux
437	S-1	IBM-437	49	-	OS/2
850	S-1	IBM-850	49	-	OS/2
819	S-1	ISO8859-1	49		SCO
819	S-1	ISO8859-1	49	de_DE	SCO
819	S-1	ISO8859-1	49		Solaris
923	S-1	ISO8859-15	49	de.ISO8859-15	Solaris
1208	N-1	UTF-8	49	de.UTF-8	Solaris
1252	S-1	1252	49	-	Windows

表 58. ギリシャ、テリトリー ID: GR

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
813	S-7	ISO8859-7	30	el_GR	AIX
1208	N-1	UTF-8	30	EL_GR	AIX
423	S-7	IBM-423	30	-	ホスト
875	S-7	IBM-875	30	-	ホスト
813	S-7	iso88597	30	el_GR.iso88597	HP-UX
813	S-7	ISO-8859-7	30	el_GR	Linux
813	S-7	ISO8859-7	30	-	OS/2
869	S-7	IBM-869	30	-	OS/2
813	S-7	ISO8859-7	30	el_GR.ISO8859-7	SCO
737	S-7	737	30	-	Windows
1253	S-7	1253	30	-	Windows

表 59. ハンガリー、テリトリー ID: HU

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
912	S-2	ISO8859-2	36	hu_HU	AIX
1208	N-1	UTF-8	36	HU_HU	AIX
870	S-2	IBM-870	36	-	ホスト
1153	S-2	IBM-1153	36	-	ホスト
912	S-2	iso88592	36	hu_HU.iso88592	HP-UX
912	S-2	ISO-8859-2	36	hu_HU	Linux
852	S-2	IBM-852	36	-	OS/2
912	S-2	ISO8859-2	36	hu_HU.ISO8859-2	SCO

表 59. ハンガリー、テリトリー ID: HU (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1250	S-2	1250	36	-	Windows

表 60. アイスランド、テリトリー ID: IS

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	354	is_IS	AIX
850	S-1	IBM-850	354	Is_IS	AIX
923	S-1	ISO8859-15	354	is_IS.8859-15	AIX
1208	N-1	UTF-8	354	IS_IS	AIX
871	S-1	IBM-871	354	-	ホスト
1149	S-1	IBM-1149	354	-	ホスト
819	S-1	iso88591	354	is_IS.iso88591	HP-UX
923	S-1	iso885915	354	-	HP-UX
1051	S-1	roman8	354	is_IS.roman8	HP-UX
819	S-1	ISO-8859-1	354	is_IS	Linux
923	S-1	ISO-8859-15	354	-	Linux
850	S-1	IBM-850	354	-	OS/2
819	S-1	ISO8859-1	354		SCO
819	S-1	ISO8859-1	354	is_IS	SCO
819	S-1	ISO8859-1	354	-	Solaris
923	S-1	ISO8859-15	354	-	Solaris
1252	S-1	1252	354	-	Windows

表 61. インド、テリトリー ID: IN

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
806	S-13	IBM-806	91	hi_IN	-
1137	S-13	IBM-1137	91	-	ホスト

表 62. インドネシア、テリトリー ID: ID

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1252	S-1	1252	62	-	Windows

表 63. アイルランド、テリトリー ID: IE

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	353	-	AIX
850	S-1	IBM-850	353	-	AIX
923	S-1	ISO8859-15	353	-	AIX
1208	N-1	UTF-8	353	-	AIX

表 63. アイルランド、テリトリー ID: IE (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
285	S-1	IBM-285	353	-	ホスト
1146	S-1	IBM-1146	353	-	ホスト
819	S-1	iso88591	353	-	HP-UX
923	S-1	iso885915	353	-	HP-UX
1051	S-1	roman8	353	-	HP-UX
819	S-1	ISO-8859-1	353	en_IE	Linux
923	S-1	ISO-8859-15	353	en_IE@euro	Linux
437	S-1	IBM-437	353	-	OS/2
850	S-1	IBM-850	353	-	OS/2
819	S-1	ISO8859-1	353	en_IE.ISO8859-1	SCO
819	S-1	ISO8859-1	353	en_IE	Solaris
923	S-1	ISO8859-15	353	en_IE.ISO8859-15	Solaris
1252	S-1	1252	353	-	Windows

表 64. イスラエル、テリトリー ID: IL

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
856	S-8	IBM-856	972	Iw_IL	AIX
916	S-8	ISO8859-8	972	iw_IL	AIX
1208	N-1	UTF-8	972	HE-IL	AIX
916	S-8	ISO-8859-8	972	iw_IL	Linux
424	S-8	IBM-424	972	-	ホスト
862	S-8	IBM-862	972	-	OS/2
1255	S-8	1255	972	-	Windows

表 65. イタリア、テリトリー ID: IT

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	39	it_IT	AIX
850	S-1	IBM-850	39	It_IT	AIX
923	S-1	ISO8859-15	39	it_IT.8859-15	AIX
1208	N-1	UTF-8	39	It_IT	AIX
280	S-1	IBM-280	39	-	ホスト
1144	S-1	IBM-1144	39	-	ホスト
819	S-1	iso88591	39	it_IT.iso88591	HP-UX
923	S-1	iso885915	39	_	HP-UX
1051	S-1	roman8	39	it_IT.roman8	HP-UX
819	S-1	ISO-8859-1	39	it_IT	Linux
923	S-1	ISO-8859-15	39	it_IT@euro	Linux
437	S-1	IBM-437	39	-	OS/2
850	S-1	IBM-850	39	-	OS/2
819	S-1	ISO8859-1	39		SCO
819	S-1	ISO8859-1	39	it_IT	SCO
819	S-1	ISO8859-1	39		Solaris
923	S-1	ISO8859-15	39	it.ISO8859-15	Solaris
1208	N-1	UTF-8	39	it.UTF-8	Solaris

表 65. イタリア、テリトリー ID: IT (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1252	S-1	1252	39	-	Windows

表 66. 日本、テリトリー ID: JP

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
932	D-1	IBM-932	81	Ja_JP	AIX
943	D-1	IBM-943	81	Ja_JP	AIX
2 (294 ページ) を参照してください。					
954	D-1	IBM-eucJP	81	ja_JP	AIX
1208	N-1	UTF-8	81	JA_JP	AIX
930	D-1	IBM-930	81	-	ホスト
939	D-1	IBM-939	81	-	ホスト
5026	D-1	IBM-5026	81	-	ホスト
5035	D-1	IBM-5035	81	-	ホスト
1390	D-1		81	-	ホスト
1399	D-1		81	-	ホスト
954	D-1	eucJP	81	ja_JP.eucJP	HP-UX
5039	D-1	SJIS	81	ja_JP.SJIS	HP-UX
954	D-1	EUC-JP	81	ja_JP	Linux
932	D-1	IBM-932	81	-	OS/2
942	D-1	IBM-942	81	-	OS/2
943	D-1	IBM-943	81	-	OS/2
954	D-1	eucJP	81	ja	SCO
954	D-1	eucJP	81	ja_JP	SCO
954	D-1	eucJP	81	ja_JP.EUC	SCO
954	D-1	eucJP	81	ja_JP.eucJP	SCO
943	D-1	IBM-943	81	ja_JP.PCK	Solaris
954	D-1	eucJP	81	ja	Solaris
954	D-1	eucJP	81	japanese	Solaris
1208	N-1	UTF-8	81	ja_JP.UTF-8	Solaris
943	D-1	IBM-943	81	-	Windows
1394	D-1		81	-	
3 (294 ページ) を参照してください。					

表 67. カザフスタン、テリトリー ID: KZ

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1251	S-5	1251	7	-	Windows

表 68. 韓国、テリトリー ID: KR

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
970	D-3	IBM-eucKR	82	ko_KR	AIX

表 68. 韓国、テリトリー ID: KR (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1208	N-1	UTF-8	82	KO_KR	AIX
933	D-3	IBM-933	82	-	ホスト
1364	D-3	IBM-1364	82	-	ホスト
970	D-3	eucKR	82	ko_KR.eucKR	HP-UX
970	D-3	EUC-KR	82	ko_KR	Linux
949	D-3	IBM-949	82	-	OS/2
970	D-3	eucKR	82	ko_KR.eucKR	SGI
970	D-3	5601	82	ko	Solaris
1208	N-1	UTF-8	82	ko.UTF-8	Solaris
1363	D-3	1363	82	-	Windows

表 69. ラテンアメリカ、テリトリー ID: Lat

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	3	-	AIX
850	S-1	IBM-850	3	-	AIX
923	S-1	ISO8859-15	3	-	AIX
1208	N-1	UTF-8	3	-	AIX
284	S-1	IBM-284	3	-	ホスト
1145	S-1	IBM-1145	3	-	ホスト
819	S-1	iso88591	3	-	HP-UX
923	S-1	iso885915	3	-	HP-UX
1051	S-1	roman8	3	-	HP-UX
819	S-1	ISO-8859-1	3	-	Linux
923	S-1	ISO-8859-15	3	-	Linux
437	S-1	IBM-437	3	-	OS/2
850	S-1	IBM-850	3	-	OS/2
819	S-1	ISO8859-1	3	-	Solaris
923	S-1	ISO8859-15	3	-	Solaris
1252	S-1	1252	3	-	Windows

表 70. ラトビア、テリトリー ID: LV

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
921	S-10	IBM-921	371	Lv_LV	AIX
1208	N-1	UTF-8	371	LV_LV	AIX
1112	S-10	IBM-1112	371	-	ホスト
1156	S-10	IBM-1156	371	-	ホスト
921	S-10	IBM-921	371	-	OS/2
1257	S-10	1257	371	-	Windows

表 71. リトアニア、テリトリー ID: LT

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
921	S-10	IBM-921	370	Lt_LT	AIX
1208	N-1	UTF-8	370	LT_LT	AIX
1112	S-10	IBM-1112	370	-	ホスト
1156	S-10	IBM-1156	370	-	ホスト
921	S-10	IBM-921	370	-	OS/2
1257	S-10	1257	370	-	Windows

表 72. マレーシア、テリトリー ID: ID

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1252	S-1	1252	60	-	Windows

表 73. オランダ、テリトリー ID: NL

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	31	nl_NL	AIX
850	S-1	IBM-850	31	Nl_NL	AIX
923	S-1	ISO8859-15	31	nl_NL.8859-15	AIX
1208	N-1	UTF-8	31	NL_NL	AIX
37	S-1	IBM-37	31	-	ホスト
1140	S-1	IBM-1140	31	-	ホスト
819	S-1	iso88591	31	nl_NL.iso88591	HP-UX
923	S-1	iso885915	31	_	HP-UX
1051	S-1	roman8	31	nl_NL.roman8	HP-UX
819	S-1	ISO-8859-1	31	nl_NL	Linux
923	S-1	ISO-8859-15	31	nl_NL@euro	Linux
437	S-1	IBM-437	31	-	OS/2
850	S-1	IBM-850	31	-	OS/2
819	S-1	ISO8859-1	31	nl	SCO
819	S-1	ISO8859-1	31	nl_NL	SCO
819	S-1	ISO8859-1	31	nl	Solaris
923	S-1	ISO8859-15	31	nl.ISO8859-15	Solaris
1252	S-1	1252	31	-	Windows

表 74. ニュージーランド、テリトリー ID: NZ

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	64	-	AIX
850	S-1	IBM-850	64	-	AIX
923	S-1	ISO8859-15	64	-	AIX
1208	N-1	UTF-8	64	-	AIX
37	S-1	IBM-37	64	-	ホスト
1140	S-1	IBM-1140	64	-	ホスト

表 74. ニュージーランド、テリトリー ID: NZ (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	64	-	HP-UX
923	S-1	ISO8859-15	64	-	HP-UX
850	S-1	IBM-850	64	-	OS/2
819	S-1	ISO8859-1	64	en_NZ	SCO
819	S-1	ISO8859-1	64	en_NZ	Solaris
923	S-1	ISO8859-15	64	-	Solaris
1252	S-1	1252	64	-	Windows

表 75. ノルウェー、テリトリー ID: NO

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	47	no_NO	AIX
850	S-1	IBM-850	47	No_NO	AIX
923	S-1	ISO8859-15	47	no_NO.8859-15	AIX
1208	N-1	UTF-8	47	NO_NO	AIX
277	S-1	IBM-277	47	-	ホスト
1142	S-1	IBM-1142	47	-	ホスト
819	S-1	iso88591	47	no_NO.iso88591	HP-UX
923	S-1	iso885915	47	-	HP-UX
1051	S-1	roman8	47	no_NO.roman8	HP-UX
819	S-1	ISO-8859-1	47	no_NO	Linux
923	S-1	ISO-8859-15	47	-	Linux
850	S-1	IBM-850	47	-	OS/2
819	S-1	ISO8859-1	47	no	SCO
819	S-1	ISO8859-1	47	no_NO	SCO
819	S-1	ISO8859-1	47	no	Solaris
923	S-1	ISO8859-15	47	-	Solaris
1252	S-1	1252	47	-	Windows

表 76. ポーランド、テリトリー ID: PL

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
912	S-2	ISO8859-2	48	pl_PL	AIX
1208	N-1	UTF-8	48	PL_PL	AIX
870	S-2	IBM-870	48	-	ホスト
1153	S-2	IBM-1153	48	-	ホスト
912	S-2	iso88592	48	pl_PL.iso88592	HP-UX
912	S-2	ISO-8859-2	48	pl_PL	Linux
852	S-2	IBM-852	48	-	OS/2
912	S-2	ISO8859-2	48	pl_PL.ISO8859-2	SCO
1250	S-2	1250	48	-	Windows

表 77. ポルトガル、テリトリー *ID: PT*

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	351	pt_PT	AIX
850	S-1	IBM-850	351	Pt_PT	AIX
923	S-1	ISO8859-15	351	pt_PT.8859-15	AIX
1208	N-1	UTF-8	351	PT_PT	AIX
37	S-1	IBM-37	351	-	ホスト
1140	S-1	IBM-1140	351	-	ホスト
819	S-1	iso88591	351	pt_PT.iso88591	HP-UX
923	S-1	iso885915	351	-	HP-UX
1051	S-1	roman8	351	pt_PT.roman8	HP-UX
819	S-1	ISO-8859-1	351	pt_PT	Linux
923	S-1	ISO-8859-15	351	pt_PT@euro	Linux
850	S-1	IBM-850	351	-	OS/2
860	S-1	IBM-860	351	-	OS/2
819	S-1	ISO8859-1	351	pt	SCO
819	S-1	ISO8859-1	351	pt_PT	SCO
819	S-1	ISO8859-1	351	pt	Solaris
923	S-1	ISO8859-15	351	pt.ISO8859-15	Solaris
1252	S-1	1252	351	-	Windows

表 78. ルーマニア、テリトリー *ID: RO*

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
912	S-2	ISO8859-2	40	ro_RO	AIX
1208	N-1	UTF-8	40	RO_RO	AIX
870	S-2	IBM-870	40	-	ホスト
1153	S-2	IBM-1153	40	-	ホスト
912	S-2	iso88592	40	ro_RO.iso88592	HP-UX
912	S-2	ISO-8859-2	40	ro_RO	Linux
852	S-2	IBM-852	40	-	OS/2
912	S-2	ISO8859-2	40	ro_RO.ISO8859-2	SCO
1250	S-2	1250	40	-	Windows

表 79. ロシア、テリトリー *ID: RU*

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
915	S-5	ISO8859-5	7	ru_RU	AIX
1208	N-1	UTF-8	7	RU_RU	AIX
1025	S-5	IBM-1025	7	-	ホスト
1154	S-5	IBM-1154	7	-	ホスト
915	S-5	iso88595	7	ru_RU.iso88595	HP-UX
878	S-5	KOI8-R	7	ru_RU.koi8-r	Linux、Solaris
915	S-5	ISO-8859-5	7	ru_RU	Linux
866	S-5	IBM-866	7	-	OS/2
915	S-5	ISO8859-5	7	-	OS/2
915	S-5	ISO8859-5	7	ru_RU.ISO8859-5	SCO

表 79. ロシア、テリトリー ID: RU (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1251	S-5	1251	7	-	Windows

表 80. セルビア/モンテネグロ、テリトリー ID: SP

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
915	S-5	ISO8859-5	381	sr_SP	AIX
1208	N-1	UTF-8	381	SR_SP	AIX
1025	S-5	IBM-1025	381	-	ホスト
1154	S-5	IBM-1154	381	-	ホスト
915	S-5	iso88595	381	-	HP-UX
855	S-5	IBM-855	381	-	OS/2
915	S-5	ISO8859-5	381	-	OS/2
1251	S-5	1251	381	-	Windows

表 81. スロバキア、テリトリー ID: SK

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
912	S-2	ISO8859-2	422	sk_SK	AIX
1208	N-1	UTF-8	422	SK_SK	AIX
870	S-2	IBM-870	422	-	ホスト
1153	S-2	IBM-1153	422	-	ホスト
912	S-2	iso88592	422	sk_SK.iso88592	HP-UX
852	S-2	IBM-852	422	-	OS/2
912	S-2	ISO8859-2	422	sk_SK.ISO8859-2	SCO
1250	S-2	1250	422	-	Windows

表 82. スロベニア、テリトリー ID: SI

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
912	S-2	ISO8859-2	386	sl_SI	AIX
1208	N-1	UTF-8	386	SL_SI	AIX
870	S-2	IBM-870	386	-	ホスト
1153	S-2	IBM-1153	386	-	ホスト
912	S-2	iso88592	386	sl_SI.iso88592	HP-UX
912	S-2	ISO-8859-2	386	sl_SI	Linux
852	S-2	IBM-852	386	-	OS/2
912	S-2	ISO8859-2	386	sl_SI.ISO8859-2	SCO
1250	S-2	1250	386	-	Windows

表 83. 南アフリカ、テリトリー ID: ZA

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	27	en_ZA	AIX
850	S-1	IBM-850	27	En_ZA	AIX
923	S-1	ISO8859-15	27	en_ZA.8859-15	AIX
1208	N-1	UTF-8	27	EN_ZA	AIX
285	S-1	IBM-285	27	-	ホスト
1146	S-1	IBM-1146	27	-	ホスト
819	S-1	iso88591	27	-	HP-UX
923	S-1	iso885915	27	-	HP-UX
1051	S-1	roman8	27	-	HP-UX
437	S-1	IBM-437	27	-	OS/2
850	S-1	IBM-850	27	-	OS/2
819	S-1	ISO8859-1	27	en_ZA.ISO8859-1	SCO
819	S-1	ISO8859-1	27	-	Solaris
923	S-1	ISO8859-15	27	-	Solaris
1252	S-1	1252	27	-	Windows

表 84. スペイン、テリトリー ID: ES

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	34	es_ES	AIX
850	S-1	IBM-850	34	Es_ES	AIX
923	S-1	ISO8859-15	34	es_ES.8859-15	AIX
1208	N-1	UTF-8	34	ES_ES	AIX
284	S-1	IBM-284	34	-	ホスト
1145	S-1	IBM-1145	34	-	ホスト
819	S-1	iso88591	34	es_ES.iso88591	HP-UX
923	S-1	iso885915	34	-	HP-UX
1051	S-1	roman8	34	es_ES.roman8	HP-UX
819	S-1	ISO-8859-1	34	es_ES	Linux
923	S-1	ISO-8859-15	34	es_ES@euro	Linux
437	S-1	IBM-437	34	-	OS/2
850	S-1	IBM-850	34	-	OS/2
819	S-1	ISO8859-1	34	es	SCO
819	S-1	ISO8859-1	34	es_ES	SCO
819	S-1	ISO8859-1	34	es	Solaris
923	S-1	ISO8859-15	34	es.ISO8859-15	Solaris
1208	N-1	UTF-8	34	es.UTF-8	Solaris
1252	S-1	1252	34	-	Windows

表 85. スペイン (カタロニア語)、テリトリー ID: ES

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	34	ca_ES	AIX
850	S-1	IBM-850	34	Ca_ES	AIX
923	S-1	ISO8859-15	34	ca_ES.8859-15	AIX
1208	N-1	UTF-8	34	CA_ES	AIX

表 86. スウェーデン、テリトリー ID: SE

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	46	sv_SE	AIX
850	S-1	IBM-850	46	Sv_SE	AIX
923	S-1	ISO8859-15	46	sv_SE.8859-15	AIX
1208	N-1	UTF-8	46	SV_SE	AIX
278	S-1	IBM-278	46	-	ホスト
1143	S-1	IBM-1143	46	-	ホスト
819	S-1	iso88591	46	sv_SE.iso88591	HP-UX
923	S-1	iso885915	46	-	HP-UX
1051	S-1	roman8	46	sv_SE.roman8	HP-UX
819	S-1	ISO-8859-1	46	sv_SE	Linux
923	S-1	ISO-8859-15	46	-	Linux
437	S-1	IBM-437	46	-	OS/2
850	S-1	IBM-850	46	-	OS/2
819	S-1	ISO8859-1	46	sv	SCO
819	S-1	ISO8859-1	46	sv_SE	SCO
819	S-1	ISO8859-1	46	sv	Solaris
923	S-1	ISO8859-15	46	sv.ISO8859-15	Solaris
1208	N-1	UTF-8	46	sv.UTF-8	Solaris
1252	S-1	1252	46	-	Windows

表 87. スイス、テリトリー ID: CH

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	41	de_CH	AIX
850	S-1	IBM-850	41	De_CH	AIX
923	S-1	ISO8859-15	41	de_CH.8859-15	AIX
1208	N-1	UTF-8	41	DE_CH	AIX
500	S-1	IBM-500	41	-	ホスト
1148	S-1	IBM-1148	41	-	ホスト
819	S-1	iso88591	41	-	HP-UX
923	S-1	iso885915	41	-	HP-UX
1051	S-1	roman8	41	-	HP-UX
819	S-1	ISO-8859-1	41	de_CH	Linux
923	S-1	ISO-8859-15	41	-	Linux
437	S-1	IBM-437	41	-	OS/2
850	S-1	IBM-850	41	-	OS/2
819	S-1	ISO8859-1	41	de_CH	SCO
819	S-1	ISO8859-1	41	fr_CH	SCO
819	S-1	ISO8859-1	41	it_CH	SCO
819	S-1	ISO8859-1	41	de_CH	Solaris
923	S-1	ISO8859-15	41	-	Solaris
1252	S-1	1252	41	-	Windows

表 88. 台湾、テリトリー ID: TW

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
950	D-2	big5	88	Zh_TW	AIX
8 (295 ページ) を参照してください。					
964	D-2	IBM-eucTW	88	zh_TW	AIX
1208	N-1	UTF-8	88	ZH_TW	AIX
937	D-2	IBM-937	88	-	ホスト
1371	D-2	IBM-1371	88	-	ホスト
950	D-2	big5	88	zh_TW.big5	HP-UX
964	D-2	eucTW	88	zh_TW.eucTW	HP-UX
950	D-2	BIG5	88	zh_TW	Linux
938	D-2	IBM-938	88	-	OS/2
948	D-2	IBM-948	88	-	OS/2
950	D-2	big5	88	-	OS/2
950	D-2	big5	88	zh_TW.BIG5	Solaris
964	D-2	cns11643	88	zh_TW	Solaris
1208	N-1	UTF-8	88	zh_TW.UTF-8	Solaris
950	D-2	big5	88	-	Windows
8 (295 ページ) を参照してください。					

表 89. タイ、テリトリー ID: TH

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
874	S-20	TIS620-1	66	th_TH	AIX
1208	N-1	UTF-8	66	TH_TH	AIX
838	S-20	IBM-838	66	-	ホスト
1160	S-20	IBM-1160	66	-	ホスト
874	S-20	tis620	66	th_TH.tis620	HP-UX
874	S-20	TIS620-1	66	-	OS/2
874	S-20	TIS620-1	66	-	Windows

表 90. トルコ、テリトリー ID: TR

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
920	S-9	ISO8859-9	90	tr_TR	AIX
1208	N-1	UTF-8	90	TR_TR	AIX
1026	S-9	IBM-1026	90	-	ホスト
1155	S-9	IBM-1155	90	-	ホスト
920	S-9	iso88599	90	tr_TR.iso88599	HP-UX
920	S-9	ISO-8859-9	90	tr_TR	Linux
857	S-9	IBM-857	90	-	OS/2
920	S-9	ISO8859-9	90	tr_TR.ISO8859-9	SCO
1254	S-9	1254	90	-	Windows

表 91. 英国、テリトリー ID: GB

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	44	en_GB	AIX
850	S-1	IBM-850	44	En_GB	AIX
923	S-1	ISO8859-15	44	en_GB.8859-15	AIX
1208	N-1	UTF-8	44	EN_GB	AIX
285	S-1	IBM-285	44	-	ホスト
1146	S-1	IBM-1146	44	-	ホスト
819	S-1	iso88591	44	en_GB.iso88591	HP-UX
923	S-1	iso885915	44	-	HP-UX
1051	S-1	roman8	44	en_GB.roman8	HP-UX
819	S-1	ISO-8859-1	44	en_GB	Linux
923	S-1	ISO-8859-15	44	-	Linux
437	S-1	IBM-437	44	-	OS/2
850	S-1	IBM-850	44	-	OS/2
819	S-1	ISO8859-1	44	en_GB	SCO
819	S-1	ISO8859-1	44	en	SCO
819	S-1	ISO8859-1	44	en_GB	Solaris
923	S-1	ISO8859-15	44	en_GB.ISO8859-15	Solaris
1252	S-1	1252	44	-	Windows

表 92. ウクライナ、テリトリー ID: UA

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1124	S-12	IBM-1124	380	Uk_UA	AIX
1208	N-1	UTF-8	380	UK_UA	AIX
1123	S-12	IBM-1123	380	-	ホスト
1158	S-12	IBM-1158	380	-	ホスト
1168	S-12	KOI8-U	380	uk_UA.koi8u	Linux
1125	S-12	IBM-1125	380	-	OS/2
1251	S-12	1251	380	-	Windows

表 93. 米国、テリトリー ID: US

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	1	en_US	AIX
850	S-1	IBM-850	1	En_US	AIX
923	S-1	ISO8859-15	1	en_US.8859-15	AIX
1208	N-1	UTF-8	1	EN_US	AIX
37	S-1	IBM-37	1	-	ホスト
1140	S-1	IBM-1140	1	-	ホスト
819	S-1	iso88591	1	en_US.iso88591	HP-UX
923	S-1	iso885915	1	-	HP-UX
1051	S-1	roman8	1	en_US.roman8	HP-UX
819	S-1	ISO-8859-1	1	en_US	Linux
923	S-1	ISO-8859-15	1	-	Linux
437	S-1	IBM-437	1	-	OS/2
850	S-1	IBM-850	1	-	OS/2

表 93. 米国、テリトリー ID: US (続き)

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
819	S-1	ISO8859-1	1	en_US	SCO
819	S-1	ISO8859-1	1	en_US	SGI
819	S-1	ISO8859-1	1	en_US	Solaris
923	S-1	ISO8859-15	1	en_US.ISO8859-15	Solaris
1208	N-1	UTF-8	1	en_US.UTF-8	Solaris
1252	S-1	1252	1	-	Windows

表 94. ベトナム、テリトリー ID: VN

コード・ページ	グループ	コード・セット	テリトリー・コード	ロケール	オペレーティング・システム
1129	S-11	IBM-1129	84	Vi_VN	AIX
1208	N-1	UTF-8	84	VI_VN	AIX
1130	S-11	IBM-1130	84	-	ホスト
1164	S-11	IBM-1164	84	-	ホスト
1129	S-11	IBM-1129	84	-	OS/2
1258	S-11	1258	84	-	Windows

注:

1. ロードまたはインポート・ユーティリティーで CCSID 1392 および 5488 (GB 18030) を使用して、 CCSID 1392 および 5488 から DB2 UDB Unicode データベースにデータを移動したり、 DB2 UDB Unicode データベースから CCSID 1392 または 5488 にエクスポートすることができます。
2. AIX 4.3 以降では、コード・ページは 943 です。 AIX 4.2 以前をご使用の場合、コード・ページは 932 です。
3. ロードまたはインポート・ユーティリティーでコード・ページ 1394 (Shift JIS X0213) を使用して、コード・ページ 1394 から DB2 UDB Unicode データベースにデータを移動したり、 DB2 UDB Unicode データベースからコード・ページ 1394 にエクスポートすることができます。
4. 以下は、アラビア語圏の国/地域 (AA) にマップされます。
   - アラビア語 (サウジアラビア)
   - アラビア語 (イラク)
   - アラビア語 (エジプト)
   - アラビア語 (リビア)
   - アラビア語 (アルジェリア)
   - アラビア語 (モロッコ)
   - アラビア語 (チュニジア)
   - アラビア語 (オマーン)
   - アラビア語 (イエメン)
   - アラビア語 (シリア)
   - アラビア語 (ヨルダン)

- アラビア語 (レバノン)
- アラビア語 (クウェート)
- アラビア語 (アラブ首長国連邦)
- アラビア語 (バーレーン)
- アラビア語 (カタール)

5. 以下は、英語 (US) にマップされます。
   - 英語 (ジャマイカ)
   - 英語 (カリブ海)
6. 以下は、ラテンアメリカ (Lat) にマップされます。
   - スペイン語 (メキシコ)
   - スペイン語 (グアテマラ)
   - スペイン語 (コスタリカ)
   - スペイン語 (パナマ)
   - スペイン語 (ドミニカ共和国)
   - スペイン語 (ベネズエラ)
   - スペイン語 (コロンビア)
   - スペイン語 (ペルー)
   - スペイン語 (アルゼンチン)
   - スペイン語 (エクアドル)
   - スペイン語 (チリ)
   - スペイン語 (ウルグアイ)
   - スペイン語 (パラグアイ)
   - スペイン語 (ボリビア)
7. ヒンディ語、グジャラート語、カンナダ語、コンカニー語、マラーティー語、パンジャブ語、サンスクリット語、タミール語、およびテルグ語といったインド語派スクリプトは、 Unicode でサポートされています。
8. コード・ページ 950 は Big5 としても知られています。 Microsoft コード・ページ 950 と IBM コード・ページ 950 は以下の点で異なります。

範囲	説明	IBM	Microsoft	違い
X’8140’ から X’8DFE’	ユーザー定義文字	ユーザー定義エリア	ユーザー定義エリア	同じ
X’8E40’ から X’A0FE’	ユーザー定義文字	ユーザー定義エリア	ユーザー定義エリア	同じ
X’A140’ から X’A3BF’	特殊記号	システム文字	システム文字	同じ
X’A3C0’ から X’A3E0’	制御記号	システム文字	空	違う
X’A3E1’ から X’A3FE’	予約済み	空	空	同じ
X’A440’ から X’C67E’	基本使用文字	システム文字	システム文字	同じ
X’C6A1’ から X’C878’	Eten 追加記号	システム文字	ユーザー定義エリア	違う
X’C879’ から X’C8CC’	Eten 追加記号	空	ユーザー定義エリア	違う

範囲	説明	IBM	Microsoft	違い
X’C8CD’ から X’C8D3’	Eten 追加記号	システム文字	ユーザー定義エリア	違う
X’C8D4’ から X’C8FD’	予約済み	システム文字	ユーザー定義エリア	違う
X’C8FE’	無効/未定義文字	システム文字	ユーザー定義エリア	違う
X’C940’ から X’F9D5’	2 次使用文字	システム文字	システム文字	同じ
X’F9D6’ から X’F9FE’	Big-5 用 Eten 拡張	ユーザー定義エリア	システム文字	違う
X’FA40’ から X’FEFE’	ユーザー定義文字	ユーザー定義エリア	ユーザー定義エリア	同じ
X’8181’ から X’8C82’	ユーザー定義文字	ユーザー定義エリア	空	違う
X’F286’ から X’F9A0’	IBM 選択文字	システム文字	空	違う
合計文字数		14 060	13 502	
合計ユーザー定義文字数		6 204	6 217	
合計定義コード・ポイント数		20 264	19 719	

関連タスク:

- 328 ページの『コード・ページ 1394 と Unicode 間の以前の変換表をインストールする』

ユーロ記号サポートの使用可能化および使用不可

DB2 Universal Database™ (DB2 UDB) では、ユーロ通貨記号のサポートが提供されています。非常に多くのコード・ページにユーロ記号が追加されています。`sqllib/conv/` にある多くの DB2 UDB 内部コード・ページの変換表、多くの外部コード・ページ変換表ファイルが拡張されて、ユーロ記号をサポートするように拡張されています。

位置 X’80’ にユーロ通貨記号を含めるように、 Microsoft ANSI コード・ページが変更されています。文字 DOTLESS I (位置 X’D5’ にある) をユーロ通貨記号に置換するように、コード・ページ 850 が変更されています。 DB2 UDB 内部コード・ページの変換ルーチンは、修正されたこれらのコード・ページ定義をデフォルトとして使用して、ユーロ記号サポートを提供します。

ただし、コード・ページ変換表の非ユーロ定義を使用する場合、インストールが完了する前に以下の手順を行ってください。

前提条件:

既存の外部コード・ページの変換表ファイルを置換する際、非ユーロ・バージョンを上書きコピーする前に、現在のファイルをバックアップすることもできます。

そのファイルはディレクトリー `sqllib/conv/` にあります。 UNIX では、`sqllib/conv/` が DB2 UDB のインストール・パスにリンクされています。

手順:

以下を行って、ユーロ記号を使用不可にします。

1. DB2 UDB インスタンスを停止する。
2. バイナリー形式の適切な変換表ファイルをダウンロードします。
   - ビッグ・エンディアン・プラットフォームの場合は、ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/BigEndian/ から。この FTP サーバーは anonymous なので、コマンド行から接続する場合、ユーザー "anonymous" としてログインし、パスワードとしてご自分の E メール・アドレスを使用してください。ログイン後に、 cd ps/products/db2/info/vr8/conv/BigEndian/ として変換表ディレクトリーに移動します。
   - リトル・エンディアン・プラットフォームの場合は、ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/LittleEndian/ から。この FTP サーバーは anonymous なので、コマンド行から接続する場合、ユーザー "anonymous" としてログインし、パスワードとしてご自分の E メール・アドレスを使用してください。ログイン後に、 cd ps/products/db2/info/vr8/conv/LittleEndian として変換表ディレクトリーに移動します。
3. ファイルを `sqllib/conv/` ディレクトリーにコピーします。
4. DB2 UDB インスタンスを再始動する。

コード・ページ 819 および 1047:

コード・ページ 819 (ISO 8859-1 Latin 1 ASCII) および 1047 (Latin 1 Open System EBCDIC) の場合、ユーロ置換コード・ページである、 923 (ISO 8859-15 Latin 9 ASCII) および 924 (Latin 9 Open System EBCDIC) のそれぞれには、ユーロ記号だけでなく新しい文字もいくつか含まれています。 DB2 UDB のデフォルトでは、これらの 2 つのコード・ページおよび変換表の古い (非ユーロ) 定義 (819 と 1047) が引き続き使用されます。新しい 923/924 コード・ページ、および関連する変換表を活動化する場合、以下の 2 つの方法があります。

- 新しいコード・ページを使用する新しいデータベースを作成します。以下に例を示します。

  ```
  DB2 CREATE DATABASE dbname USING CODESET ISO8859-15 TERRITORY US
  ```

- `sqllib/conv/` ディレクトリー内の 923 または 924 変換表ファイルを、それぞれ 819 または 1047 に名前変更するか、あるいはコピーします。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『文字変換』

関連資料:

- 305 ページの『コード・ページ 923 および 924 の変換表』
- 298 ページの『ユーロを使用可能なコード・ページ変換表ファイル』

ユーロを使用可能なコード・ページ変換表ファイル

以下の表は、ユーロ通貨記号をサポートするように拡張された変換表をリストしています。ユーロ記号サポートを使用不可にしたい場合、「変換表ファイル」欄に示された変換表ファイルをダウンロードしてください。

アラビア語:

データベース・サーバー **CCSID/CPGID**	データベース・クライアント **CCSID/CPGID**	変換表ファイル
420、16804	1046、9238	04201046.cnv、IBM00420.ucs
420、16804	1256、5352	04201256.cnv、IBM00420.ucs
420、16804	1200、1208、13488、17584	IBM00420.ucs
864、17248	1046、9238	08641046.cnv、10460864.cnv、IBM00864.ucs
864、17248	1256、5352	08641256.cnv、12560864.cnv、IBM00864.ucs
864、17248	1200、1208、13488、17584	IBM00864.ucs
1046、9238	864、17248	10460864.cnv、08641046.cnv、IBM01046.ucs
1046、9238	1089	10461089.cnv、10891046.cnv、IBM01046.ucs
1046、9238	1256、5352	10461256.cnv、12561046.cnv、IBM01046.ucs
1046、9238	1200、1208、13488、17584	IBM01046.ucs
1089	1046、9238	10891046.cnv、10461089.cnv
1256、5352	864、17248	12560864.cnv、08641256.cnv、IBM01256.ucs
1256、5352	1046、9238	12561046.cnv、10461256.cnv、IBM01256.ucs
1256、5352	1200、1208、13488、17584	IBM01256.ucs

バルト語:

データベース・サーバー **CCSID/CPGID**	データベース・クライアント **CCSID/CPGID**	変換表ファイル
921、901	1257	09211257.cnv、12570921.cnv、IBM00921.ucs
921、901	1200、1208、13488、17584	IBM00921.ucs
1112、1156	1257、5353	11121257.cnv
1257、5353	921、901	12570921.cnv、09211257.cnv、IBM01257.ucs
1257、5353	922、902	12570922.cnv、09221257.cnv、IBM01257.ucs
1257、5353	1200、1208、13488、17584	IBM01257.ucs

ベラルーシ:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
1131、849	1251、5347	11311251.cnv、12511131.cnv
1131、849	1283	11311283.cnv

キリル文字:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
855、872	866、808	08550866.cnv、08660855.cnv
855、872	1251、5347	08551251.cnv、12510855.cnv
866、808	855、872	08660855.cnv、08550866.cnv
866、808	1251、5347	08661251.cnv、12510866.cnv
1025、1154	855、872	10250855.cnv、IBM01025.ucs
1025、1154	866、808	10250866.cnv、IBM01025.ucs
1025、1154	1131、849	10251131.cnv、IBM01025.ucs
1025、1154	1251、5347	10251251.cnv、IBM01025.ucs
1025、1154	1200、1208、13488、17584	IBM01025.ucs
1251、5347	855、872	12510855.cnv、08551251.cnv、IBM01251.ucs
1251、5347	866、808	12510866.cnv、08661251.cnv、IBM01251.ucs
1251、5347	1124	12511124.cnv、11241251.cnv、IBM01251.ucs
1251、5347	1125、848	12511125.cnv、11251251.cnv、IBM01251.ucs
1251、5347	1131、849	12511131.cnv、11311251.cnv、IBM01251.ucs
1251、5347	1200、1208、13488、17584	IBM01251.ucs

エストニア語:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
922、902	1257	09221257.cnv、12570922.cnv、IBM00922.ucs
922、902	1200、1208、13488、17584	IBM00922.ucs
1122、1157	1257、5353	11221257.cnv

ギリシャ語:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
423	869、9061	04230869.cnv
813、4909	869、9061	08130869.cnv、08690813.cnv、IBM00813.ucs
813、4909	1253、5349	08131253.cnv、12530813.cnv、IBM00813.ucs
813、4909	1200、1208、13488、17584	IBM00813.ucs
869、9061	813、4909	08690813.cnv、08130869.cnv
869、9061	1253、5349	08691253.cnv、12530869.cnv
875、4971	813、4909	08750813.cnv、IBM00875.ucs
875、4971	1253、5349	08751253.cnv、IBM00875.ucs
875、4971	1200、1208、13488、17584	IBM00875.ucs
1253、5349	813、4909	12530813.cnv、08131253.cnv、IBM01253.ucs
1253、5349	869、9061	12530869.cnv、08691253.cnv、IBM01253.ucs
1253、5349	1200、1208、13488、17584	IBM01253.ucs

ヘブライ語:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
424、12712	856、9048	04240856.cnv、IBM00424.ucs
424、12712	862、867	04240862.cnv、IBM00424.ucs
424、12712	916	04240916.cnv、IBM00424.ucs
424、12712	1255、5351	04241255.cnv、IBM00424.ucs
424、12712	1200、1208、13488、17584	IBM00424.ucs
856、9048	862、867	08560862.cnv、08620856.cnv、IBM0856.ucs
856、9048	916	08560916.cnv、09160856.cnv、IBM0856.ucs
856、9048	1255、5351	08561255.cnv、12550856.cnv、IBM0856.ucs
856、9048	1200、1208、13488、17584	IBM0856.ucs
862、867	856、9048	08620856.cnv、08560862.cnv、IBM00862.ucs
862、867	916	08620916.cnv、09160862.cnv、IBM00862.ucs
862、867	1255、5351	08621255.cnv、12550862.cnv、IBM00862.ucs
862、867	1200、1208、13488、17584	IBM00862.ucs
916	856、9048	09160856.cnv、08560916.cnv

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
916	862、867	09160862.cnv、08620916.cnv
1255、5351	856、9048	12550856.cnv、08561255.cnv、IBM01255.ucs
1255、5351	862、867	12550862.cnv、08621255.cnv、IBM01255.ucs
1255、5351	1200、1208、13488、17584	IBM01255.ucs

日本語:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
290、8482	850、858	02900850.cnv
1027、5123	850、858	10270850.cnv

Latin-1:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
37、1140	437	00370437.cnv、IBM00037.ucs
37、1140	850、858	00370850.cnv、IBM00037.ucs
37、1140	860	00370860.cnv、IBM00037.ucs
37、1140	1051	00371051.cnv、IBM00037.ucs
37、1140	1252、5348	00371252.cnv、IBM00037.ucs
37、1140	1275	00371275.cnv、IBM00037.ucs
37、1140	1200、1208、13488、17584	IBM00037.ucs
273、1141	437	02730437.cnv、IBM00273.ucs
273、1141	850、858	02730850.cnv、IBM00273.ucs
273、1141	1051	02731051.cnv、IBM00273.ucs
273、1141	1252、5348	02731252.cnv、IBM00273.ucs
273、1141	1275	02731275.cnv、IBM00273.ucs
273、1141	1200、1208、13488、17584	IBM00273.ucs
277、1142	437	02770437.cnv、IBM00277.ucs
277、1142	850、858	02770850.cnv、IBM00277.ucs
277、1142	1051	02771051.cnv、IBM00277.ucs
277、1142	1252、5348	02771252.cnv、IBM00277.ucs
277、1142	1275	02771275.cnv、IBM00277.ucs
277、1142	1200、1208、13488、17584	IBM00277.ucs
278、1143	437	02780437.cnv、IBM00278.ucs
278、1143	850、858	02780850.cnv、IBM00278.ucs
278、1143	1051	02781051.cnv、IBM00278.ucs
278、1143	1252、5348	02781252.cnv、IBM00278.ucs

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
278、1143	1275	02781275.cnv、IBM00278.ucs
278、1143	1200、1208、13488、17584	IBM00278.ucs
280、1144	437	02800437.cnv、IBM00280.ucs
280、1144	850、858	02800850.cnv、IBM00280.ucs
280、1144	1051	02801051.cnv、IBM00280.ucs
280、1144	1252、5348	02801252.cnv、IBM00280.ucs
280、1144	1275	02801275.cnv、IBM00280.ucs
280、1144	1200、1208、13488、17584	IBM00280.ucs
284、1145	437	02840437.cnv、IBM00284.ucs
284、1145	850、858	02840850.cnv、IBM00284.ucs
284、1145	1051	02841051.cnv、IBM00284.ucs
284、1145	1252、5348	02841252.cnv、IBM00284.ucs
284、1145	1275	02841275.cnv、IBM00284.ucs
284、1145	1200、1208、13488、17584	IBM00284.ucs
285、1146	437	02850437.cnv、IBM00285.ucs
285、1146	850、858	02850850.cnv、IBM00285.ucs
285、1146	1051	02851051.cnv、IBM00285.ucs
285、1146	1252、5348	02851252.cnv、IBM00285.ucs
285、1146	1275	02851275.cnv、IBM00285.ucs
285、1146	1200、1208、13488、17584	IBM00285.ucs
297、1147	437	02970437.cnv、IBM00297.ucs
297、1147	850、858	02970850.cnv、IBM00297.ucs
297、1147	1051	02971051.cnv、IBM00297.ucs
297、1147	1252、5348	02971252.cnv、IBM00297.ucs
297、1147	1275	02971275.cnv、IBM00297.ucs
297、1147	1200、1208、13488、17584	IBM00297.ucs
437	850、858	04370850.cnv、08500437.cnv
500、1148	437	05000437.cnv、IBM00500.ucs
500、1148	850、858	05000850.cnv、IBM00500.ucs
500、1148	857、9049	05000857.cnv、IBM00500.ucs
500、1148	920	05000920.cnv、IBM00500.ucs
500、1148	1051	05001051.cnv、IBM00500.ucs
500、1148	1114、5210	05001114.cnv、IBM00500.ucs
500、1148	1252、5348	05001252.cnv、IBM00500.ucs
500、1148	1254、5350	05001254.cnv、IBM00500.ucs
500、1148	1275	05001275.cnv、IBM00500.ucs
500、1148	1200、1208、13488、17584	IBM00500.ucs
850、858	437	08500437.cnv、04370850.cnv
850、858	860	08500860.cnv、08600850.cnv
850、858	1114、5210	08501114.cnv、11140850.cnv

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
850、858	1275	08501275.cnv、12750850.cnv
860	850、858	08600850.cnv、08500860.cnv
871、1149	437	08710437.cnv、IBM00871.ucs
871、1149	850、858	08710850.cnv、IBM00871.ucs
871、1149	1051	08711051.cnv、IBM00871.ucs
871、1149	1252、5348	08711252.cnv、IBM00871.ucs
871、1149	1275	08711275.cnv、IBM00871.ucs
871、1149	1200、1208、13488、17584	IBM00871.ucs
1275	850、858	12750850.cnv、08501275.cnv

Latin-2:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
852、9044	1250、5346	08521250.cnv、12500852.cnv
870、1153	852、9044	08700852.cnv、IBM00870.ucs
870、1153	1250、5346	08701250.cnv、IBM00870.ucs
870、1153	1200、1208、13488、17584	IBM00870.ucs
1250、5346	852、9044	12500852.cnv、08521250.cnv、IBM01250.ucs
1250、5346	1200、1208、13488、17584	IBM01250.ucs

中国語 (簡体字):

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
837、935、1388	1200、1208、13488、17584	1388ucs2.cnv
1386	1200、1208、13488、17584	1386ucs2.cnv、ucs21386.cnv

中国語 (繁体字):

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
937、835、1371	950、1370	09370950.cnv、0937ucs2.cnv
937、835、1371	1200、1208、13488、17584	0937ucs2.cnv
1114、5210	850、858	11140850.cnv、08501114.cnv

タイ:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
838、1160	874、1161	08380874.cnv、IBM00838.ucs

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
838、1160	1200、1208、13488、17584	IBM00838.ucs
874、1161	1200、1208、13488、17584	IBM00874.ucs

トルコ語:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
857、9049	1254、5350	08571254.cnv、12540857.cnv
1026、1155	857、9049	10260857.cnv、IBM01026.ucs
1026、1155	1254、5350	10261254.cnv、IBM01026.ucs
1026、1155	1200、1208、13488、17584	IBM01026.ucs
1254、5350	857、9049	12540857.cnv、08571254.cnv、IBM01254.ucs
1254、5350	1200、1208、13488、17584	IBM01254.ucs

ウクライナ:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
1123、1158	1124	11231124.cnv
1123、1158	1125、848	11231125.cnv
1123、1158	1251、5347	11231251.cnv
1124	1251、5347	11241251.cnv、12511124.cnv
1125、848	1251、5347	11251251.cnv、12511125.cnv

Unicode:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
1200、1208、13488、17584	813、4909	IBM00813.ucs
1200、1208、13488、17584	862、867	IBM00862.ucs
1200、1208、13488、17584	864、17248	IBM00864.ucs
1200、1208、13488、17584	874、1161	IBM00874.ucs
1200、1208、13488、17584	921、901	IBM00921.ucs
1200、1208、13488、17584	922、902	IBM00922.ucs
1200、1208、13488、17584	1046、9238	IBM01046.ucs

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
1200、1208、13488、17584	1250、5346	IBM01250.ucs
1200、1208、13488、17584	1251、5347	IBM01251.ucs
1200、1208、13488、17584	1253、5349	IBM01253.ucs
1200、1208、13488、17584	1254、5350	IBM01254.ucs
1200、1208、13488、17584	1255、5351	IBM01255.ucs
1200、1208、13488、17584	1256、5352	IBM01256.ucs
1200、1208、13488、17584	1386	ucs21386.cnv、1386ucs2.cnv

ベトナム語:

データベース・サーバー CCSID/CPGID	データベース・クライアント CCSID/CPGID	変換表ファイル
1130、1164	1258、5354	11301258.cnv
1258、5354	1129、1163	12581129.cnv

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『文字変換』

関連タスク:

- 296 ページの『ユーロ記号サポートの使用可能化および使用不可』

コード・ページ 923 および 924 の変換表

以下は、コード・ページ 923 および 924 と関連のあるすべてのコード・ページ変換表ファイルのリストです。それぞれのファイルは、XXXXYYYY.cnv または ibmZZZZZ.ucs の形式で成り立っています。ここで、XXXXX はソース・コード・ページ番号であり、 YYYY はターゲット・コード・ページ番号です。ファイル ibmZZZZZ.ucs は、コード・ページ ZZZZZ と Unicode との間の変換をサポートします。

特定のコード・ページ変換表を活動化するには、 2 番目の列に示されているように、その変換表ファイルをその新規名に名前変更するか、またはコピーしてください。

たとえば、8859-1/15 (Latin 1/9) クライアントを Windows 1252 データベースに接続するときに、ユーロ記号をサポートするには、 sqllib/conv/ ディレクトリー内の以下のコード・ページ変換表ファイルを名前変更するか、またはコピーする必要があります。

- 09231252.cnv から 08191252.cnv に
- 12520923.cnv から 12520819.cnv に
- ibm00923.ucs から ibm00819.ucs に

sqlllib/conv/ ディレクトリー内の 923 および 924 変換表ファイル	**新規名**
00370923.cnv	00370819.cnv
02730923.cnv	02730819.cnv
02770923.cnv	02770819.cnv
02780923.cnv	02780819.cnv
02800923.cnv	02800819.cnv
02840923.cnv	02840819.cnv
02850923.cnv	02850819.cnv
02970923.cnv	02970819.cnv
04370923.cnv	04370819.cnv
05000923.cnv	05000819.cnv
08500923.cnv	08500819.cnv
08600923.cnv	08600819.cnv
08630923.cnv	08630819.cnv
08710923.cnv	08710819.cnv
09230437.cnv	08190437.cnv
09230500.cnv	08190500.cnv
09230850.cnv	08190850.cnv
09230860.cnv	08190860.cnv
09230863.cnv	08190863.cnv
09231043.cnv	08191043.cnv
09231051.cnv	08191051.cnv
09231114.cnv	08191114.cnv
09231252.cnv	08191252.cnv
09231275.cnv	08191275.cnv
09241252.cnv	10471252.cnv
10430923.cnv	10430819.cnv
10510923.cnv	10510819.cnv
11140923.cnv	11140819.cnv
12520923.cnv	12520819.cnv
12750923.cnv	12750819.cnv
ibm00923.ucs	ibm00819.ucs

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『文字変換』

関連タスク:

- 296 ページの『ユーロ記号サポートの使用可能化および使用不可』

データベースの言語の選択

データベースを作成する場合、データを保管する言語を決定しなければなりません。データベースを作成する際に、テリトリーおよびコード・セットを指定できます。テリトリーおよびコード・セットは、現在のオペレーティング・システムの設定と異なっていても構いません。データベースの作成時に、テリトリーおよびコード・セットを明示的に選択しない場合、現在のロケールを使用してデータベースが作成されます。コード・セットを選択する場合、すべての文字を、使用する言語でエンコードできるか確認してください。

別の選択肢は、データを Unicode データベースに保管するという方法です。この場合、特定の言語を選択する必要がありません。 Unicode のエンコードには、世界中で使用されているほとんどすべての言語の文字が含まれています。

DB2 Administration Server のロケール設定

DB2 Administration Server のインスタンスのロケールが、DB2 Universal Database™ (DB2 UDB) インスタンスのロケールと互換性があるか確認してください。そうでない場合、DB2 UDB インスタンスは DB2 Administration Server とは通信できません。

DB2 Administration Server のユーザー・プロファイルで、LANG 環境変数が設定されていないなら、 DB2 Administration Server はデフォルトのシステム・ロケールで始動します。デフォルトのシステム・ロケールが定義されていないと、DB2 Administration Server はコード・ページ 819 で始動します。 DB2 UDB インスタンスが DBCS ロケールの 1 つを使用し、DB2 Administration Server がコード・ページ 819 で始動された場合、そのインスタンスは DB2 Administration Server と通信することができません。 DB2 Administration Server のロケールと DB2 UDB インスタンスのロケールは互換性がある必要があります。

たとえば、中国語 (簡体字) Linux システムで `LANG=zh_CN` は、 DB2 Administration Server のユーザー・プロファイルで設定されている必要があります。

双方向サポートの使用可能化

双方向レイアウト変換は、新しい コード化文字セット ID (CCSID) 定義を使って DB2 Universal Databaseで実装されます。新しい双方向固有の CCSID の場合、レイアウト変換はコード・ページ変換の代わりに、あるいはコード・ページ変換に加えて実行されます。このサポートを使用するためには、DB2BIDI レジストリー変数を YES に設定しなければなりません。デフォルトの設定では、この変数は設定されていません。この変数はサーバーによってすべての変換で使用され、サーバーが開始したときにのみ設定できます。 DB2BIDI を YES に設定すると、変換の追加チェックとレイアウトが原因で、パフォーマンスにいくらかの悪影響を与える可能性があります。

制約事項:

次の制限が適用されます。

- クライアント・プラットフォームのコード・ページまたはストリング・タイプに適切でない CCSID を選択すると、予期しない結果が発生する可能性があります。非互換の CCSID を選択した場合 (たとえば、アラビア語データベースへの接続にヘブライ語 CCSID を選択した場合)、または DB2BIDI がサーバーに設定されていない場合には、接続しようとするとエラー・メッセージを受け取ります。
- Windows オペレーティング・システム上の DB2 Universal Database™ (DB2 UDB) コマンド行プロセッサーには、双方向サポートはありません。
- CCSID オーバーライドは、HOST EBCDIC プラットフォームがクライアントで、かつ DB2 UDB がサーバーの場合にはサポートされません。

あるアラビア語 CCSID から別のアラビア語 CCSID に変換する場合、 DB2 UDB はラームとアリフの合字を切り離す (展開する) ために、次のロジックを使用します。変換元のアラビア語 CCSID のテキスト成形 (Text Shaping) 属性が成形 (shaped) で、変換先のアラビア語 CCSID の Text Shaping 属性が非成形 (unshaped) である場合には、切り離しが実行されます。

ラームとアリフの合字の切り離しのロジックは、次のとおりです。

1. データ・ストリームの最後の 文字がブランク文字なら、ラームとアリフの合字の後のすべての文字がデータ・ストリームの末尾に向かってシフトされ、現在のラームとアリフの合字を切り離して (展開して) 構成要素となっているラームとアリフの 2 文字にするために必要な空位置が利用可能になります。
2. そうでない場合、データ・ストリームの最初の 文字がブランク文字なら、ラームとアリフの合字の前のすべての文字がデータ・ストリームの先頭に向かってシフトされ、現在のラームとアリフの合字を切り離して (展開して) 構成要素となっているラームとアリフの 2 文字にするために必要な空位置が利用可能になります。
3. それ以外の場合、このデータ・ストリームの先頭も末尾もブランク文字ではないので、ラームとアリフの合字を切り離すことはできません。ターゲット CCSID にラームとアリフの合字が含まれているなら、ラームとアリフの合字はそのまま保たれます。含まれていないなら、ラームとアリフの合字は、ターゲット CCSID での置換文字で置き換えられます。

反対に、テキスト成形 (Text Shaping) 属性が非成形 (unshaped) であるアラビア語 CCSID から、 Text Shaping 属性が成形 (shaped) であるアラビア語 CCSID に変換する場合、変換元にラーム文字とアリフ文字があると、それらは結合されて 1 個の合字にされ、ターゲット域のデータ・ストリームの末尾にブランク文字が挿入されます。

手順:

非 DRDA 環境で特定の双方向 CCSID を指定するには、以下を行います。

- DB2BIDI レジストリー変数を必ず YES に設定します。
- クライアントの特性と一致する CCSID を選んで、 DB2CODEPAGE をその値に設定します。
- データベースにすでに接続している場合は、TERMINATE コマンドを発行し、さらに DB2CODEPAGE の新しい設定を有効にするために接続し直す必要があります。

DRDA 環境では、 HOST EBCDIC プラットフォームがこれらの双方向 CCSID をもサポートしていれば、 DB2CODEPAGE を設定するだけで済みます。サーバー・データベースの DCS データベース・ディレクトリー項目の PARMS フィールドにおいて、 BIDI パラメーターに同じ CCSID をさらに指定することはできないことに注意してください。そのように指定すると、余分の bidi レイアウト変換が実行されることになり、アラビア語データのすべてが誤って反転されて表示されることになります。しかし、HOST プラットフォームがこれらの CCSID をサポートしていない場合、接続している HOST データベース・サーバー用の CCSID オーバーライドを指定しなければなりません。これを行うには、サーバー・データベースの DCS データベース・ディレクトリーの PARMS フィールドで BIDI パラメーターを使います。このオーバーライドは必須です。DRDA 環境では、コード・ページ変換とレイアウト変換はデータの受信側で実行されるからです。しかし、HOST サーバーがこれらの双方向 CCSID をサポートしない場合、 DB2 UDB から受信するデータ上のレイアウト変換は実行されません。 CCSID オーバーライドを使用しているなら、 DB2 UDB クライアントはアウトバウンド・データ上でレイアウト変換も実行します。

関連概念:

- 312 ページの『DB2 Connect による双方向サポート』
- 「*DB2 Connect ユーザーズ・ガイド*」の『BiDi データの処理』

関連資料:

- 309 ページの『双方向特有の CCSID』
- 「*管理ガイド: パフォーマンス*」の『汎用レジストリー変数』

双方向特有の CCSID

以下の双方向属性は、異なるプラットフォーム上で双方向データの正しい処理を行うために必要です。

- テキスト・タイプ
- 数値の成形
- 方向
- テキストの成形
- 対称スワッピング

プラットフォームごとにデフォルト値が異なるため、 DB2 Universal Database™ (DB2 UDB) データをあるプラットフォームから別のプラットフォームに移すと問題が生じる可能性があります。たとえば、 Windows オペレーティング・システムは LOGICAL UNSHAPED データを使用しますが、 z/OS および OS/390 は通常 SHAPED VISUAL データを使用します。したがって、双方向属性がサポートされていない場合、 DB2 Universal Database for z/OS and OS/390 から Windows 32 ビット・オペレーティング・システム上の DB2 UDB に送られたデータは正しく表示されない可能性があります。

DB2 UDB は、特別な双方向コード化文字セット ID (CCSID) を介して、双方向データ属性をサポートします。 310 ページの表 95 で示されているように、以下の双方向 CCSID はすでに定義されており、DB2 UDB で実装されます。 CDRA ストリ

ング・タイプは、 311 ページの表 96 で示されているように定義されます。

表 95. 双方向 CCSID

CCSID	コード・ページ	ストリング・タイプ
420	420	4
424	424	4
856	856	5
862	862	4
864	864	5
867	862	4
916	916	5
1046	1046	5
1089	1089	5
1200	1200	10
1208	1208	10
1255	1255	5
1256	1256	5
5351	1255	5
5352	1256	5
8612	420	5
8616	424	10
9048	856	5
9238	1046	5
12712	424	4
13488	13488	10
16804	420	4
17248	864	5
62208	856	4
62209	862	10
62210	916	4
62211	424	5
62213	862	5
62215	1255	4
62218	864	4
62220	856	6
62221	862	6
62222	916	6
62223	1255	6
62224	420	6
62225	864	6
62226	1046	6
62227	1089	6

表 95. 双方向 CCSID (続き)

CCSID	コード・ページ	ストリング・タイプ
62228	1256	6
62229	424	8
62230	856	8
62231	862	8
62232	916	8
62233	420	8
62234	420	9
62235	424	6
62236	856	10
62237	1255	8
62238	916	10
62239	1255	10
62240	424	11
62241	856	11
62242	862	11
62243	916	11
62244	1255	11
62245	424	10
62246	1046	8
62247	1046	9
62248	1046	4
62249	1046	12
62250	420	12

表 96. CDRA ストリング・タイプ

ストリング・タイプ	テキスト・タイプ	数値の成形	方向	テキストの成形	対称スワッピング
4	表示	パススルー	LTR	成形	オフ
5	暗黙	アラビア	LTR	非成形	オン
6	暗黙	アラビア	RTL	非成形	オン
7*	表示	パススルー	コンテキスト依存*	非成形合字	オフ
8	表示	パススルー	RTL	成形	オフ
9	表示	パススルー	RTL	成形	オン
10	暗黙	アラビア	コンテキスト依存 LTR	非成形	オン
11	暗黙	アラビア	コンテキスト依存 RTL	非成形	オン
12	暗黙	アラビア	RTL	成形	オフ

注: * 最初のアルファベット文字がローマ字なら、ストリングの方向は左から右 (left-to-right (LTR)) になります。アラビア語またはヘブライ語文字の場合は、右から左 (right-to-left (RTL)) になります。文字が非成形になっていても、LamAlef 合字は保たれており、各構成要素に分けられてはいません。

関連概念:

- 312 ページの『DB2 Connect による双方向サポート』

関連タスク:

- 307 ページの『双方向サポートの使用可能化』

DB2 Connect による双方向サポート

DB2® Connect とサーバー上のデータベースとの間でデータを交換する場合、着信データの変換を実行するのは、通常は受信側です。この同じ規則は、通常のコード・ページ変換の規則とともに、双方向レイアウト変換にも当てはまります。 DB2 Connect™ は、サーバー・データベースから受け取るデータだけでなく、サーバー・データベースへ送るデータに対して、双方向レイアウト変換を実行するためのオプションを備えています。

サーバー・データベースへの発信データに対して、 DB2 Connect で双方向レイアウト変換を実行するには、サーバー・データベースの双方向 (CCSID) をオーバーライドする必要があります。これを行うには、サーバー・データベースの DCS データベース・ディレクトリーの PARMS フィールドで BIDI パラメーターを使います。

注: DB2 Universal Database™ (DB2 UDB) ホストまたは iSeries™ データベースに送る予定のデータに対して、 DB2 Connect でレイアウト変換を実行するのであれば、 CCSID をオーバーライドする必要がなくても、 DCS データベース・ディレクトリーの PARMS フィールドに BIDI パラメーターを追加する必要があります。その場合、指定する CCSID は、デフォルトの DB2 UDB ホストまたは iSeries データベース CCSID になります。

デフォルトのサーバー・データベース双方向 CCSID をオーバーライドするときに使う双方向 CCSID とともに、 BIDI パラメーターを PARMS フィールドの 9 番目のパラメーターとして以下のように指定します。

```
",,,,,,,,BIDI=xyz"
```

xyz には CCSID のオーバーライドが入ります。

注: レジストリー変数 DB2BIDI は YES にセットして、BIDI パラメーターを有効にしなければなりません。

この機能の使用方法について、わかりやすい例をあげましょう。

CCSID 62213 (双方向ストリング・タイプ 5) を実行するヘブライ語の DB2 UDB クライアントを使っていて、 CCSID 00424 (双方向ストリング・タイプ 4) を実行する DB2 UDB ホストまたは iSeries データベースにアクセスしたいとします。しかし、DB2 UDB ホストまたは iSeries データベースに含まれているデータは、CCSID 08616 (双方向ストリング・タイプ 6) をベースにしたものであることが分かっています。

ここでは 2 つの問題があります。最初の問題は、DB2 UDB ホストまたは iSeries データベース側では、 CCSID 00424 と 08616 での双方向ストリング・タイプの違いを認識していないということです。 2 番目の問題は、DB2 UDB ホストまたは iSeries データベースは、 DB2 UDB クライアント CCSID (62213) を認識しないということです。サポートされているのは CCSID 00862 だけであり、 CCSID 62213 と同じコード・ページをベースにしているものです。

まず DB2 UDB ホストまたは iSeries データベースに送るデータが、双方向ストリング・タイプ 6 形式であることを確認してから、 DB2 UDB ホストまたは iSeries データベースから受け取るデータに対して双方向変換を実行しなければならないことを、 DB2 Connect に指示する必要があります。 DB2 UDB ホストまたは iSeries データベースに次のカタログ・コマンドを使用する必要があります。

```
db2 catalog dcs database nydb1 as telaviv parms ",,,,,,,,BIDI=08616"
```

このコマンドは、 DB2 UDB ホストまたは iSeries データベースの CCSID 00424 を 08616 でオーバーライドするよう DB2 Connect に指示します。このオーバーライド作業には、以下の処理が含まれています。

1. DB2 Connect は CCSID 00862 を使用して DB2 UDB ホストまたは iSeries データベースへ接続します。
2. DB2 Connect は、DB2 UDB ホストまたは iSeries データベースに送る 予定のデータに対して、双方向レイアウト変換を実行します。この変換は、CCSID 62213 (双方向ストリング・タイプ 5) から CCSID 62221 (双方向ストリング・タイプ 6) への変換です。
3. DB2 Connect は、DB2 UDB ホストまたは iSeries データベースから受け取るデータに対して、双方向レイアウト変換を実行します。この変換は、 CCSID 08616 (双方向ストリング・タイプ 6) から CCSID 62213 (双方向ストリング・タイプ 5) への変換です。

注: 場合によっては、双方向 CCSID を使用すると、 SQL 照会そのものが変更されてしまい、 DB2 UDB サーバーによって認識されなくなることがあります。特に、異なるストリング・タイプが使われる可能性のあるときは、 IMPLICIT CONTEXTUAL と IMPLICIT RIGHT-TO-LEFT CCSID の使用を避けてください。 CONTEXTUAL CCSID を使うと、SQL 照会に引用符付きストリングが含まれる場合に、予期しない結果を生じる可能性があります。 SQL ステートメントでは引用符付きストリングを使わないようにし、その代わりにできる限りホスト変数を使用してください。

特定の双方向 CCSID が問題を引き起こしていて、前述の方法では修正できない場合には、DB2BIDI を NO に設定してください。

関連概念:

- 「*DB2 Connect ユーザーズ・ガイド*」の『BiDi データの処理』

関連資料:

- 309 ページの『双方向特有の CCSID』

照合シーケンス

データベース・マネージャーは、*照合シーケンス* によって文字データを比較します。照合シーケンスとは、一組の文字のうちどちらを大きくまたは小さく、あるいは等しく判断するかの順序付けです。

注: FOR BIT DATA 属性および BLOB データで定義した文字ストリング・データは、バイナリー・ソート順序に従ってソートされます。

たとえば、照合シーケンスは特定の文字の小文字と大文字を同等に判断するために使用されます。

データベース・マネージャーにより、固有の照合シーケンスを持つデータベースの作成が可能となります。 Unicode データベースの場合、サポートされるさまざまな照合シーケンスについては、 『DB2 Universal Database での Unicode のインプリメンテーション』を参照してください。以降の節では、データベースで使用する特定の照合シーケンスの決定と実際の使用に役立つ情報を示します。

データベースの 1 バイト文字はそれぞれ、内部では 0 から 255 まで (16 進数表記で X'00' から X'FF' まで) のユニークな番号で表されます。この番号を、文字の *コード・ポイント* といいます。文字に数字を割り当てたセットを集合的に*コード・ページ* と呼びます。照合シーケンスは、ソート後の順序列の中で各文字を配置したい位置とコード・ポイントとの間のマッピングです。位置の数値は、照合シーケンスにおける文字の*重み* と呼ばれます。最も単純な照合シーケンスでは、コード・ポイントと重みとが同じです。この順序を *Identity* と呼びます。

たとえば、文字 B と b のコード・ポイントがそれぞれ X'42' と X'62' であるとします。照合シーケンス表に従い、いずれもソートの重みが X'42' (B) である場合、これらは同様に照合されます。 B のソートの重みが X'9E' で、 b のソートの重みが X'9D' の場合は、 b が B の前にソートされます。照合シーケンス表は各文字の重みを指定します。この表はコード・ページとは異なります。コード・ページは各文字のコード・ポイントを指定します。

次の例を考えてみてください。 ASCII 文字 A から Z までは X'41' から X'5A' までで表されます。照合シーケンスを記述する場合、これらが (無関係な文字が間に入らずに) ソートされて連続する順序になっていれば、 X'41'、 X'42'、 ... X'59'、 X'5A' と記述できます。

マルチバイト文字の 16 進数値もまた重みとして使用されます。たとえば、2 バイト文字 A と B のコード・ポイントがそれぞれ X'8260' と X'8261' であるとします。そして、X'82'、X'60'、および X'61' の照合の重みを使用して、これらの 2 つの文字をコード・ポイントに従ってソートします。

照合シーケンスにおける重みは、ユニークである必要はありません。たとえば、ある文字の大文字の文字と小文字に同じ重みを与えることができます。

照合シーケンスによりすべての 256 コード・ポイントの重みを決めると、照合シーケンスの指定が容易になります。各文字の重みは、その文字のコード・ポイントを使用して判別することができます。

すべての場合に、 DB2 Universal Database™ (DB2 UDB) はデータベース作成時に指定された照合表を使用します。コード・ポイント表に現れる順序でマルチバイト文字をソートする場合は、データベースの作成時に照合シーケンスとして IDENTITY を指定しなければなりません。

注: Unicode データベースの場合、サポートされるさまざまな照合シーケンスについては、 『DB2 Universal Database での Unicode のインプリメンテーション』を参照してください。

いったん照合シーケンスが定義されると、そのデータベースでそれ以降行われるすべての文字比較はその照合シーケンスによって実行されます。 FOR BIT DATA または BLOB データとして定義された文字データを除き、その照合シーケンスはすべての SQL 比較および ORDER BY 文節で使用され、索引や統計のセットアップにも使用されます。

以下の場合には、問題が発生する可能性があります。

- アプリケーションが、データベースから取り出したソート済みデータを、別の照合シーケンスを使用してソートされたアプリケーション・データとマージする。
- アプリケーションが、あるデータベースから取り出したソート済みデータを、別のデータベースから取り出したソート済みデータとマージするが、それらのデータベースで異なる照合シーケンスが使用されている。
- アプリケーションが、ソート済みデータに関して、関係する照合シーケンスには当てはまらない事項を想定している。たとえば、特定の照合シーケンスでは、数字の照合シーケンスが英字より小さくならないことがあります。

最後に覚えておくべき点として、文字コード・ポイントの直接比較に基づくソートの結果は、 IDENTITY 照合シーケンスを使用して配列された照会結果にのみ一致します。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『文字変換』
- 321 ページの『DB2 Universal Database での Unicode のインプリメンテーション』
- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『照合順序に基づく文字比較』

タイ語文字の照合

タイ語には、特殊な母音 (″先頭の母音″)、音階記号、および連続して並べられないほかの特殊文字が含まれています。

制約事項:

タイ語のロケールおよびコード・セットを使用してデータベースを作成するか、または Unicode データベースを作成する必要があります。

手順:

タイ語とそれに対応するコード・セットを使用してデータベースを作成する場合には、 CREATE DATABASE コマンドで COLLATE USING NLSCHAR 文節を使用します。 Unicode データベースを作成する場合、 CREATE DATABASE コマンドで COLLATE USING UCA400_LTH 文節を使用します。

関連概念:

- 314 ページの『照合シーケンス』

関連資料:

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

テリトリー・コードによる日時の形式

日時の形式に関する文字ストリングの表示は、アプリケーションのテリトリー・コードに関連する日時の値のデフォルトの形式です。このデフォルトの形式は、プログラムがプリコンパイルされるかデータベースにバインドされるときに、DATETIME 形式オプションの指定によってオーバーライドすることができます。

次に、日時の入力および出力形式について説明します。

- 入力時刻形式
  - デフォルトの入力時刻形式はありません。
  - 時刻形式はすべて、すべてのテリトリー・コードの入力として使用することができます。
- 出力時刻形式
  - デフォルトの出力時刻形式は、ローカルの時刻形式になります。
- 入力の日付形式
  - デフォルトの入力日付形式はありません。
  - 日付のローカルの形式が ISO、JIS、EUR、または USA 日付形式と矛盾している場合は、ローカル形式が日付入力として認められます。たとえば、表 97 で UK の項目を見てください。
- 出力の日付形式
  - デフォルトの出力日付形式が、表 97 に示されています。

注: 表 97 には、各種のテリトリー・コードのストリング形式のリストが示されています。

表 97. テリトリー・コードによる日時の形式

テリトリー・コード	ローカル日付形式	ローカル時刻形式	デフォルトの出力の日付形式	入力日付形式
355 アルバニア	yyyy-mm-dd	JIS	LOC	LOC、 USA、 EUR、 ISO
785 アラブ	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
001 オーストラリア (1)	mm-dd-yyyy	JIS	LOC	LOC、 USA、 EUR、 ISO

表 97. テリトリー・コードによる日時の形式 (続き)

テリトリー・コード	ローカル日付形式	ローカル時刻形式	デフォルトの出力の日付形式	入力日付形式
061 オーストラリア	dd-mm-yyyy	JIS	LOC	LOC、 USA、 EUR、 ISO
032 ベルギー	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
055 ブラジル	dd.mm.yyyy	JIS	LOC	LOC、 EUR、 ISO
359 ブルガリア	dd.mm.yyyy	JIS	EUR	LOC、 USA、 EUR、 ISO
001 カナダ	mm-dd-yyyy	JIS	USA	LOC、 USA、 EUR、 ISO
002 カナダ (フランス語圏)	dd-mm-yyyy	ISO	ISO	LOC、 USA、 EUR、 ISO
385 クロアチア	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
042 チェコ共和国	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
045 デンマーク	dd-mm-yyyy	ISO	ISO	LOC、 USA、 EUR、 ISO
358 フィンランド	dd/mm/yyyy	ISO	EUR	LOC、 EUR、 ISO
389 FYR マケドニア	dd.mm.yyyy	JIS	EUR	LOC、 USA、 EUR、 ISO
033 フランス	dd/mm/yyyy	JIS	EUR	LOC、 EUR、 ISO
049 ドイツ	dd/mm/yyyy	ISO	ISO	LOC、 EUR、 ISO
030 ギリシャ	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
036 ハンガリー	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
354 アイスランド	dd-mm-yyyy	JIS	LOC	LOC、 USA、 EUR、 ISO
091 インド	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
972 イスラエル	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
039 イタリア	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
081 日本	mm/dd/yyyy	JIS	ISO	LOC、 USA、 EUR、 ISO
082 韓国	mm/dd/yyyy	JIS	ISO	LOC、 USA、 EUR、 ISO

表 97. テリトリー・コードによる日時の形式 (続き)

テリトリー・コード	ローカル日付形式	ローカル時刻形式	デフォルトの出力の日付形式	入力日付形式
001 ラテンアメリカ (1)	mm-dd-yyyy	JIS	LOC	LOC、 USA、 EUR、 ISO
003 ラテンアメリカ	dd-mm-yyyy	JIS	LOC	LOC、 EUR、 ISO
031 オランダ	dd-mm-yyyy	JIS	LOC	LOC、 USA、 EUR、 ISO
047 ノルウェー	dd/mm/yyyy	ISO	EUR	LOC、 EUR、 ISO
048 ポーランド	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
351 ポルトガル	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
086 中華人民共和国	mm/dd/yyyy	JIS	ISO	LOC、 USA、 EUR、 ISO
040 ルーマニア	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
007 ロシア	dd/mm/yyyy	ISO	LOC	LOC、 EUR、 ISO
381 セルビア/モンテネグロ	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
042 スロバキア	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
386 スロベニア	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
034 スペイン	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
046 スウェーデン	dd/mm/yyyy	ISO	ISO	LOC、 EUR、 ISO
041 スイス	dd/mm/yyyy	ISO	EUR	LOC、 EUR、 ISO
088 台湾	mm-dd-yyyy	JIS	ISO	LOC、 USA、 EUR、 ISO
066 タイ (2)	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
090 トルコ	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
044 英国	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
001 米国	mm-dd-yyyy	JIS	USA	LOC、 USA、 EUR、 ISO
084 ベトナム	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO

表 97. テリトリー・コードによる日時の形式 (続き)

テリトリー・コード	ローカル日付形式	ローカル時刻形式	デフォルトの出力の日付形式	入力日付形式
注: 1. デフォルトの C ロケールを使用する国や地域には、テリトリー・コード 001 が割り当てられます。 2. 仏教暦 yyyy は、グレゴリオ暦 + 543 年にあたります (タイのみで適用)。				

関連資料:

- 「コマンド・リファレンス」の『BIND コマンド』
- 「コマンド・リファレンス」の『PRECOMPILE コマンド』

Unicode 文字のエンコード

Unicode 文字エンコード規格は、固定長の文字エンコード方式です。これには、世界で実際に使われているほとんどすべての言語の文字が含まれています。

Unicode に関する詳細は、最新版の Unicode Standard 資料、および Unicode Consortium の Web サイト (www.unicode.org) をご覧ください。

Unicode では、8 ビットと 16 ビットの 2 つのエンコード形式が使用されます。デフォルトのエンコード形式は 16 ビットです。この場合、それぞれの文字の幅は 16 ビット (2 バイト) で、通常は U+hhhh (hhhh の部分は、文字の 16 進数コード・ポイント) と表されます。世界の主要な言語で使われているほとんどの文字をエンコードするのであれば、 65 000 のコード・エレメントがあれば十分ですが、Unicode 規格の拡張メカニズムでは、さらに 100 万文字をエンコードすることが可能です。この拡張メカニズムでは、高位のサロゲート文字と低位のサロゲート文字の組み合わせを使用して、外字および補足文字をエンコードします。前者 (高位) のサロゲート文字には、U+D800 から U+DBFF の間のコード値が含まれ、後者 (低位) のサロゲート文字には U+DC00 から U+DFFF の間のコード値が含まれます。

UCS-2

国際標準化機構 (ISO) および国際電気標準会議 (IEC) 規格 10646 (ISO/IEC 10646) の定める Universal Multiple-Octet Coded Character Set (UCS) には、16 ビット版 (UCS-2) と 32 ビット版 (UCS-4) があります。 UCS-2 は、サロゲートのない Unicode 16 ビット形式と同一です。 UCS-2 では、 Unicode のバージョン 3.0 のレパートリーで定義されている、すべての (16 ビット) 文字をエンコードできます。 Unicode のバージョン 3.1 から導入されている、新しい補足文字のそれぞれをエンコードするには、 2 つの UCS-2 文字 — 高位のサロゲートのあとに低位のサロゲートが続く — が必要です。これらの補足文字は、オリジナルの 16 ビットの Basic Multilingual Plane (BMP または Plane 0) の外部で定義されています。

UTF-8

16 ビットの Unicode 文字には、バイト試行の ASCII ベース・アプリケーションとファイル・システムに関する大きな問題があります。たとえば、Unicode を認識し

ないアプリケーションでは、大文字 'A' (U+0041) の先頭の 8 つの 0 ビットを単一バイトの ASCII NULL 文字として誤って解釈してしまう場合があります。

UTF-8 (UCS 変換形式 8) は、一種の変換アルゴリズムであり、固定長の Unicode 文字を可変長の ASCII セーフ・バイト・ストリングに変換します。 UTF-8 では、ASCII 文字や制御文字はそれぞれ通常の単一バイト・コードで表されますが、他の文字は 2 バイト以上の長さになります。 UTF-8 では、非補足文字と補足文字の両方をエンコードできます。

UTF-16

ISO/IEC 10646 ではまた、一部の UCS-4 文字を 2 個の UCS-2 文字でエンコードする拡張の技法も定義されています。この拡張は UTF-16 といい、サロゲートのある Unicode 16 ビットのエンコード形式と同じです。つまり、UTF-16 文字のレパートリーには、すべての UCS-2 文字に加えて、サロゲートのペアを通してアクセス可能な 100 万の文字が含まれることになります。

16 ビットの Unicode 文字をバイトにシリアル化する際は、プロセッサーによって、最も重要なバイトが先頭の位置に置かれる場合 (ビッグ・エンディアン順) と、最も重要性の低いバイトが先頭に置かれる場合 (リトル・エンディアン順) があります。 Unicode のデフォルトのバイト・オーダーはビッグ・エンディアンです。

UTF-8 形式におけるそれぞれの UTF-16 文字のバイト数は、 UTF 形式であり、表 98 から判断できます。

表 98. UTF-8 ビットの配布

コード値 **(バイナリー数)**	**UTF-16** **(バイナリー数)**	**最初のバイト** **(バイナリー数)**	**2 番目のバイト** **(バイナリー数)**	**3 番目のバイト** **(バイナリー数)**	**4 番目のバイト** **(バイナリー数)**
00000000 0xxxxxxx	00000000 0xxxxxxx	0xxxxxxx			
00000yyy yyxxxxxx	00000yyy yyxxxxxx	110yyyyy	10xxxxxx		
zzzzyyyy yyxxxxxx	zzzzyyyy yyxxxxxx	1110zzzz	10yyyyyy	10xxxxxx	
uuuuu zzzzyyyy yyxxxxxx	110110ww wwzzzzyy 110111yy yyxxxxxx	11110uuu (uuuuu = wwww+1)	10uuzzzz	10yyyyyy	10xxxxxx

上記のいずれの場合でも、一連の u's、w's、x's、y's、および z's は、文字のビット表示です。たとえば、U+0080 は バイナリー数で 11000010 10000000 にトランスフォームされ、サロゲート文字のペア U+D800 U+DC00 は バイナリー数で 11110000 10010000 10000000 10000000 になります。

関連概念:

- 321 ページの『DB2 Universal Database での Unicode のインプリメンテーション』
- 324 ページの『データ・タイプの Unicode 処理』
- 326 ページの『Unicode リテラル』

関連タスク:

- 325 ページの『Unicode データベースの作成』

DB2 Universal Database での Unicode のインプリメンテーション

DB2® Universal Database (DB2 UDB) は UTF-8 および UCS-2 をサポートします。

Unicode データベースが作成される際、CHAR、VARCHAR、LONG VARCHAR、および CLOB データは UTF-8 形式で格納され、GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、および DBCLOB データは UCS-2 ビッグ・エンディアン形式で格納されます。

バージョン 7.2 フィックスパック 4 以前の DB2 UDB では、サロゲート・ペアの 2 つの文字が 2 つの独立した Unicode 文字として扱われていました。そのため、UTF-16/UCS-2 から UTF-8 にペアをトランスフォームすると、 2 つの 3 バイト・シーケンスになります。 DB2 UDB バージョン 7.2 フィックスパック 4 からは、UTF-16/UCS-2 と UTF-8 との間のトランスフォームでサロゲート・ペアが認識されるようになり、 UTF-16 サロゲートのペアが 1 つの UTF-8 4 バイト・シーケンスなりました。ほかの使用法では、DB2 UDB はサロゲート・ペアを、引き続き 2 つのそれぞれ別個の UCS-2 文字として扱います。補足文字を非補足文字と区別する方法を知っているのであれば、安全に補足文字を DB2 UDB Unicode データベースに保管できます。

DB2 UDB では、揚音アクセントの結合文字 (U+0301) のような (字配りを行わない) 文字を含め、それぞれの Unicode 文字が個々の文字として扱われます。したがって、DB2 UDB は、揚音付きのラテン小文字 A (U+00E1) とラテン小文字 A (U+0061) の後に揚音アクセントの結合文字 (U+0301) を続けたものとが正規に同一であるとは認識しません。

UCS-2 Unicode データベースのデフォルトの照合シーケンスは IDENTITY です。このシーケンスでは、文字の順番はコード・ポイント順になります。したがって、デフォルトでは、すべての Unicode 文字が順番に並べられ、それぞれのコード・ポイントによって比較されます。非補足 Unicode 文字の場合、 UTF-8 でエンコードされたときのバイナリー照合の順番と UCS-2 でエンコードされたときのバイナリー照合の順番は同じになります。しかし、エンコードのためにサロゲート文字のペアを必要とする補足文字が含まれている場合は、 UTF-8 での文字のエンコードは末尾に向かって照合され、同じ文字を UCS-2 でエンコードしたときは中間のどこかで照合が実行されます。そして、この 2 つのサロゲート文字は別々に分けることができます。この理由は、外字を UTF-8 でエンコードしたときに 4 バイトのバイナリー・コード 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx ができ、これが UTF-8 エンコードの U+FFFF (つまり X’EFBFBF’) よりも大きくなってしまうことにあります。しかし UCS-2 では、同じ補足文字が UCS-2 の高位のサロゲート文字と低位のサロ

ゲート文字の組み合わせとしてエンコードされ、 1101 1000 xxxx xxxx 1101 1100 xxxx xxxx というバイナリー・フォームができ、これは UCS-2 エンコードの U+FFFF よりも小さいです。

IDENTITY_16BIT 照合オプションを使用して、 Unicode データベースを作成することもできます。 IDENTITY_16BIT コレーターは、 CESU-8 (*Compatibility Encoding Scheme for UTF-16: 8-Bit (UTF-16 互換の 8 ビット・エンコード・スキーム)*) のアルゴリズムをインプリメントします。それは Unicode Technical Consortium Web サイト (www.unicode.org) で入手可能な Unicode Technical Report #26 で指定されているものです。 CESU-8 は、Unicode 補足文字を除き、バイナリーが UTF-8 と同じです。つまり、Unicode 補足文字は 16 ビットの Basic Multilingual Plane (BMP または Plane 0) の外部で定義されています。 UTF-8 エンコードでは、補足文字は 1 つの 4 バイト・シーケンスによって表されますが、 CESU-8 では同じ文字に 2 つの 3 バイト・シーケンスが必要です。 IDENTITY_16BIT 照合オプションを使用すれば、文字と GRAPHIC データの両方で同じ照合シーケンスに従うことができます。

DB2 UDB バージョン 8.2 では、 Unicode データベースのための 2 つの新しい照合順序キーワードとして、 UCA400_NO および UCA400_LTH がサポートされています。 UCA400_NO コレーターは、 Unicode 規格バージョン 4.00 に基づく UCA (Unicode 照合アルゴリズム) で、正規化が暗黙的にオンに設定されたものをインプリメントします。また UCA400_LTH コレーターも UCA バージョン 4.00 をインプリメントしますが、その場合はタイ語のすべての文字が Royal Thai Dictionary の順序でソートされます。 UCA の詳細については、 Unicode Consortium の Web サイト (www.unicode.org) で入手できる Unicode Technical Standard #10 を参照してください。

日時形式、小数点など、地域に関係するパラメーターはすべて、クライアントの現在の地域に基づいています。

Unicode データベースでは、 DB2 UDB によってサポートされているすべてのコード・ページからの接続が可能です。データベース・マネージャーは、クライアントのコード・ページと Unicode の間にある文字と GRAPHIC ストリングに対して、コード・ページ変換を自動的に実行します。

各クライアントは、それぞれの環境でサポートされている文字レパートリー、入力方式、およびフォントによって制限を受けますが、 UCS-2 データベース自体はすべての UCS-2 文字を受け入れて格納します。そのため、各クライアントでは、通常は UCS-2 文字のサブセットを処理しますが、データベース・マネージャーでは UCS-2 文字の全レパートリーを処理できます。

文字をローカル・コード・ページから Unicode に変換すると、バイト数が増大する可能性があります。バージョン 8 以前は、SQL ステートメントのセマンティクスに基づいて、文字データはクライアントのコード・ページでエンコードされているとマークされていたため、データベース・サーバーは、クライアントのコード・ページで全ステートメントを操作していました。この操作では、データが拡張する可能性がありました。バージョン 8 以降では、一度 SQL ステートメントがデータベース・サーバーに入力されると、データベース・サーバーのコード・ページでのみ操作されます。この場合には、サイズの変更はありません。

コード・ページ/CCSID 番号

IBM® では、UCS-2 コード・ページを、文字セットが増えていくコード・ページ 1200 として登録しています。つまり、あるコード・ページに新しい文字が追加されるときに、そのコード・ページ番号が変わってしまうことはありません。コード・ページ 1200 は、常に現行バージョンの Unicode を表します。

また、Unicode 2.0 および ISO/IEC 10646-1 で定義されている特別なバージョンの UCS 規格が、IBM 内部で CCSID 13488 として登録されています。この CCSID は、GRAPHIC ストリング・データを IBM eucJP (Japan) および IBM eucTW (Taiwan) データベースに格納するときに、DB2 UDB によって内部的に使用されます。 CCSID 13488 およびコード・ページ 1200 はどちらも UCS-2 を参照し、値が「2 バイト」(DBCS) のスペースの場合を除き、同じ方法で処理されます。

CP/CCSID	単一バイト (SBCS) スペース	2 バイト (DBCS) スペース
1200	N/A	U+0020
13488	N/A	U+3000

注: UCS-2 データベースでは、U+3000 には特別な意味はありません。

変換表によると、コード・ページ 1200 は CCSID 13488 のスーパーセットであるため、どちらにも同じ (スーパーセット) 表が使われます。

IBM では、UTF-8 は、文字セットが増やされた CCSID 1208 として登録されています (コード・ページ 1208 ともいう)。この規格に新しい文字が追加されたとしても、この番号 (1208) は変わりません。

この MBCS コード・ページ番号は 1208 です。これは、データベースのコード・ページ番号、そしてそのデータベース内に含まれる文字ストリング・データのコード・ページです。 UCS-2 用の 2 バイトのコード・ページ番号は 1200 です。これは、データベース内の GRAPHIC ストリング・データのコード・ページです。

タイ語と Unicode の照合アルゴリズムの違い

Thai Industrial Standard (TIS) TIS620-1 (コード・ページ 874) の、 NLSCHAR 照合オプションを指定したタイ語データベースで使用される照合アルゴリズムは、UCA400_LTH 照合オプションの Unicode データベースで使用される照合アルゴリズムとよく似ていますが、同じではありません。その違いは次のとおりです。

- TIS620-1 データをソートする場合は、 1 文字の重みは 1 つだけであり、照合ではその重みを使用することによって別の文字の重みと比較されます。 Unicode データのソートの場合、1 文字にはいくつかの重みがあり、照合ではその文字のすべての重みが使用されます。
- TIS620-1 データをソートする場合、スペース文字 X’20’、ハイフン文字 X’2D’、および完全停止文字 X’2E’ は、いずれもその重みはどのタイ文字よりも小さくなっています。しかし Unicode データをソートする場合、それらの 3 文字は句読記号であると見なされ、比較においてそれが使用されるのは、比較する 2 つのストリングのうちのその他の文字がすべて等しい場合だけです。

- TIS620-1 データベースの Paiyannoi 文字 X'CF' と Maiyamok 文字 X'E6' は、それがそれ以外のタイ文字の直後にある場合には句読記号として処理されますが、それがストリングの先頭に出現する場合は、独自の重み値を持つ通常の文字として処理されます。 Unicode データベースの場合の同じ 2 文字 (それぞれ U+0E2F および U+0E46) は、常に句読記号として処理されるため、比較においてそれが使用されるのは、比較する 2 つのストリングの中のその他の文字がすべて等しい場合だけです。

タイ文字に関する詳細については、 Unicode 規格書、バージョン 4.0 (ISBN 0-321-18578-1) の第 10.1 章『Thai』を参照してください。

関連概念:

- 319 ページの『Unicode 文字のエンコード』
- 324 ページの『データ・タイプの Unicode 処理』
- 326 ページの『Unicode リテラル』

関連タスク:

- 325 ページの『Unicode データベースの作成』

データ・タイプの Unicode 処理

DB2® Universal Database (DB2 UDB) によってサポートされているデータ・タイプはすべて、 UCS-2 データベースでもサポートされています。特に、GRAPHIC ストリング・データは UCS-2 データベースでサポートされており、 UCS-2/Unicode で格納されます。 SBCS クライアントを含むそれぞれのクライアントでは、 UCS-2 データベースへ接続するときに、 UCS-2/Unicode の GRAPHIC ストリング・データ・タイプを処理できます。

UCS-2 データベースは、MBCS データベースに似ています。このデータベースでは文字ストリング・データがバイト数で測定されます。文字ストリング・データを UTF-8 で処理する場合、それぞれの文字を 1 バイトと見なすことはできません。マルチバイト UTF-8 エンコードの場合、各 ASCII 文字は 1 バイトですが、非 ASCII 文字はそれぞれ 2 から 4 バイトになります。 CHAR フィールドを定義するときには、このことを考慮するようにします。 ASCII 文字と非 ASCII 文字の比率に応じて、サイズが *n* バイトの CHAR フィールドには、 *n*/4 文字から *n* 文字までの任意のものを指定できます。

さらに、GRAPHIC ストリング UCS-2 データ・タイプに対して文字ストリング UTF-8 エンコードを使うことも、全体のストレージ要件に影響します。文字の大多数が ASCII で、非 ASCII 文字が少ししか含まれない場合、ストレージ要件は 1 文字当たり 1 バイトに近づくため、 UTF-8 データを格納することはより良い選択かもしれません。一方、文字の大多数が非 ASCII 文字で、 3 バイトか 4 バイトの UTF-8 シーケンスで展開される場合 (たとえば表意文字) は、 UCS-2 GRAPHIC ストリング形式を選択する方が良いでしょう。それは、すべてが 3 バイトの UTF-8 シーケンスが 1 つの 16 ビット UCS-2 文字に、それぞれが 4 バイトの UTF-8 シーケンスは 2 つの 16 ビット UCS-2 文字になるためです。

MBCS 環境では、文字ストリングに対して機能する SQL 関数 (LENGTH、SUBSTR、POSSTR、MAX、MIN など) は、文字数ではなくバイト数に基づいて機

能します。この動作は UCS-2 データベースでも同じですが、これらの値は常にデータベース・コード・ページのコンテキスト内に定義されるため、 UCS-2 データベースにオフセットと長さを指定するときには、特別な注意が必要です。つまり、UCS-2 データベースの場合、それらのオフセットは UTF-8 で定義する必要があります。単一バイト・データベースに文字の中には UTF-8 で 2 バイト以上を必要とするものもあるため、単一バイト・データベースに有効な SUBSTR の位置指定は、 UCS-2 データベースには有効ではない場合があります。不適当な位置指定をすると、SQLCODE -191 (SQLSTATE 22504) が戻されます。

SQL CHAR データ・タイプは、ユーザー・プログラムにおける C 言語の `char` データ・タイプによってサポートされています。 SQL GRAPHIC データ・タイプは、ユーザー・プログラムの `sqldbchar` によってサポートされています。ただし UCS-2 データベースでは、`sqldbchar` データは常にビッグ・エンディアン (バイト数の大きい順番) 形式であるということに注意してください。アプリケーション・プログラムを UCS-2 データベースに接続すると、文字ストリング・データは DB2 UDB によってアプリケーション・コード・ページと UTF-8 との間で変換され、GRAPHIC ストリング・データはアプリケーション GRAPHIC コード・ページと UCS-2 との間で変換されます。

Unicode データベースからデータを取り出して、 SBCS、EUC、または Unicode コード・ページを使用しないアプリケーションに渡すと、定義済みの置換文字が、GRAPHIC 列に埋め込まれている各ブランクに戻されます。 DB2 UDB は、固定長の Unicode GRAPHIC 列を、 ASCII ブランク (U+0200) (純粋な DBCS コード・ページに同等のものがない文字) で埋めます。その結果として、GRAPHIC 列の埋め込みに使用される各 ASCII ブランクは、検索時に置換文字に変換されます。同様に、DATE、TIME、または TIMESTAMP ストリングでも、純粋な DBCS に同等なものがないすべての SBCS 文字も、 Unicode データベースから検索されて SBCS、EUC、または Unicode コード・ページを使用しないアプリケーションに渡されるときに、置換文字に変換されます。

注: バージョン 8 より前のバージョンでは、GRAPHIC ストリング・データは必ず UCS-2 であると想定されていました。以前の DB2 UDB の動作に依存するアプリケーションとの後方互換性を持たせるために、レジストリー変数 DB2GRAPHICUNICODESERVER が導入されています。このデフォルト値は OFF です。この変数の値を ON に変更すると、DB2 UDB は以前のバージョンの動作が使えるようになり、 GRAPHIC ストリング・データは必ず UCS-2 であると想定されます。さらに、DB2 UDB サーバーはクライアント上で実行されている DB2 UDB のバージョンを検査し、クライアントがバージョン 7 を実行している場合は、バージョン 7 の動作をシミュレートします。

関連概念:

- 319 ページの『Unicode 文字のエンコード』
- 321 ページの『DB2 Universal Database での Unicode のインプリメンテーション』

Unicode データベースの作成

手順:

デフォルトでは、データベースは、データベースを作成しているアプリケーションのコード・ページで作成されます。したがって、Unicode (UTF-8) クライアントからデータベースを作成する場合 (たとえば、 AIX の UNIVERSAL ロケール、またはそのクライアントの DB2CODEPAGE レジストリー変数を 1208 にセットした場合)、データベースは Unicode データベースとして作成されます。別の方法としては、CODESET 名として「UTF-8」を指定し、 DB2 Universal Database™ (DB2 UDB) によってサポートされている任意のテリトリー (TERRITORY) コードを使用できます。

米国のテリトリー・コードを指定して Unicode データベースを作成するには、以下のようにします。

```
DB2 CREATE DATABASE dbname USING CODESET UTF-8 TERRITORY US
```

sqlecrea API を使用して Unicode データベースを作成するには、それぞれの値を *sqledbterritoryinfo* にセットする必要があります。たとえば、SQLDBCODESET を `UTF-8` にセットし、SQLDBLOCALE を任意の有効な地域コード (たとえば、 `US`) にセットします。

関連概念:

- 321 ページの『DB2 Universal Database での Unicode のインプリメンテーション』

関連資料:

- 「*管理 API リファレンス*」の『sqlecrea - データベースの作成』
- 「*コマンド・リファレンス*」の『CREATE DATABASE コマンド』

Unicode リテラル

Unicode リテラルは、以下の 2 つの方法で指定できます。

- G'...' または N'....' の形式を使用した GRAPHIC ストリング定数として指定する。この方法で指定したリテラルはすべて、データベース・マネージャーによってアプリケーション・コード・ページから 16 ビット Unicode に変換されます。
- UX'....' または GX'....' の形式を使用した Unicode 16 進数ストリングとして指定する。 UX または GX の後の引用符の間に指定された定数は、ビッグ・エンディアン順に 4 の倍数桁の 16 進数字でなければなりません。それぞれの 4 桁は 1 つの 16 ビット Unicode コード・ポイントを表します。なお、サロゲート文字は常にペアになっているので、高位と低位のサロゲート文字を表すには 4 桁のグループが 2 つ必要であることに注意してください。

コマンド行プロセッサー (CLP) を使用する場合、 UCS-2 文字がローカル・アプリケーションのコード・ページに存在すれば、最初の方法の方が簡単です (たとえば、コード・ページ 850 を使用している端末からコード・ページ 850 の文字を入力する場合)。 2 番目の方式は、アプリケーション・コード・ページのレパートリー外の文字のために使うようにします (たとえば、コード・ページ 850 を使用している端末から日本語の文字を指定する場合)。

関連概念:

- 319 ページの『Unicode 文字のエンコード』

- 321 ページの『DB2 Universal Database での Unicode のインプリメンテーション』

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『定数』

Unicode データベースでのストリングの比較

パターン照合において、既存の MBCS データベースの動作は UCS-2 データベースの動作とは少し異なります。

DB2® Universal Database (DB2 UDB) の MBCS データベースの現在の動き方は次のようになっています。パターンマッチングの式に MBCS データが含まれていれば、そのパターンには SBCS 文字と非 SBCS 文字の両方が含まれる可能性があります。そのパターンに含まれる特殊文字は以下のように解釈されます。

- SBCS の半角下線は 1 つの SBCS 文字を表す。
- 非 SBCS の全角下線は 1 つの非 SBCS 文字を表す。
- パーセント記号 (SBCS 半角または非 SBCS 全角のいずれか) は、ゼロ個以上の SBCS 文字または非 SBCS 文字を表す。

Unicode データベースでは、「シングルバイト」文字と「非シングルバイト」文字とは全く区別されません。 UTF-8 形式では、Unicode 文字が「混合バイト」でエンコードされますが、 UTF-8 において SBCS 文字と非 SBCS 文字との間の区別はありません。すべての文字は、UTF-8 形式上でのバイト数に関係なく 1 文字の Unicode 文字です。 Unicode の GRAPHIC 列において、半角下線 (U+005F) と全角パーセント (U+0025) を含む非補足文字は、すべて 2 バイト幅です。 Unicode データベースにおいて、パターンに含まれる特殊文字は以下のように解釈されます。

- 文字ストリングにおいて、半角下線 (X’5F’) または全角下線 (X’EFBCBF’) は、1 個の Unicode 文字を表します。半角パーセント (X’25’) および全角パーセント (X’EFBC85’) は 0 個以上の Unicode 文字を表します。
- GRAPHIC ストリングにおいて、半角下線 (U+005F) および全角下線 (U+FF3F) は、1 個の Unicode 文字を表します。半角パーセント (U+0025) または全角パーセント (U+FF05) は、 0 個以上の Unicode 文字を表します。

注: GRAPHIC 列において 1 個の Unicode 補足文字に一致させるには 2 個の下線が必要です。なぜならこのような文字は GRAPHIC 列では 2 個の UCS-2 の文字として表されるためです。一方で CHAR 列の場合、1 個の Unicode 補足文字と一致させるために必要なのは 1 個の下線だけです。

オプションの ″エスケープ式″ を使用して、下線およびパーセント記号の特別な意味を変更する際に使われる文字を指定する場合、式の指定に使用できるのは以下のいずれかです。

- 定数値
- 特殊レジスター
- ホスト変数
- 上記のいずれかのオペランドを持つスカラー関数
- 上記を連結する式

以下の制約があります。

- 式のエレメントは LONG VARCHAR、CLOB、 LONG VARGRAPHIC、または DBCLOB のタイプにはできません。さらに、BLOB ファイル参照変数にはできません。
- CHAR 列の場合、式の結果は 1 文字か、ちょうど 1 バイトを内容とするバイナリー・ストリングでなければなりません (SQLSTATE 22019)。 GRAPHIC 列の場合、式の結果は 1 文字でなければなりません (SQLSTATE 22019)。

関連概念:

- 319 ページの『Unicode 文字のエンコード』
- 321 ページの『DB2 Universal Database での Unicode のインプリメンテーション』

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『文字ストリング』
- 「*SQL リファレンス 第 1 巻*」の『GRAPHIC ストリング』

コード・ページ 1394 と Unicode 間の以前の変換表をインストールする

コード・ページ 1394 (シフト JIS X0213 とも呼ばれる) と Unicode 間の変換表は、拡張されています。日本語シフト JIS X0213 (1394) と Unicode 間の変換は、JIS X0213 文字の最終 ISO/IEC 10646-1:2000 Amendment-1 に準拠します。以前のバージョンの変換表は、`ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/` から FTP で入手可能です。

手順:

シフト JIS X0213 と Unicode 間の変換用の以前の定義をインストールするには、以下のようにします。

1. DB2 Universal Database™ (DB2 UDB) インスタンスを停止する。
2. Web ブラウザーに `ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/` を指定するか、または FTP を使用して `ftp.software.ibm.com` サイトに接続する。この FTP サーバーは anonymous です。
3. コマンド行から接続している場合は、`anonymous` をユーザー ID、 E メール・アドレスをパスワードとしてログインする。
4. ログイン後に、以下の変換表ディレクトリーに移動する。

   ```
   cd ps/products/db2/info/vr8/conv
   ```

5. 2 つのファイル 1394ucs4.cnv および ucs41394.cnv をバイナリー形式で、`sqllib/conv/` ディレクトリーにコピーする。
6. DB2 UDB インスタンスを再始動する。

関連概念:

- 321 ページの『DB2 Universal Database での Unicode のインプリメンテーション』

関連資料:

• 273 ページの『サポートされているテリトリー・コードおよびコード・ページ』

コード化文字セット ID (CCSID) 943 用の Alternative Unicode 変換表

コード化文字セット ID (CCSID) 943 に含まれる一部の文字には、それぞれ 2 つのコード・ポイントがあります。そのうち 1 つは、一般に NEC コード・ポイントと呼ばれます。もう 1 つは、一般に IBM® コード・ポイントと呼ばれます。また他にも、NEC コード・ポイントと JIS コード・ポイントで 2 つのコード・ポイントがある文字も CCSID 943 には含まれています。たとえば、ローマ数字 1 は X'8754' (NEC コード・ポイント) および X'FA4A' (IBM コード・ポイント) と表されます。同様に、和集合演算記号は X'879C' (NEC コード・ポイント) および X'81BE' (JIS コード・ポイント) と表されます。

DB2® Universal Database (DB2 UDB) デフォルト CCSID 943 の Unicode 変換表を使用すると、ある記号が 2 つのコード・ポイントによって表される場合には、両方のコード・ポイントが同じ Unicode 文字に変換されます。一方、Unicode から CCSID 943 に変換するときには、文字は JIS または IBM コード・ポイントにのみ変換されます。たとえば、デフォルト変換表を使ってローマ数字 1 を CCSID 943 から Unicode に変換した後、元に戻した場合、以下のことが発生します。

- X'FA4A' (IBM コード・ポイント) は U+2160 に変換された後、元の X'FA4A' に変換されますが、
- X'8754' (NEC コード・ポイント) は U+2160 に変換された後、 X'FA4A' に変換されます。

一方で、Microsoft® の CCSID 943 用 Unicode 変換表を使って同様の変換を行った場合は、 X'FA4A' と X'8754' の両方が X'8754' に変換されてしまいます。

さらに、CCSID 943 の中には、使用する変換表によって、異なる Unicode 文字に変換される文字もあります。たとえば、CCSID 943 文字 X'815C' は、 DB2 UDB デフォルト変換表を使用すれば Unicode 文字 U+2014 に変換されますが、Microsoft 変換表を使用すれば U+2015 に変換されます。 DB2 UDB で Microsoft の変換表を使用したい場合には、 DB2 UDB 変換表を Microsoft 変換表に置き換える必要があります。

このような Microsoft 変換表を使用するのは、 CCSID 943 の DB2 UDB データベースと CCSID 943 の DB2 UDB クライアントとの間の閉鎖環境で、データベースとクライアントの両方が Microsoft の変換表を使用する場合に限られます。 DB2 UDB デフォルトの変換表を使用する DB2 UDB クライアントと、 Microsoft の変換表を使用する DB2 UDB クライアントが混在し、両方のクライアントが CCSID 943 の同じ DB2 UDB データベースに接続する場合、同じ文字が 2 つの異なるコード・ポイントとしてデータベースに保管される可能性があります。

関連概念:

• 319 ページの『Unicode 文字のエンコード』

関連タスク:

• 330 ページの『コード化文字セット (CCSID) 943 の Unicode 変換表を Microsoft 変換表に置換する』

コード化文字セット (CCSID) 943 の Unicode 変換表を Microsoft 変換表に置換する

デフォルトでは、CCSID 943 と Unicode との間で変換するとき、 DB2 Universal Database™ (DB2 UDB) のデフォルト・コード・ページ変換表が使用されます。Microsoft バージョンの変換表など、異なるバージョンの変換表を使用する場合、デフォルトの変換表 (.cnv) ファイルを手動で置換する必要があります。

前提条件:

`sqllib/conv` ディレクトリー内の既存のコード・ページ変換表ファイルを置換する前に、それらを変更前の状態に戻す場合に備えて、ファイルのバックアップをとる必要があります。 UNIX では、`sqllib/conv` が DB2 UDB のインストール・パスにリンクされています。

制約事項:

これを有効にするには、同じデータベースに接続するすべての DB2 UDB クライアントが、変更済みの変換表を持っている必要があります。そうでない場合には、異なるクライアントが異なるコード・ポイントを使用して同じ文字を保管する可能性があります。

手順:

CCSID 943 と Unicode との間で変換するための DB2 UDB のデフォルトの変換表を置換するには、以下のようにします。

1. `sqllib/conv/ms/0943ucs2.cnv` を `sqllib/conv/0943ucs2.cnv` にコピーします。
2. `sqllib/conv/ms/ucs20943.cnv` を `sqllib/conv/ucs20943.cnv` にコピーします。
3. DB2 UDB を再始動します。

関連概念:

- 329 ページの『コード化文字セット ID (CCSID) 943 用の Alternative Unicode 変換表』

付録 C. 64 ビット環境におけるラージ・ページのサポートの使用可能化 (AIX)

IBM eServer pSeries システムの POWER4 プロセッサーでは、従来のページ・サイズ 4 KB に加えて、新しく 16 MB のページ・サイズがサポートされるようになりました。 5100-02 推奨保守パッケージが適用されている AIX 5L for POWER バージョン 5.1 や、 AIX バージョン 5.2 には、16 MB のページのサポートが装備されています。この環境下で稼働していれば、 DB2 Universal Database™ (DB2 UDB) for AIX 64 ビット・エディションでも、これらのラージ・ページを使用可能にできます。

ラージ・ページの使用は主に、高性能コンピューティング・アプリケーションのパフォーマンスの向上を意図したものです。集中的なメモリー・アクセスを必要とし、大量の仮想メモリーを使用するアプリケーションでは、このラージ・ページの使用によってパフォーマンスを向上できるでしょう。

注:

1. **vmtune** コマンドや **vmo** コマンドの実行方法に関する詳しい説明は、 AIX のマニュアルを参照してください。
2. メモリーを固定してラージ・ページをサポートするようにシステムを構成する場合には、十分な注意が必要です。あまり多くのメモリーを固定しすぎると、固定されていないメモリー・ページにかかるページング・アクティビティーの負荷が大きくなります。ラージ・ページに物理メモリーを割り振りすぎると、4 KB ページのサポートにメモリーが不足して、かえってシステムのパフォーマンスを低下させることになります。
3. DB2_LGPAGE_BP レジストリー変数の設定によっても、メモリーは固定されます。

前提条件:

AIX 5.x 以降の 64 ビット環境を使用している必要があります。 AIX オペレーティング・システム・コマンドを実行できる root 権限が必要です。

手順:

ラージ・ページのサポートを使用可能にするには、次のようにする必要があります。

1. ラージ・ページがサポートされるように AIX サーバーを構成します。

   AIX 5.1 オペレーティング・システムの場合は、次のフラグを立てて **vmtune** コマンドを発行します。

   ```
   vmtune -g <LargePageSize> -L <LargePages>
   ```

   AIX 5.2 オペレーティング・システムの場合は、次のフラグを立てて **vmo** コマンドを発行します。

   ```
   vmo -r -o lgpg_size=<LargePageSize> lgpg_regions=<LargePages>
   ```

   ここで、

   `<LargePageSize>`

ハードウェアでサポートされているラージ・ページのサイズをバイト単位で指定します。

```
<LargePages>
```

予約するラージ・ページの数を指定します。

たとえば、ラージ・ページのサポート用に 25 GB を割り振る必要がある場合は、次のコマンドを実行します。

AIX 5.1 オペレーティング・システムの場合:

```
vmtune -g 16777216 -L 1600
```

AIX 5.2 オペレーティング・システムの場合:

```
vmo -r -o lgpg_size=16777216 lgpg_regions=1600
```

2. **bosboot** コマンドを実行します。これによって次のシステム・ブート時に、先に実行した **vmtune** または **vmo** コマンドが有効になります。
3. サーバーが立ち上がったなら、これを固定したメモリーに使用できるようにします。

   AIX 5.1 オペレーティング・システムの場合は、次のフラグを立てて **vmtune** コマンドを発行します。

   ```
   vmtune -S 1
   ```

   AIX 5.2 オペレーティング・システムの場合は、次のフラグを立てて **vmo** コマンドを発行します。

   ```
   vmo -o v_pinshm=1
   ```

4. **db2set** コマンドを使用して DB2_LGPAGE_BP レジストリー変数を 『YES』に設定し、次いで DB2 UDB を開始します。

   ```
   db2set DB2_LGPAGE_BP=YES
   db2start
   ```

関連概念:

- 101 ページの『表スペースの設計』
- 104 ページの『システム管理スペース』
- 106 ページの『データベース管理スペース』

付録 D. DB2 Universal Database の技術情報の概要

DB2 ドキュメンテーションおよびヘルプ

DB2 の技術情報は、以下のツールや手段によって利用できます。

- DB2 インフォメーション・センター
  - トピック
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- ダウンロード可能な PDF ファイルおよび印刷された資料
  - ガイド
  - リファレンス・マニュアル
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ
  - SQL 状態ヘルプ
- インストールされているソース・コード
  - サンプル・プログラム

ibm.com において、オンラインでも付加的な DB2 Universal Database の技術情報を利用できます。その中には、技術情報、技術白書、およびレッドブック (Redbooks) が含まれています。 DB2 Information Management Library サイト (www.ibm.com/software/data/db2/udb/support.html) をご覧ください。

DB2 ドキュメンテーションの更新

IBM では、利用可能な DB2 インフォメーション・センターに対するドキュメンテーションのフィックスパックや、その他のドキュメンテーション更新情報を定期的に作成する場合があります。 http://publib.boulder.ibm.com/infocenter/db2help/ から DB2 インフォメーション・センターを利用する場合は、常にほとんど最新の情報を参照できます。 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た場合、更新された情報を表示するためには、更新情報をインストールする必要があります。ドキュメンテーション更新情報をインストールすると、新しい情報が利用可能になった場合に、 *DB2* インフォメーション・センター *CD* からインストールした情報が更新されます。

インフォメーション・センターは、 PDF やハードコピー資料に比べて更新が頻繁です。 DB2 の最新の技術情報を入手するには、ドキュメンテーション更新情報が利用可能になった時点でそれをインストールするか、または www.ibm.com サイトの DB2 インフォメーション・センターを利用してください。

関連概念:

- 「*コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻*」の『CLI サンプル・プログラム』
- 「*アプリケーション開発ガイド アプリケーションの構築および実行*」の『Java サンプル・プログラム』
- 334 ページの『DB2 インフォメーション・センター』

関連タスク:

- 355 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 345 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 356 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 356 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 357 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 347 ページの『DB2 PDF 資料および印刷された資料』

DB2 インフォメーション・センター

DB2® インフォメーション・センターを使用すると、 DB2 Universal Database™、 DB2 Connect™、 DB2 Information Integrator および DB2 Query Patroller™ などの DB2 ファミリー製品を最大限に活用するのに必要なすべての情報にアクセスできます。また、DB2 インフォメーション・センターは、 DB2 の主な機能とコンポーネントに関する情報を提供します (レプリケーション、データウェアハウジング、および DB2 の種々の Extender など)。

Mozilla 1.0 以上または Microsoft® Internet Explorer 5.5 以上で表示する場合、 DB2 インフォメーション・センターには以下の機能があります。以下のいくつかの機能では、 JavaScript™ のサポートを使用可能にする必要があります:

柔軟なインストール・オプション

以下の中から、 ご使用の環境に最も適したオプションを使って DB2 資料を表示できます。

- 最新の資料を常に自動的に利用できるようにするには、 IBM® の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターからすべての資料に直接アクセスします。
- 更新処理を最小化し、イントラネット内のネットワーク・トラフィックだけに制限するには、イントラネット上の 1 つのサーバーに DB2 資料をインストールします。
- 柔軟性を改善し、ネットワーク接続への依存を軽減するには、個々のコンピューターに DB2 資料をインストールします。

検索

「**検索**」テキスト・フィールドに検索語を入力することにより、 DB2 インフォメーション・センターのすべてのトピックを検索できます。複数の語句を引用符で囲めば、完全一致を検索できます。また、ワイルドカード演算子 (*、?) とブール演算子 (AND、 NOT、OR) を使用して検索を絞り込むことができます。

タスク指向の目次

単一の目次の中から、DB2 資料のトピックを見付けることができます。目次は、主に実行するタスクの種類に従って編成されていますが、そのほかに製品概要、特定のゴール (目的) の情報、参照情報、索引、および用語集も含まれます。

- 製品概要では、DB2 ファミリーで使用可能な製品間の関係、そうした各製品で提供される機能、および各製品の最新リリース情報について説明されています。
- インストール、管理および開発などのゴール・カテゴリーには、タスクを迅速に完了し、そのための背景情報をよく理解できるようにするトピックが含まれています。
- 「参照」トピックでは、その対象に関する詳細な情報 (ステートメントとコマンドの構文、メッセージ・ヘルプ、構成パラメーターなど) が説明されています。

現在のトピックを目次に表示する

現在のトピックが目次のどの部分に該当するかを表示するには、目次フレーム内の**「リフレッシュ/現在のトピックの表示 (Refresh/Show Current Topic)」**ボタンをクリックするか、コンテンツ・フレーム内の**「目次に表示 (Show in Table of Contents)」**ボタンをクリックします。幾つかのファイルで関連トピックへの複数のリンクをたどった場合、または検索結果からトピックにアクセスした場合には、この機能が役立ちます。

索引 索引から、すべての資料にアクセスすることができます。索引では、用語が 50 音順に編成されています。

用語集 用語集を見れば、DB2 資料で使われているさまざまな用語の定義を調べることができます。用語集では、用語が 50 音順に編成されています。

組み込まれているローカライズ情報

DB2 インフォメーション・センターは、ブラウザーで設定された言語でトピックを表示します。設定された言語のトピックが利用できない場合、DB2 インフォメーション・センターにはそのトピックの英語版が表示されます。

iSeries™ 技術情報については、 IBM eServer™ iSeries Information Center (www.ibm.com/eserver/iseries/infocenter/) を参照してください。

関連概念:

- 336 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 345 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 346 ページの『DB2 インフォメーション・センターのトピックを特定の言語で表示する方法』
- 344 ページの『DB2 インフォメーション・センターの呼び出し』
- 339 ページの『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 (UNIX)』

• 342 ページの『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 (Windows)』

DB2 インフォメーション・センターのインストール・シナリオ

さまざまな作業環境にある人々が、それぞれの環境に応じた方法で DB2 製品資料にアクセスする必要があります。それで、DB2 製品資料にアクセスする方法には、IBM Web サイトからアクセスする方法、イントラネット・サーバーからアクセスする方法、そしてコンピューター上にインストールしてアクセスする方法の 3 種類があります。それら 3 種類のいずれにおいても、資料は DB2 インフォメーション・センターに含まれています。それは、ブラウザーで表示できるトピック単位の情報として設計された Web です。 DB2 製品は、デフォルトで IBM Web サイトから DB2 インフォメーション・センターにアクセスするようになっています。しかし、DB2 インフォメーション・センターをイントラネット・サーバーからアクセスしたり、自分のコンピューターでアクセスしたりしたい場合には、製品メディア・パックに含まれている DB2 インフォメーション・センター CD を使用することによって、 DB2 インフォメーション・センターをインストールする必要があります。以下に示す 3 つのシナリオを参考にすることにより、自分にとって、または自分の作業環境においてどの方法で DB2 インフォメーション・センターにアクセスするのが最善かを決定し、インストールに関して考慮しなければならない問題は何かを判別してください。

シナリオ: IBM Web サイトから DB2 インフォメーション・センターにアクセスする場合:

コリン氏は、ある教育機関の IT 関係の顧問をしています。彼は、データベース・テクノロジーや SQL を専門としており、北米全体の企業を対象として、 DB2 Universal Database Express Edition を使用したセミナーを開催しています。コリン氏のセミナーのある部分では、教材として DB2 のドキュメンテーションを使用します。たとえば、SQL の講座においてコリン氏は、データベース照会の基本的な構文や上級者向けの構文を教える手段として、 SQL に関する DB2 ドキュメンテーションを使用します。

コリン氏がセミナーを開く企業のほとんどにおいて、インターネットへのアクセスが可能です。それでコリン氏は、自分のモバイル・コンピューターに DB2 Universal Database Express Edition の最新版をインストールする際に、 IBM Web サイトにある DB2 インフォメーション・センターにアクセスするように設定することにします。それによりコリン氏は、セミナーの中で、最新の DB2 ドキュメンテーションにオンラインでアクセスできます。

しかし、コリン氏が旅行中はインターネットにアクセスできません。これは少し問題です。特に、セミナーの準備のために DB2 ドキュメンテーションにアクセスすることが必要になった場合に困ります。そのような状況に対処するため、コリン氏は、自分のモバイル・コンピューターにも DB2 インフォメーション・センターのコピーをインストールすることにします。

DB2 ドキュメンテーションのコピーがいつでも手元にあるので、コリン氏は柔軟に対応することができるようになりました。 **db2set** コマンドを使用して自分のモバイル・コンピューターのレジストリー変数の設定を容易に変更することにより、状

況に応じて、IBM Web サイトの DB2 インフォメーション・センターにアクセスしたり、自分のモバイル・コンピューター上にあるものを利用したりできます。

シナリオ: イントラネット・サーバー上の DB2 インフォメーション・センターにアクセスする場合:

エバ氏は、ある生命保険会社の主任データベース管理者です。管理者としての彼女の責任には、 DB2 Universal Database Enterprise Server Edition の最新バージョンを会社の UNIX データベース・サーバーにインストールおよび構成することが含まれます。彼女の会社では、最近従業員に対して、セキュリティー上の理由により業務中はインターネットへのアクセスができなくなることが通知されました。彼女の会社はネットワーク環境にあるため、エバ氏は DB2 インフォメーション・センターのコピーをイントラネット・サーバーにインストールすることにします。そのようにすれば、その会社のデータウェアハウスを定常的に使用する従業員の全員 (営業員、営業部長、および経営分析担当者) が DB2 インフォメーション・センターにアクセスできます。

DB2 インフォメーション・センターをイントラネット・サーバーにインストールする際にエバ氏は、 DB2 セットアップ・ウィザードから、 DB2 インフォメーション・センターがネットワーク上の他のコンピューターからの通信を受信するためのポートを指定するよう求められます。そこで彼女は、 DB2 インフォメーション・センターをインストールするイントラネット・サーバーのサービス名とポート番号を指定します。

次にエバ氏は、自分のデータベース・チームに対して、応答ファイルを使用して全従業員のコンピューターに DB2 Universal Database の最新バージョンをインストールするよう指示します。それにより各コンピューターは、イントラネット・サーバーのホスト名とポート番号を使用して DB2 インフォメーション・センターにアクセスするよう構成されます。

ところが、エバ氏のチームのデータベース管理者の一人であるミゲルが指示を聞き間違えて、イントラネット・サーバー上の DB2 インフォメーション・センターにアクセスするよう DB2 Universal Database を構成するのでなく、何人かの従業員のコンピューターに DB2 インフォメーション・センターのコピーをインストールしてしまいました。この状況を正すためエバ氏は、ミゲルに対して、 **db2set** コマンドを使用することにより、それらの各コンピューターの DB2 インフォメーション・センター・レジストリー変数 (ホスト名の変数は DB2_DOCHOST、ポート番号の変数は DB2_DOCPORT) を変更するように指示します。これで、ネットワーク上の該当するすべてのコンピューターは、 DB2 インフォメーション・センターにアクセスできるようになり、従業員は DB2 に関する質問の回答を DB2 ドキュメンテーションから調べることができるようになりました。

シナリオ: 自分のコンピューター上の DB2 インフォメーション・センターにアクセスする場合:

ツーチェン氏は、小さな町で工場を経営しています。そこにはインターネット・アクセスを提供する地元の ISP がありません。彼は、在庫管理、製品注文情報、銀行口座の情報、および経費を管理するために、DB2 Universal Database Personal

Edition を購入しました。それまで一度も DB2 製品を使用したことがなかったので、ツーチェン氏は、その使い方を調べるため DB2 製品資料を必要としています。

標準のインストール・オプションを使用して DB2 Universal Database Personal Edition を自分のコンピューターにインストールした後、ツーチェン氏は DB2 ドキュメンテーションを開こうとします。しかしブラウザーに、開こうとしたページが見つからないというエラー・メッセージが表示されます。ツーチェン氏は、「*DB2 Universal Database Personal Edition 概説およびインストール*」を見て、コンピューター上の DB2 ドキュメンテーションにアクセスするには DB2 インフォメーション・センターをインストールする必要があることを知ります。そこで、メディア・パックから、 *DB2 インフォメーション・センター CD* を取り出し、それをインストールします。

これでツーチェン氏は、オペレーティング・システムのアプリケーション・ランチャーから DB2 インフォメーション・センターを利用できるようになり、 DB2 製品を利用して作業効率を改善する方法について調べることができるようになりました。

DB2 ドキュメンテーションへのアクセス方法のサマリー:

各作業環境に応じて、 DB2 インフォメーション・センターに含まれる DB2 製品資料にアクセスするためにどのオプションが最善かを、次の表にまとめます。

インターネット・アクセス	イントラネット・アクセス	推奨
Yes	Yes	IBM Web サイトの DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーにインストールされている DB2 インフォメーション・センターにアクセスする。
Yes	No	IBM Web サイトの DB2 インフォメーション・センターにアクセスする。
No	Yes	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・センターにアクセスする。
No	No	ローカル・コンピューター上の DB2 インフォメーション・センターにアクセスする。

関連概念:

- 334 ページの『DB2 インフォメーション・センター』

関連タスク:

- 345 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 339 ページの『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 (UNIX)』
- 342 ページの『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 (Windows)』

関連資料:

- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』

DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール (UNIX)

DB2 製品資料にアクセスするには、IBM Web サイトからアクセスする方法、イントラネット・サーバーからアクセスする方法、そしてコンピューター上にインストールしてアクセスする方法の 3 種類の方法があります。 DB2 製品は、デフォルトで IBM Web サイトから DB2 ドキュメンテーションにアクセスするようになっています。イントラネット・サーバーまたは自分のコンピューターから DB2 ドキュメンテーションにアクセスするには、 *DB2 インフォメーション・センター CD* からドキュメンテーションをインストールする必要があります。 DB2 セットアップ・ウィザードを使用すると、インストール・システムのさまざまな設定値を定義し、 UNIX オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールすることができます。

前提条件:

ここでは、 UNIX コンピューターに DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の要件のリストを示します。

- **ハードウェア要件**

  以下のうちいずれか 1 つのプロセッサーが必要です。

  – PowerPC (AIX)
  – HP 9000 (HP-UX)
  – Intel 32 ビット (Linux)
  – Solaris UltraSPARC コンピューター (Solaris オペレーティング環境)

- **オペレーティング・システム要件**

  以下のうちいずれか 1 つのオペレーティング・システムが必要です。

  – IBM AIX 5.1 (PowerPC)
  – HP-UX 11i (HP 9000)
  – Redhat Linux 8.0 (Intel 32 ビット)
  – SuSE Linux 8.1 (Intel 32 ビット)
  – Sun Solaris バージョン 8 (Solaris オペレーティング環境 UltraSPARC コンピューター)

- **ソフトウェア要件**

  – 以下のブラウザーがサポートされています。

    - Mozilla バージョン 1.0 以上

- DB2 セットアップ・ウィザードは、グラフィック・インストーラーです。ご使用のコンピューターで DB2 セットアップ・ウィザードを実行するには、グラフィカル・ユーザー・インターフェースを表示できる X Window System ソフトウェ

アが必要です。 DB2 セットアップ・ウィザードを実行する前に、ディスプレイを正しくエクスポートしたことを確認してください。たとえば、以下のコマンドをコマンド・プロンプトに入力します。

```
export DISPLAY=9.26.163.144:0.
```

- **通信要件**
  - TCP/IP

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、

1. システムにログオンします。
2. DB2 インフォメーション・センター CD をシステムに挿入し、マウントします。
3. 次のコマンドを入力することによって、 CD がマウントされているディレクトリーに移動します。

   ```
   cd /cd
   ```

   /cd は CD のマウント・ポイントです。
4. **./db2setup** コマンドを入力して、 DB2 セットアップ・ウィザードを開始します。
5. **「IBM DB2 セットアップ・ランチパッド」** が表示されます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを表示するには、**「ヘルプ」**をクリックします。**「キャンセル (Cancel)」**を押せば、いつでもインストールを終了できます。
6. **「インストールしたい製品を選択します」**ウィンドウで、**「次へ」**をクリックします。
7. **「DB2 インフォメーション・センターの DB2 セットアップ・ウィザードへようこそ」**ウィンドウで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードにより、プログラムのセットアップ・プロセスが案内されます。
8. インストールを続行するには、ご使用条件に同意する必要があります。**「ご使用条件」**ウィンドウで、**「使用条件の条項に同意します」**を選択し、**「次へ」**をクリックします。
9. **「インストール・アクションの選択」**ウィンドウで、 DB2 インフォメーション・センターのインストール先を選択します。後で応答ファイルを使用してこのコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをインストールする場合は、**「設定を応答ファイルに保管する」**を選択します。**「次へ (Next)」**をクリックします。
10. **「インストールする言語の選択」**ウィンドウで、 DB2 インフォメーション・センターのインストール言語を選択します。**「次へ (Next)」**をクリックします。
11. **「DB2 インフォメーション・センター・ポートの指定」**で、 DB2 インフォメーション・センターが受け付ける通信について構成します。**「次へ」**をクリックして、インストールを続行します。

12. **「ファイルのコピーの開始」**ウィンドウで、それまでに選択したインストール・オプションを確認します。設定を確認または変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

DB2 インフォメーション・センターは、応答ファイルを使用してインストールすることもできます。

インストール・ログ db2setup.his、db2setup.log、および db2setup.err は、デフォルトでは /tmp ディレクトリーに入っています。ログ・ファイルの位置は指定可能です。

db2setup.log ファイルには、エラーを含めてすべての DB2 製品インストール情報が入れられます。 db2setup.his ファイルには、そのコンピューター上のすべての DB2 製品インストールが記録されます。 DB2 は db2setup.log ファイルを db2setup.his ファイルに付加します。 db2setup.err ファイルには、例外やトラップ情報など、Java から戻されたエラー出力が入れられます。

インストールが完了すると、使用している UNIX オペレーティング・システムに応じて、以下のいずれかのディレクトリーに 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ます。

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris オペレーティング環境 /opt/IBM/db2/V8.1

関連概念:

- 334 ページの『DB2 インフォメーション・センター』
- 336 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 のインストール (UNIX)』
- 345 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 346 ページの『DB2 インフォメーション・センターのトピックを特定の言語で表示する方法』
- 344 ページの『DB2 インフォメーション・センターの呼び出し』
- 342 ページの『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 (Windows)』

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 (Windows)

DB2 製品資料にアクセスするには、IBM Web サイトからアクセスする方法、イントラネット・サーバーからアクセスする方法、そしてコンピューター上にインストールしてアクセスする方法の 3 種類の方法があります。 DB2 製品は、デフォルトで IBM Web サイトから DB2 ドキュメンテーションにアクセスするようになっています。イントラネット・サーバーまたは自分のコンピューターから DB2 ドキュメンテーションにアクセスするには、 *DB2 インフォメーション・センター CD* から DB2 ドキュメンテーションをインストールする必要があります。 DB2 セットアップ・ウィザードを使用すると、インストール・システムのさまざまな設定値を定義し、 Windows オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールすることができます。

前提条件:

ここでは、 Windows に DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の要件のリストを示します。

- **ハードウェア要件**

  以下のプロセッサーが必要です。

  – 32 ビット・コンピューター: Pentium または Pentium と互換の CPU。

- **オペレーティング・システム要件**

  以下のうちいずれか 1 つのオペレーティング・システムが必要です。

  – Windows 2000

  – Windows XP

- **ソフトウェア要件**

  – 次のブラウザーがサポートされています。

    - Mozilla 1.0 以上

    - Internet Explorer バージョン 5.5 または 6.0 (Windows XP の場合はバージョン 6.0)

- **通信要件**

  – TCP/IP

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、

1. DB2 インフォメーション・センターのインストールのために定義したアカウントで、システムにログオンします。
2. CD をドライブに挿入します。自動実行機能を使用可能にしている場合には、それによって IBM DB2 セットアップ・ランチパッドが起動されます。
3. DB2 セットアップ・ウィザードによりシステム言語が判別され、その言語用のセットアップ・プログラムが起動されます。セットアップ・プログラムを英語

以外の言語で実行したい場合や、セットアップ・プログラムが自動始動に失敗する場合には、 DB2 セットアップ・ウィザードを手動で開始することができます。

DB2 セットアップ・ウィザードを手動で開始するには、

a. **「スタート」**をクリックし、**「ファイル名を指定して実行」**を選択します。
b. **「開く」**フィールドで、次のコマンドを入力します。

```
x:¥setup language
```

x: は CD のドライブ、 *language* はセットアップ・プログラムが実行されている言語です。

c. **「OK」**をクリックします。

4. **「IBM DB2 セットアップ・ランチパッド」** が表示されます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを表示するには、**「ヘルプ」**をクリックします。**「キャンセル (Cancel)」**を押せば、いつでもインストールを終了できます。
5. **「インストールしたい製品を選択します」**ウィンドウで、**「次へ」**をクリックします。
6. **「DB2 インフォメーション・センターの DB2 セットアップ・ウィザードへようこそ」**ウィンドウで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードにより、プログラムのセットアップ・プロセスが案内されます。
7. インストールを続行するには、ご使用条件に同意する必要があります。**「ご使用条件」**ウィンドウで、**「使用条件の条項に同意します」**を選択し、**「次へ」**をクリックします。
8. **「インストール・アクションの選択」**ウィンドウで、 DB2 インフォメーション・センターのインストール先を選択します。後で応答ファイルを使用してこのコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをインストールする場合は、**「設定を応答ファイルに保管する」**を選択します。**「次へ (Next)」**をクリックします。
9. **「インストールする言語の選択」**ウィンドウで、 DB2 インフォメーション・センターのインストール言語を選択します。**「次へ (Next)」**をクリックします。
10. **「DB2 インフォメーション・センター・ポートの指定」**で、 DB2 インフォメーション・センターが受け付ける通信について構成します。**「次へ」**をクリックして、インストールを続行します。
11. **「ファイルのコピーの開始」**ウィンドウで、それまでに選択したインストール・オプションを確認します。設定を確認または変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

DB2 インフォメーション・センターは、応答ファイルを使用してインストールできます。また、**db2rspgn** コマンドを使用することによって、既存のインストール・システムに基づく応答ファイルを生成することもできます。

インストール時に検出されるエラーの詳細については、 db2.log と db2wi.log のファイルを参照してください。それらのファイルは、'My Documents'¥DB2LOG¥ ディレクトリーに入っています。 My Documents ディレクトリーのロケーションは、ご使用のコンピューターの設定によって異なります。

db2wi.log ファイルには、最後の DB2 インストール情報が入れられます。 db2.log には、DB2 製品インストールの履歴が入れられます。

関連概念:

- 334 ページの『DB2 インフォメーション・センター』
- 336 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 製品のインストール (Windows)』
- 345 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 346 ページの『DB2 インフォメーション・センターのトピックを特定の言語で表示する方法』
- 344 ページの『DB2 インフォメーション・センターの呼び出し』
- 339 ページの『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 (UNIX)』

関連資料:

- 「*コマンド・リファレンス*」の『db2rspgn - 応答ファイル生成プログラム・コマンド』

DB2 インフォメーション・センターの呼び出し

DB2 インフォメーション・センターは、Linux、UNIX、および Windows オペレーティング・システム用の DB2 製品 (DB2 Universal Database、 DB2 Connect、DB2 Information Integrator、 DB2 Query Patroller など) を使用するために必要なすべての情報を提供します。

DB2 インフォメーション・センターは、以下の場所から呼び出すことができます。

- DB2 UDB クライアントまたはサーバーがインストールされているコンピューター
- DB2 インフォメーション・センターがインストールされているイントラネット・サーバーまたはローカル・コンピューター
- IBM の Web サイト

前提条件:

DB2 インフォメーション・センターを呼び出すための要件は、以下のとおりです。

- *オプショナル*: 希望する言語でトピックを表示するようブラウザーを構成する
- *オプショナル*: コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを使用するよう DB2 クライアントを構成する

手順:

DB2 UDB クライアントまたはサーバーがインストールされているコンピューターから DB2 インフォメーション・センターを呼び出すには、以下のようにします。

- (Windows オペレーティング・システムの)「スタート」メニューから: **「スタート」→「プログラム」→「IBM DB2」→「情報」→「インフォメーション・センター」**をクリックします。
- コマンド行プロンプトから:
  - Linux および UNIX オペレーティング・システムの場合、 **db2icdocs** コマンドを発行します。
  - Windows オペレーティング・システムの場合、 **db2icdocs.exe** コマンドを発行します。

イントラネット・サーバーまたはローカル・コンピューターにインストール済みの DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ http://<host-name>:<port-number>/ を開きます (<host-name> はホスト名、 <port-number> は DB2 インフォメーション・センターを利用可能なポート番号)。

IBM Web サイトにある DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ publib.boulder.ibm.com/infocenter/db2help/ を開きます。

関連概念:

- 334 ページの『DB2 インフォメーション・センター』

関連タスク:

- 346 ページの『DB2 インフォメーション・センターのトピックを特定の言語で表示する方法』
- 355 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 345 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 356 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 356 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 357 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール

http://publib.boulder.ibm.com/infocenter/db2help/ から利用できる DB2 インフォメーション・センターは、資料の新規追加または変更によって定期的に更新されます。さらに、更新された DB2 インフォメーション・センターをコンピューターまたはイントラネット・サーバーにダウンロードしてインストールできる場合もあります。DB2 インフォメーション・センターを更新しても、 DB2 クライアント製品またはサーバー製品は更新されません。

前提条件:

インターネットに接続されたコンピューターへのアクセスが必要です。

手順:

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

1. IBM の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターを開きます。
2. 「DB2 インフォメーション・センターにようこそ」ページの見出し「サービスおよびサポート」の「ダウンロード」セクションで、**「DB2 資料」**リンクをクリックします。
3. 最新のドキュメンテーション・イメージのレベルと、インストール済みのドキュメンテーション・レベルを比較して、 DB2 インフォメーション・センターを更新する必要があるかどうかを確認します。「DB2 インフォメーション・センターにようこそ」ページに、インストール済みのドキュメンテーションのレベルがリストされます。
4. より新しいバージョンの DB2 インフォメーション・センターが存在する場合、ご使用のオペレーティング・システムに対応する最新の *DB2* インフォメーション・センター・イメージをダウンロードします。
5. 最新の *DB2* インフォメーション・センター・イメージをインストールするには、 Web ページの指示に従ってください。

関連概念:

- 336 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 344 ページの『DB2 インフォメーション・センターの呼び出し』
- 339 ページの『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 (UNIX)』
- 342 ページの『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 (Windows)』

DB2 インフォメーション・センターのトピックを特定の言語で表示する方法

DB2 インフォメーション・センターを表示した場合、ブラウザーの設定値で指定された言語でトピックを表示することが試行されます。希望する言語に翻訳されていないトピックがある場合、そのトピックは英語で表示されます。

手順:

Internet Explorer ブラウザーにおいて、希望する言語でトピックを表示するには、

1. Internet Explorer で、**「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」ウィンドウが表示されます。
2. 言語リストの中で、希望する言語が最初の項目として指定されていることを確認してください。

- 新しい言語をリストに追加するには、**「追加...」**ボタンをクリックします。

  注: 言語を追加しても、その言語でトピックを表示するために必要なフォントがそのコンピューター上にあることが保証されるわけではありません。

- ある言語をリストの先頭に移動するには、その言語を選択してから、その言語が言語リストの先頭になるまで**「上へ」**ボタンをクリックします。

3. DB2 インフォメーション・センターが希望する言語で表示されるようにするため、ページをリフレッシュします。

Mozilla 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、

1. Mozilla で、**「編集 (Edit)」**→**「設定 (Preferences)」**→**「言語 (Languages)」**ボタンを選択します。「設定 (Preferences)」ウィンドウに「言語 (Languages)」パネルが表示されます。
2. 言語リストの中で、希望する言語が最初の項目として指定されていることを確認してください。
   - 新しい言語をリストに追加するには、**「追加... (Add...)」**ボタンをクリックします。
   - ある言語をリストの先頭に移動するには、その言語を選択してから、その言語が言語リストの先頭になるまで**「上へ」**ボタンをクリックします。
3. DB2 インフォメーション・センターが希望する言語で表示されるようにするため、ページをリフレッシュします。

関連概念:

- 334 ページの『DB2 インフォメーション・センター』

DB2 PDF 資料および印刷された資料

以下の表は、正式な資料名、資料番号、および PDF ファイル名を示しています。ハードコピー版の資料を注文するには、正式な資料名を知っておく必要があります。 PDF ファイルを印刷するには、PDF ファイル名を知っておく必要があります。

DB2 資料は、以下のカテゴリーに分類されています。

- DB2 中核情報
- 管理情報
- アプリケーション開発情報
- ビジネス・インテリジェンス情報
- DB2 Connect 情報
- 入門情報
- チュートリアル情報
- オプショナル・コンポーネント情報
- リリース・ノート

以下の表は、DB2 ライブラリー内の各資料について、その資料のハードコピー版を注文したり、PDF 版を印刷または表示したりするのに必要な情報を示しています。

DB2 ライブラリー内の各資料に関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。

DB2 の基本情報

こうした資料の情報は、すべての DB2 ユーザーに基本的なもので、プログラマーおよびデータベース管理者にとって役立つ情報であるとともに、DB2 Connect、DB2 Warehouse Manager、または他の DB2 製品を使用するユーザーにとっても役立つ内容です。

表 99. DB2 の基本情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database コマンド・リファレンス*」	SC88-9140	db2n0j81
「*IBM DB2 Universal Database 用語集*」	資料番号なし	db2t0j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 1 巻*」	GC88-9152 (ハードコピーなし)	db2m1j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 2 巻*」	GC88-9153 (ハードコピーなし)	db2m2j81
「*IBM DB2 Universal Database 新機能*」	SC88-9158	db2q0j81

管理情報

これらの資料の情報は、DB2 データベース、データウェアハウス、およびフェデレーテッド・システムを効果的に設計し、インプリメントし、保守するために必要なトピックを扱っています。

表 100. 管理情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database 管理ガイド: プランニング*」	SC88-9135	db2d1j81
「*IBM DB2 Universal Database 管理ガイド: インプリメンテーション*」	SC88-9133	db2d2j81
「*IBM DB2 Universal Database 管理ガイド: パフォーマンス*」	SC88-9134	db2d3j81
「*IBM DB2 Universal Database 管理 API リファレンス*」	SC88-9136	db2b0j81
「*IBM DB2 Universal Database データ移動ユーティリティーガイドおよびリファレンス*」	SC88-9142	db2dmj81
「*IBM DB2 Universal Database データ・リカバリーと高可用性ガイドおよびリファレンス*」	SC88-9143	db2haj81

表 100. 管理情報 (続き)

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database データウェアハウス・センター管理ガイド*」	SC88-9165	db2ddj81
「*IBM DB2 Universal Database SQL リファレンス 第 1 巻*」	SC88-9155	db2s1j81
「*IBM DB2 Universal Database SQL リファレンス 第 2 巻*」	SC88-9156	db2s2j81
「*IBM DB2 Universal Database システム・モニター ガイドおよびリファレンス*」	SC88-9157	db2f0j81

アプリケーション開発情報

これらの資料の情報は、 DB2 Universal Database (DB2 UDB) のアプリケーション開発者またはプログラマーが特に関心を持つ内容です。サポートされるさまざまなプログラミング・インターフェース (組み込み SQL、ODBC、JDBC、SQLJ、CLI など) を使用して DB2 UDB にアクセスするのに必要な資料とともに、サポートされる言語およびコンパイラーについても紹介されています。また、DB2 インフォメーション・センターをご使用の場合には、サンプル・プログラムのソース・コードの HTML バージョンにアクセスすることもできます。

表 101. アプリケーション開発情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database アプリケーション開発ガイド アプリケーションの構築および実行*」	SC88-9137	db2axj81
「*IBM DB2 Universal Database アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」	SC88-9138	db2a1j81
「*IBM DB2 Universal Database アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」	SC88-9139	db2a2j81
「*IBM DB2 Universal Database コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻*」	SC88-9159	db2l1j81
「*IBM DB2 Universal Database コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻*」	SC88-9160	db2l2j81
「*IBM DB2 Universal Database データウェアハウス・センター アプリケーション統合ガイド*」	SC88-9166	db2adj81

表 101. アプリケーション開発情報 (続き)

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database XML Extender 管理およびプログラミングのガイド*」	SC88-9172	db2sxj81

ビジネス・インテリジェンス情報

これらの資料の情報は、さまざまなコンポーネントを使用して、 DB2 Universal Database のデータウェアハウジング機能および分析機能を拡張する方法を説明しています。

表 102. ビジネス・インテリジェンス情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Warehouse Manager Standard Edition インフォメーション・カタログ・センター 管理ガイド*」	SC88-9167	db2dij81
「*IBM DB2 Warehouse Manager Standard Edition インストール・ガイド*」	GC88-9164	db2idj81
「*IBM DB2 Warehouse Manager Standard Edition DB2 Warehouse Manager を使用時の ETI ソリューション・コンバージョン・プログラムの管理*」	SC88-9894	iwhe1mstx80

DB2 Connect 情報

このカテゴリーの情報は、 DB2 Connect Enterprise Edition または DB2 Connect Personal Edition を使用して、メインフレーム・サーバーおよびミッドレンジ・サーバー上のデータにアクセスする方法を説明しています。

表 103. DB2 Connect 情報

資料名	資料番号	PDF ファイル名
「*IBM コネクティビティー 補足*」	資料番号なし	db2h1j81
「*IBM DB2 Connect Enterprise Edition 概説およびインストール*」	GC88-9145	db2c6j81
「*IBM DB2 Connect Personal Edition 概説およびインストール*」	GC88-9146	db2c1j81
「*IBM DB2 Connect ユーザーズ・ガイド*」	SC88-9147	db2c0j81

入門情報

このカテゴリーの情報は、サーバー、クライアント、および他の DB2 製品をインストールして構成する場合に役立ちます。

表 104. 入門情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database DB2 クライアント機能 概説およびインストール*」	GC88-9144 (ハードコピーなし)	db2itj81
「*IBM DB2 Universal Database DB2 サーバー機能 概説およびインストール*」	GC88-9148	db2isj81
「*IBM DB2 Universal Database DB2 Personal Edition 概説およびインストール*」	GC88-9150	db2i1j81
「*IBM DB2 Universal Database インストールおよび構成 補足*」	GC88-9149 (ハードコピーなし)	db2iyj81
「*IBM DB2 Universal Database DB2 Data Links Manager 概説およびインストール*」	GC88-9141	db2z6j81

チュートリアル情報

チュートリアル情報は、DB2 機能を紹介し、さまざまなタスクを実行する方法を示します。

表 105. チュートリアル情報

資料名	資料番号	PDF ファイル名
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介*」	資料番号なし	db2tuj81
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド*」	資料番号なし	db2taj81
「*インフォメーション・カタログ・センター チュートリアル*」	資料番号なし	db2aij81
「*Video Central for e-business チュートリアル*」	資料番号なし	db2twj81
「*Visual Explain チュートリアル*」	資料番号なし	db2tvj81

オプショナル・コンポーネント情報

このカテゴリーの情報は、 DB2 のオプショナル・コンポーネントを使用する方法について説明しています。

表 106. オプショナル・コンポーネント情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Cube Views Guide and Reference*」	SC18–7298	db2aax81
「*IBM DB2 Query Patroller インストール、管理、使用法のガイド*」	GC88–9154	db2dwj81
「*IBM DB2 Spatial Extender and Geodetic Extender ユーザーズ・ガイドおよびリファレンス*」	SC88-9171	db2sbj81
「*IBM DB2 Universal Database Data Links Manager 管理ガイドおよびリファレンス*」	SC88-9169	db2z0x82
「*DB2 Net Search Extender 管理およびユーザーズ・ガイド*」 **注:** この資料の HTML 版は、HTML ドキュメンテーション CD からインストールされません。	SH88-8546	N/A

リリース・ノート

リリース・ノートは、ご使用の製品のリリースおよびフィックスパック・レベルに特有の追加情報を紹介します。また、リリース・ノートには、各リリース、アップデート、およびフィックスパックで組み込まれた資料上の更新の要約も含まれています。

表 107. リリース・ノート

資料名	資料番号	PDF ファイル名
「*DB2 リリース・ノート*」	「注」を参照。	「注」を参照。
「*DB2 インストール情報*」	製品 CD-ROM でのみ参照可能。	使用できません。

注: リリース・ノートは以下の形式で入手できます。

- XHTML およびテキスト形式 (製品 CD 内)
- PDF 形式 (PDF ドキュメンテーション CD 内)

さらに、リリース・ノートの中で、『*既知の問題と予備手段*』および『*リリース間の非互換性*』に関する部分は DB2 インフォメーション・センターにも表示されます。

UNIX ベースのプラットフォームでテキスト形式でリリース・ノートを確認するには、 `Release.Notes` ファイルを参照してください。このファイルは、`DB2DIR/Readme/%L` ディレクトリーに収録されています。 `%L` はロケール名を表しています。 `DB2DIR` は以下になります。

- AIX オペレーティング・システムの場合: `/usr/opt/db2_08_01`
- その他のすべての UNIX ベースのオペレーティング・システムの場合: `/opt/IBM/db2/V8.1`

関連概念:

- 333 ページの『DB2 ドキュメンテーションおよびヘルプ』

関連タスク:

- 353 ページの『PDF ファイルからの DB2 資料の印刷方法』
- 354 ページの『DB2 の印刷資料の注文方法』
- 355 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』

PDF ファイルからの DB2 資料の印刷方法

DB2 PDF ドキュメンテーション CD に収録されている DB2 資料を印刷することができます。 Adobe Acrobat Reader を使用すれば、資料全体または特定のページを印刷できます。

前提条件:

Adobe Acrobat Reader がインストールされていることを確認してください。 Adobe Acrobat Reader をインストールする必要がある場合、 Adobe Web サイト (www.adobe.com) から入手できます。

手順:

PDF ファイルから DB2 資料を印刷するには以下のようにします。

1. *DB2 PDF* ドキュメンテーション CD をドライブに挿入します。 UNIX オペレーティング・システムの場合、 DB2 PDF ドキュメンテーション CD をマウントします。 UNIX オペレーティング・システムで CD をマウントする方法については、「*概説およびインストール*」を参照してください。
2. index.htm を開きます。ブラウザー・ウィンドウにファイルが開きます。
3. 参照したい PDF のタイトルをクリックします。 Acrobat Reader で PDF が開きます。
4. **「ファイル」** → **「印刷」**を選択して、所要の資料の任意の部分を印刷します。

関連概念:

- 334 ページの『DB2 インフォメーション・センター』

関連タスク:

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (AIX)』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『HP-UX 上での CD-ROM のマウント』

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Linux)』
- 354 ページの『DB2 の印刷資料の注文方法』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Solaris)』

関連資料:

- 347 ページの『DB2 PDF 資料および印刷された資料』

DB2 の印刷資料の注文方法

ハードコピー版の資料を望む場合には、以下のいずれかの方法で注文できます。

印刷資料の注文方法:

一部の国または地域では、印刷された資料を注文することもできます。お客様がお住まいの国または地域でこのサービスが利用可能かどうかを確認するには、お住まいの国または地域の IBM Publications Web サイトをご覧ください。資料のご注文が可能な場合、以下のようにすることができます。

- 正規の IBM 製品販売業者または営業担当員に連絡してください。お客様がお住まいの地域の IBM 担当員の情報については、お手数ですが IBM の Web サイト (www.ibm.com/planetwide) の IBM Worldwide Directory of Contacts で確認してください。
- IBM Publications Center (http://www.ibm.com/shop/publications/order) にアクセスしてください。なお、IBM Publications Center から資料を注文できない国もあります。

DB2 製品がご利用可能になった時点で、印刷された資料は *DB2 PDF* ドキュメンテーション *CD* にある PDF 形式の資料と同じものです。さらに、 *DB2* インフォメーション・センター *CD* に収録されている印刷された資料の内容もまた、これらと同じです。ただし、DB2 インフォメーション・センター CD には、 PDF 資料にない追加情報も含まれます (たとえば、SQL 管理作業や HTML サンプル)。 DB2 PDF ドキュメンテーション CD に収録されている資料の中には、ハードコピーとしてご注文できない資料もあります。

注: DB2 インフォメーション・センターは、 PDF またはハードコピー の資料よりも頻繁に更新されます。ドキュメンテーションの更新が入手可能になった時点でインストールするか、 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) を参照して最新の情報を入手してください。

関連タスク:

- 353 ページの『PDF ファイルからの DB2 資料の印刷方法』

関連資料:

- 347 ページの『DB2 PDF 資料および印刷された資料』

DB2 ツールからコンテキスト・ヘルプを呼び出す

コンテキスト・ヘルプは、特定のウィンドウ、ノートブック、ウィザード、またはアドバイザーに関連したタスクまたはコントロールの情報を提供します。コンテキスト・ヘルプは、グラフィカル・ユーザー・インターフェースのある DB2 管理ツールおよび開発ツールから利用できます。コンテキスト・ヘルプには、以下の 2 種類があります。

- それぞれのウィンドウまたはノートブックにある**「ヘルプ」**ボタンからアクセス可能なヘルプ
- infopop (ポップアップ情報ウィンドウ)。これは、マウス・カーソルを特定のフィールドまたはコントロール上に置いたとき、またはウィンドウ、ノートブック、ウィザード、アドバイザー内でフィールドまたはコントロールを選択して F1 を押すと表示されます。

「ヘルプ」ボタンを押すと、概説、前提条件、およびタスク情報が表示されます。infopop は、それぞれのフィールドおよびコントロールについて説明します。

手順:

コンテキスト・ヘルプを呼び出すには、以下のようにします。

- ウィンドウおよびノートブックのヘルプを表示するには、いずれかの DB2 ツールを開始して、任意のウィンドウまたはノートブックを開きます。ウィンドウまたはノートブックの右下隅にある**「ヘルプ」**ボタンをクリックして、コンテキスト・ヘルプを呼び出します。

  また、それぞれの DB2 ツール・センターの上部にある**「ヘルプ」**メニュー項目からコンテキスト・ヘルプにアクセスすることもできます。

  ウィザードおよびアドバイザーでは、最初のページの「タスクの概要」リンクをクリックすると、コンテキスト・ヘルプを表示できます。
- ウィンドウまたはノートブック上の各コントロールの infopop ヘルプを表示するには、コントロールをクリックしてから、**F1** を押します。コントロールの詳細情報を示すポップアップ情報が、黄色いウィンドウに表示されます。

  注: フィールドまたはコントロールにマウス・カーソルを置いておくだけで infopops が表示されるようにするには、「ツール設定」ノートブックの**「文書 (Documentation)」**ページの**「infopops の自動表示」**チェック・ボックスを選択します。

  infopop に似た別のコンテキスト・ヘルプに、診断ポップアップ情報があります。これにはデータ入力規則が示されます。診断ポップアップ情報は、無効または不十分なデータが入力されたとき、紫色のウィンドウに表示されます。診断ポップアップ情報は、以下に関して表示されます。

  – 必須フィールド。
  – 日付フィールドのように、正確なフォーマットを必要とするデータのフィールド。

関連タスク:

- 344 ページの『DB2 インフォメーション・センターの呼び出し』
- 356 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』

- 356 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 357 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』
- 『DB2 UDB ヘルプの使用法: Common GUI help』

コマンド行プロセッサーからメッセージ・ヘルプを呼び出す

メッセージ・ヘルプは、メッセージが出された原因と、エラーへの応答として実行すべきアクションを説明します。

手順:

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

```
? XXXnnnnn
```

ここで、*XXXnnnnn* は有効なメッセージ ID を表します。

たとえば、`? SQL30081` と入力すると、メッセージ SQL30081 に関するヘルプを表示します。

関連概念:

- 「*メッセージ・リファレンス 第 1 巻*」の『メッセージの概要』

関連資料:

- 「*コマンド・リファレンス*」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーからコマンド・ヘルプを呼び出す

コマンド・ヘルプは、コマンド行プロセッサーでのコマンドの構文を説明します。

手順:

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

```
? command
```

ここで *command* はキーワードまたはコマンド全体を表します。

たとえば、`? catalog` と入力すると、すべての CATALOG コマンドに関するヘルプが表示され、 `? catalog database` と入力すると、 CATALOG DATABASE コマンドのヘルプだけが表示されます。

関連タスク:

- 355 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 344 ページの『DB2 インフォメーション・センターの呼び出し』
- 356 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 357 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーから SQL 状態ヘルプを呼び出す

DB2 Univerrsal Database は、 SQL ステートメントの結果の原因となったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順:

SQL 状態ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

`? sqlstate` または `? class code`

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。

たとえば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

関連タスク:

- 344 ページの『DB2 インフォメーション・センターの呼び出し』
- 356 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 356 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』

DB2 チュートリアル

DB2® チュートリアルは、 DB2 Universal Database のさまざまな機能について学習するのを支援します。このチュートリアルでは、アプリケーションの開発、SQL 照会のパフォーマンス調整、データウェアハウスの処理、メタデータの管理、および DB2 を使用した Web サービスの開発の各分野で、段階的なレッスンが用意されています。

はじめに:

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

チュートリアルの中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、それぞれのチュートリアルを参照してください。

DB2 Universal Database チュートリアル:

以下に示すチュートリアルのタイトルをクリックすると、そのチュートリアルを表示できます。

ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介
データウェアハウス・センターを使用して簡単なデータウェアハウジング・タスクを実行します。

ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド
データウェアハウス・センターを使用して高度なデータウェアハウジング・タスクを実行します。

インフォメーション・カタログ・センター・チュートリアル
インフォメーション・カタログを作成および管理して、インフォメーション・カタログ・センターを使用してメタデータを配置し使用します。

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

DB2 トラブルシューティング情報

DB2® 製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション
トラブルシューティング情報は、DB2 インフォメーション・センター、および DB2 ライブラリーに含まれる PDF 資料の中でご利用いただけます。DB2 インフォメーション・センターで、 (ブラウザー・ウィンドウの左側の) ナビゲーション・ツリーの「サポートおよびトラブルシューティング (Support and troubleshooting)」ブランチを参照すると、 DB2 トラブルシューティング・ドキュメンテーションの詳細なリストが見つかります。

DB2 Technical Support の Web サイト
現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、 DB2 Technical Support の Web サイトを参照してください。Technical Support サイトには、最新の DB2 出版物、TechNotes、プログラム診断依頼書 (APAR)、フィックスパック、DB2 内部エラー・コードの最新リスト、その他のリソースが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/udb/winos2unix/support) にアクセスしてください。

DB2 Problem Determination Tutorial Series
DB2 製品で作業中に直面するかもしれない問題を素早く識別し、解決する方法に関する情報を見つけるには、 DB2 Problem Determination Tutorial Series の Web サイトを参照してください。あるチュートリアルでは、使用可能な DB2 問題判別機能およびツールを紹介し、それらをいつ使用すべきかを判断する助けを与えます。別のチュートリアルは、『データベース・エンジン問題判別 (Database Engine Problem Determination)』、『パフォーマンス問題判別 (Performance Problem Determination)』、『アプリケーション問題判別 (Application Problem Determination)』などの関連トピックを扱っています。

DB2 Technical Support (http://www.ibm.com/software/data/support/pdm/db2tutorials.html) には、 DB2 問題判別チュートリアルがすべて揃っています。

関連概念:

- 334 ページの『DB2 インフォメーション・センター』
- *トラブルシューティング・ガイド* の『Introduction to Problem Determination - DB2 テクニカル・サポートのチュートリアル』

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 ユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 360 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 360 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 360 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

アクセスしやすい表示

DB2 ツールには、視力の弱いユーザー、その他の視力障害をもつユーザーのためにアクセシビリティーを向上させる機能が備わっています。これらのアクセシビリティー拡張機能には、フォント・プロパティーのカスタマイズを可能にする機能も含まれています。

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、メニューおよびテキストのフォントを変更する: Common GUI help を参照してください。

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

DB2 形式は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。 XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン・ドキュメンテーションにアクセスする場合にのみ使用できます。

関連概念:

- 360 ページの『ドット 10 進シンタックス・ダイアグラム』

ドット 10 進シンタックス・ダイアグラム

スクリーン・リーダーを使用してインフォメーション・センターを利用するユーザーのために、シンタックス・ダイアグラムがドット 10 進形式で提供されます。

ドット 10 進形式では、各シンタックス・エレメントは別々の行に書き込まれます。複数のシンタックス・エレメントが常に同時に存在する (または常に同時に不在の) 場合、単一のコンパウンド・シンタックス・エレメントとみなせるので同一行に表示できます。

各行は、ドット 10 進数で開始します。たとえば、3 または 3.1 ないしは 3.1.1 です。こうした数を適切に聞き取るには、スクリーン・リーダーが句読点を読み取るように設定されていることを確認してください。同じドット 10 進数を持つすべてのシンタックス・エレメント (たとえば、 3.1 という数値を持つすべてのシンタックス・エレメント) は、相互に排他的な代替エレメントです。 3.1 USERID および 3.1 SYSTEMID という行を聞き取る場合、シンタックスには両方ではなく USERID または SYSTEMID のどちらかが含まれることが分かります。

ドット 10 進レベルは、ネストのレベルを表示します。たとえば、ドット 10 進数 3 のシンタックス・エレメントの後に、一連のドット 10 進数 3.1 のシンタックス・エレメントが続きます。 3.1 の番号が付されたシンタックス・エレメントすべては、番号 3 の付されたシンタックス・エレメントに従属します。

シンタックス・エレメントに関する情報を追加するため、ドット 10 進数の次に特定のワードおよびシンボルが使用されます。 時折、こうしたワードおよびシンボルはエレメントの最初に表示される場合もあります。 簡単に識別するため、ワードやシンボルがシンタックス・エレメントの一部である場合には、円記号 (¥) 文字が先頭に付きます。 * シンボルはドット 10 進数の次に使用でき、シンタックス・エレメントが反復することを示します。 たとえば、ドット 10 進数 3 のシンタックス・エレメント *FILE は、3 ¥* FILE という形式になります。 3* FILE という形式は、シンタックス・エレメント FILE が反復されることを示します。 3* ¥* FILE という形式は、シンタックス・エレメント * FILE が反復されることを示します。

シンタックス・エレメントのストリングを分離するのに使用されるコンマなどの文字は、シンタックス内の分離する項目の直前に表示されます。 こうした文字は、それぞれの項目と同一行に表示するか、同じドット 10 進数を持つ関連する項目のある別の行に表示できます。 またその行には、シンタックス・エレメントに関する情報を提供する別のシンボルを表示することも可能です。 たとえば、複数の LASTRUN および DELETE シンタックス・エレメントを使用している場合には、5.1*、5.1 LASTRUN、および 5.1 DELETE という行は、エレメントをコンマで区切る必要があります。区切り文字が指定されないと、各シンタックス・エレメントを区切るのにブランクが使用されると想定されます。

シンタックス・エレメントの前に % シンボルが付く場合、他の箇所で定義されている参照であることを示します。 % シンボルの後のストリングは、リテラルではなくシンタックス・フラグメントの名前です。 たとえば、2.1 %OP1 という行は別のシンタックス・フラグメント OP1 を参照すべきことを意味します。

以下のワードおよびシンボルが、ドット 10 進数の次に使用されます。

- ? は、オプションのシンタックス・エレメントであることを表します。? シンボルが後に続くドット 10 進数は、対応するドット 10 進数のシンタックス・エレメント、および任意の従属のシンタックス・エレメントがオプションであることを示します。ドット 10 進数の付いたシンタックス・エレメントが 1 つしかない場合、 ? シンボルはそのシンタックス・エレメントと同じ行に表示されます (たとえば、5? NOTIFY)。ドット 10 進数の付いたシンタックス・エレメントが複数

ある場合、 ? シンボルだけで行に表示され、その後にオプションのシンタックス・エレメントが続きます。たとえば、「5 ?、5 NOTIFY、および 5 UPDATE」という行を聞き取る場合、シンタックス・エレメント NOTIFY および UPDATE がオプションである、つまりそのいずれかを選択でき、どちらも選択しないこともできることが分かります。 ? シンボルは、線路型ダイアグラムのバイパス線に相当します。

- ! は、デフォルトのシンタックス・エレメントであることを表します。! シンボルおよびシンタックス・エレメントが後に続くドット 10 進数は、そのシンタックス・エレメントが、同じドット 10 進数を共用するシンタックス・エレメントすべてのデフォルト・オプションであることを示します。同じドット 10 進数を共用するシンタックス・エレメントのうち 1 つだけに、 ! シンボルを指定できます。たとえば、「2? FILE、2.1! (KEEP)、および 2.1 (DELETE)」という行を聞き取る場合、 FILE キーワードのデフォルト・オプションは (KEEP) になります。この例では、FILE キーワードを含めてもオプションを指定しない場合には、デフォルト・オプション KEEP が適用されます。デフォルト・オプションは、次に高位のドット 10 進数にも適用されます。 この例の場合、FILE キーワードが省略されると、デフォルトの FILE(KEEP) が使用されます。しかし、「2? FILE、2.1、2.1.1! (KEEP)、および 2.1.1 (DELETE)」という行を聞き取る場合、デフォルト・オプション KEEP は次に高位のドット 10 進数 2.1 (関連キーワードを持っていない) にのみ適用され、 2? FILE には適用されません。キーワード FILE が省略されると、どれも使用されません。
- * は、0 回以上反復できるシンタックス・エレメントを示します。 * シンボルが後に続くドット 10 進数は、このシンタックス・エレメントが 0 回以上使用できること、つまりオプションであり、なおかつ反復できることを表します。たとえば、5.1* データ域という行を聞き取る場合、 1 つまたは複数のデータ域を含めるか、またはデータ域を全く含めないことが可能です。「3*、3 HOST、および 3 STATE」という行を聞き取る場合、 HOST、STATE をどちらか一方または両方同時に含めるか、どちらも含めないことができます。

  注:

  1. ドット 10 進数の後にアスタリスク (*) が付き、ドット 10 進数の付いた項目が 1 つしかない場合には、同じ項目を複数回反復できます。
  2. ドット 10 進数の後にアスタリスクが付き、ドット 10 進数の付いた項目が複数ある場合、リストから複数の項目を使用できますが、各項目を複数回使用することはできません。 前述の例では、HOST STATE と書くことはできますが、HOST HOST とは書けません。
  3. * シンボルは、線路型シンタックス・ダイアグラムのループバック線に相当します。
- + は、1 回以上含める必要のあるシンタックス・エレメントであることを示します。 + シンボルが後に続くドット 10 進数は、このシンタックス・エレメントを 1 回以上含める必要があること、つまり少なくとも 1 回は含める必要があり、反復できることを表します。たとえば、「6.1+ データ域」という行を聞き取る場合、データ域を少なくとも 1 回は含めなければなりません。「2+、2 HOST、および 2 STATE」という行を聞き取る場合には、 HOST、STATE、またはその両方を含める必要があります。 * シンボルと同様に、+ シンボルは、ドット 10 進

数の付いた項目が 1 つしかない場合に限り、その特定の項目のみを反復できます。 * シンボルと同様、 + シンボルは線路型シンタックス・ダイアグラムのループバック線に相当します。

関連概念:

- 359 ページの『アクセス支援』

関連タスク:

- 『目次』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『構文図の見方』

DB2 Universal Database 製品の共通基準認証

DB2 Universal Database は、 Common Criteria の評価検定レベル 4 (EAL4) で認証の評価を受けています。 Common Criteria の詳細については、以下の Common Criteria の Web サイトを参照してください。 http://niap.nist.gov/cc-scheme/

付録 E. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

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

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

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

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

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

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

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

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

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

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

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

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

著作権使用許諾:

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

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

商標

以下は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

以下は、それぞれ各社の商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Pentium は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

索引

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

[ア行]

[カ行]

［サ行］

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[数字]

A

B

C

D

E

F

G

H

I

L

M

N

O

R

S

T

U

V

W

X

IBM と連絡をとる

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Universal Database 製品に関する情報は、
http://www.ibm.com/software/data/db2/udb から入手できます。

このサイトには、技術ライブラリー、資料の注文方法、製品のダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースへのリンクに関する最新情報が掲載されています。

米国以外の国で IBM に連絡する方法については、 IBM Worldwide ページ (www.ibm.com/planetwide) にアクセスしてください。

Printed in Japan

SC88-9135-01

日本アイ･ビー･エム株式会社
〒106-8711　東京都港区六本木3-2-12