IBM® DB2® ユニバーサル・データベース

管理の手引き: 計画

バージョン 7

SC88-8513-01
(英文原典：SC09-2946-01)

IBM® DB2® ユニバーサル・データベース

管理の手引き: 計画

バージョン 7

SC88-8513-01
(英文原典：SC09-2946-01)

ご注意!

本書、および本書がサポートする製品をご使用になる前に、 295ページの『付録F. 特記事項』にある一般的な情報を必ずお読みください。

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

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)

原 典:	SC09-2946-01 IBM® DB2® Universal Database Administration Guide: Planning Version 7
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 2001.8

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

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

目次

本書について

3 巻で構成される本書には、 2000 年問題に対応した 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 (*管理の手引き: 計画*)
- データベースの使用および管理についての情報 (*管理の手引き: 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 (*管理の手引き: 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、*コマンド解説書* 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 解説書* を参照してください。
- **コントロール・センター**。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グラフィックを使用して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グラフィックを使用してセットアップするための**複製管理**が含まれています。さらに、コントロール・センターでは、グラフィカル・ユーザー・インターフェースを介して DB2 ユーティリティー機能を実行できます。コントロール・センターを呼び出す方法は、プラットフォームによって異なります。たとえば、 OS/2 ではコマンド行で `db2cc` コマンドを使用し、 Windows プラットフォームでは DB2 フォルダーから コントロール・センター アイコンを選択するか、開始パネルを使用します。紹介のヘルプを表示するには、「コントロール・センター (Control Center)」ウィンドウの**「ヘルプ (Help)」**プルダウンから**「入門 (Getting started)」**を選択してください。 **Visual Explain** 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。

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。

- スクリプト・センターは、スクリプトと呼ばれる小さなアプリケーションを保管します。これらのスクリプトには、オペレーティング・システムのコマンドだけでなく、SQL ステートメントや DB2 コマンドを含めることができます。
- アラート・センターは、他の DB2 操作から出されるメッセージをモニターします。
- ツール設定は、コントロール・センター、アラート・センター、および複製について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 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。

DB2 ユニバーサル・データベースの世界

- 『第1章 DB2 ユニバーサル・データベースの管理』では、 DB2 ユニバーサル・データベースを紹介し、概要を説明します。

データベースの概念

- 『第2章 リレーショナル・データベースの基本的な概念』では、ストレージ・オブジェクトやシステム・オブジェクトを含むデータベース・オブジェクトの概要を説明します。
- 『第3章 連合システム』では、連合システム (federated database system: 複数のデータベースから構成されるが、単一のデータベース・イメージを提供するデータベース・システムを意味します) について説明します。連合システムはデータベース管理システム (DBMS) の一種で、アプリケーションやユーザーが 1 つのステートメント内で 2 つ以上の DBMS またはデータベースを参照する SQL ステートメントを実行依頼できます。
- 『第4章 並列データベース・システム』では、 DB2 で実現される並列性のタイプを紹介します。
- 『第5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。
- 『第6章 地理情報エクステンダーについて』では、地理情報エクステンダーを紹介し、この目的とこれが処理するデータについて説明します。

データベースの設計

- 『第7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8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 (データ・ストレージに関する考慮事項を含む) の指針について説明します。
- 『第9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10章 トランザクション・マネージャーの設計』では、 CICS などの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。
- 『第11章 高可用性とフェールオーバー・サポートの紹介』では、 DB2 が提供する高可用性フェールオーバー・サポートの概要を説明します。

付録

- 『付録A. 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。
- 『付録B. データベース・マイグレーションの計画』では、バージョン 7 へのデータベースのマイグレーションについて説明します。
- 『付録C. リリース間の非互換性』では、バージョン 7 までに導入された、互換性のない機能について説明します。
- 『付録D. 各国語サポート (NLS)』では、国、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E. DB2 ライブラリーの使用法』では、ウィザード、オンライン・ヘルプ、メッセージ、およびマニュアルを含む DB2 ライブラリーの構造について説明します。

管理の手引きの他の巻の概要

管理の手引き : インプリメンテーション

管理の手引き: 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。

コントロール・センターによる管理

- 『GUI ツールを使用した DB2 の管理』では、データベースの管理に使用されるグラフィカル・ユーザー・インターフェース (GUI) ツールについて説明します。

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。
- 『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。

- 『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除去に関するタスクについて説明します。

データベースのセキュリティー

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

データの移動

- 『データの移動に使用するユーティリティー』は 1 ページで構成されており、データを移動するさまざまな方法について紹介し、 *データ移動ユーティリティー手引きおよび解説書* を読むための基礎的な情報を提供します。

リカバリー

- 『データベースのリカバリー』は 1 ページで構成されており、データベースのバックアップ、復元、およびロールフォワードの概念を紹介しています。さらに詳しい情報は、*データ回復と高可用性の手引きと解説書* を参照してください。

付録

- 『分散コンピューティング環境 (DCE) ディレクトリー・サービス』では、 DCE 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。
- 『データベース・リカバリー用のユーザー出口』では、ユーザー出口プログラムでデータベース・ログ・ファイルを使用する方法について説明し、いくつかのサンプル・ユーザー出口プログラムを示します。
- 『複数のデータベース区画サーバーに対するコマンドの発行』では、コマンドを区分データベース環境内のすべての区画に送るための *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。
- 『DB2 (Windows NT 版) 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 NT 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 NT または Windows 2000 データベース区画サーバーの処理』では、Windows NT または Windows 2000 のデータベース区画サーバーで作業することを可能にするユーティリティーについての情報を提供します。
- 『複数の論理ノードの構成』では、区分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。
- 『高速ノード間通信』では、 DB2 ユニバーサル・データベースで、仮想インターフェース・アーキテクチャーを使用できるようにする方法について説明します。

- 『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。
- 『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。

管理の手引き : パフォーマンス

管理の手引き: 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 ユニバーサル・データベース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

- 『パフォーマンスの要素』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。
- 『構造と処理の概要』では、基礎となる DB2 ユニバーサル・データベース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、および実行時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ある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。

- 『データベース区画間でのデータの再配分』では、区画間でデータを再配分するために、区分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および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 DB2 ユニバーサル・データベースの世界

第1章 DB2 ユニバーサル・データベースの管理

DB2 は、広範囲のハードウェア構成で実行するための柔軟性を提供します。これによって、特定の DB2 製品の構成を使用して、ハードウェアとアプリケーションの要件が最適に一致するものを選択することができます。

DB2 はまた、複雑さのレベルが異なるデータベース環境をサポートします。各環境ごとに固有の考慮事項と作業があります。これらについては、 *管理の手引き* および DB2 ライブラリーにある他の資料で詳しく説明されています (273ページの『付録E. DB2 ライブラリーの使用法』を参照)。場合によっては、本書のセクション全体が、ある固有の環境のみに該当する場合もあります。本書のまえがき (『本書について』) を読むと、*管理の手引き* のこの巻および他の巻 (*管理の手引き: インプリメンテーション* および *管理の手引き: パフォーマンス*) のどの章がビジネスの必要に適しているかを知ることができます。

リレーショナル・データベース管理システム (RDBMS)、または DB2 が始めてである場合、『リレーショナル・データベース管理システムの概念』が役立つでしょう。これらの概念に精通している場合、またはこれらを検討する必要がない場合は、このセクションをスキップして、以下のような高度なトピックを説明しているセクションに直接行ってください。

- 『連合システム』。このセクションでは、 1 つのステートメント内で 2 つ以上の DBMS またはデータベースを参照する SQL ステートメントを実行依頼するアプリケーションとユーザーをサポートするデータベース管理システム (DBMS) について説明します。
- 『並列データベース』。このセクションでは、 DB2 で実現される並列性のタイプを紹介します。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。
- 『分散トランザクション処理』。このセクションでは、単一トランザクションで複数のデータベースにアクセスする方法、および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。

DB2 は、以下のような最も特殊化されているデータ管理要件に対応できます。

- *複製*。データを定期的に複数のリモート・データベースにコピーすることを可能にします。マスター・データベースからの更新を他のデータベースに自動的にコピーする必要がある場合、 DB2 の複製機能を使用して、コピーするデータ、データをコピーする先のデータベース表、および更新をコピーする頻度を指定できます。 DB2 の複製機能を使用したい場合は、 *DB2 レプリケーションの手引きおよび解説書* を参照してください。この資料は DB2 データ複製の概念を紹介しており、複製環境を計画し、構成し、管理する方法について説明しています。

- データウェアハウジング。「情報データ」(操作可能データから抜き出され、エンド・ユーザーの意思決定用に変換されたデータ) のストアを作成できます。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操作可能データベースからコピーして、データの要約を計算し、別個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。このデータベース (ウェアハウス) には、操作可能データベースに影響を与えることなく照会できます。データウェアハウジングの詳細については、 *データウェアハウスセンター 管理の手引き* を参照してください。
- *地理情報システム* (GIS)。地理情報エクステンダーを使用して作成できます。 GIS はオブジェクト、データ、およびアプリケーションの複合体であり、これにより地形についての地理情報を生成して分析することが可能になります。地理情報エクステンダーでは、地形を表または視点の行、またはそのような行の一部によって表すことができます。地理情報エクステンダーの使用に関する詳細については、 *地理情報エクステンダー 使用者の手引きおよび解説書* を参照してください。

管理の手引き: 計画 は、DB2 での論理データベース設計と物理データベース設計に関する考慮事項を含め、データベース設計も扱います。計画に関する他の問題、たとえばデータベースのマイグレーション計画、アプリケーションに影響する可能性のある非互換性の識別 (*非互換性* とは、 DB2 ユニバーサル・データベースのうち、既存のアプリケーションで使用した場合に、結果が予期していないものになったり、アプリケーションを変更する必要が生じたり、パフォーマンスを低下させたりするなど、 DB2 の以前のリリースと異なる動作をする部分) および各国語サポート (NLS) の利用などについても説明しています。

管理の手引き: インプリメンテーション は、データベース設計の実装の詳細を扱っています。トピックには、データベースの作成と切り替え、データベース・セキュリティー、およびコントロール・センター (DB2 グラフィカル・ユーザー・インターフェース) による DB2 の管理が含まれます。

管理の手引き: パフォーマンス では、アプリケーションや DB2 自体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。

第2部 データベースの概念

第2章 リレーショナル・データベースの基本的な概念

このセクションでは、次のようなトピックを扱います。

- 『データベース・オブジェクトの概要』
- 12ページの『ストレージ・オブジェクトの概要』
- 18ページの『システム・オブジェクトの概要』
- 19ページの『データに関する業務規則』
- 23ページの『データベースのリカバリー』
- 24ページの『データベースでの表の再編成』
- 24ページの『DB2 のセキュリティーの概説』

データベース・オブジェクトの概要

このセクションでは、以下の重要なデータベース・オブジェクトの概要について説明します。

- インスタンス
- データベース
- ノードグループ
- 表
- 視点
- 索引
- スキーマ
- システム・カタログ表

8ページの図1 では、これらの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、長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。

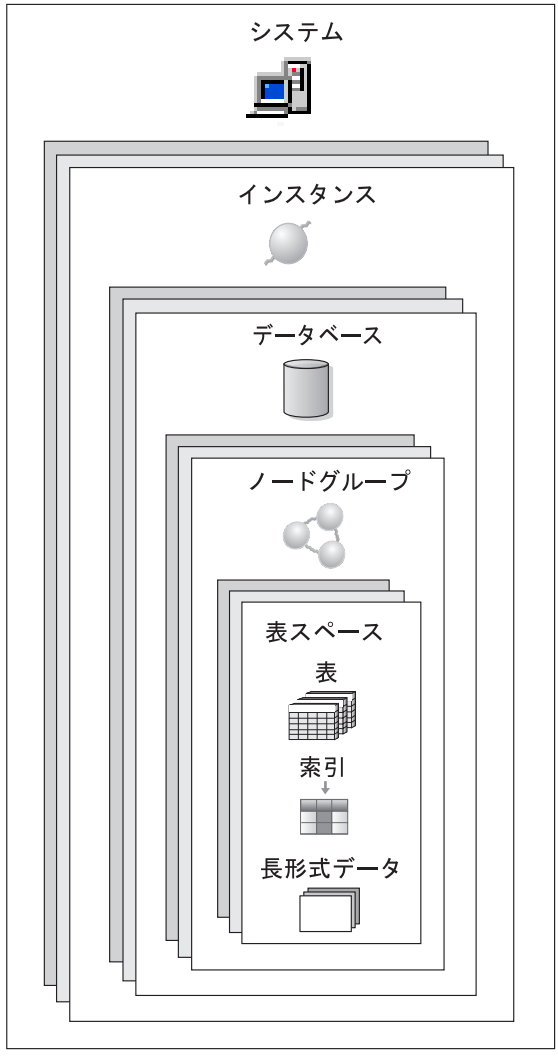

図 1. いくつかのデータベース・オブジェクトの関係

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区画が含まれます。(33ページの『第4章 並列データベース・システム』を参照。) 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区画は、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も持っています。

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、データの論理および物理構造を説明する一連のシステム・カタログ表、データベースに割り当てられているパラメーター値を含む構成ファイル、および進行中のトランザクションおよびアーカイブ可能トランザクションのリカバリー・ログが含まれます。

ノードグループ

ノードグループ は、1 つ以上のデータベース区画のセットです。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ノード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。ノードグループについての詳細は、33ページの『ノードグループおよびデータ区分化』を参照してください。データベース区画の定義については、33ページの『第4章 並列データベース・システム』を参照してください。表スペースについての詳細は、12ページの『表スペース』を参照してください。

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。表スペースについての詳細は、12ページの『表スペース』を参照してください。表の中のデータは論理的に関連付けられ、その関係は、複数の表の間で定義することができます。データは、*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言語、構造化照会言語 (SQL、*SQL 解説書* を参照) 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、ステートメントを作成するために、次の形式で SQL を使用します。

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

視点

視点 は、データを保守せずに表すための効率的な方法です。視点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。

視点には、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、視点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。

図2 は、表と視点の関連を示しています。

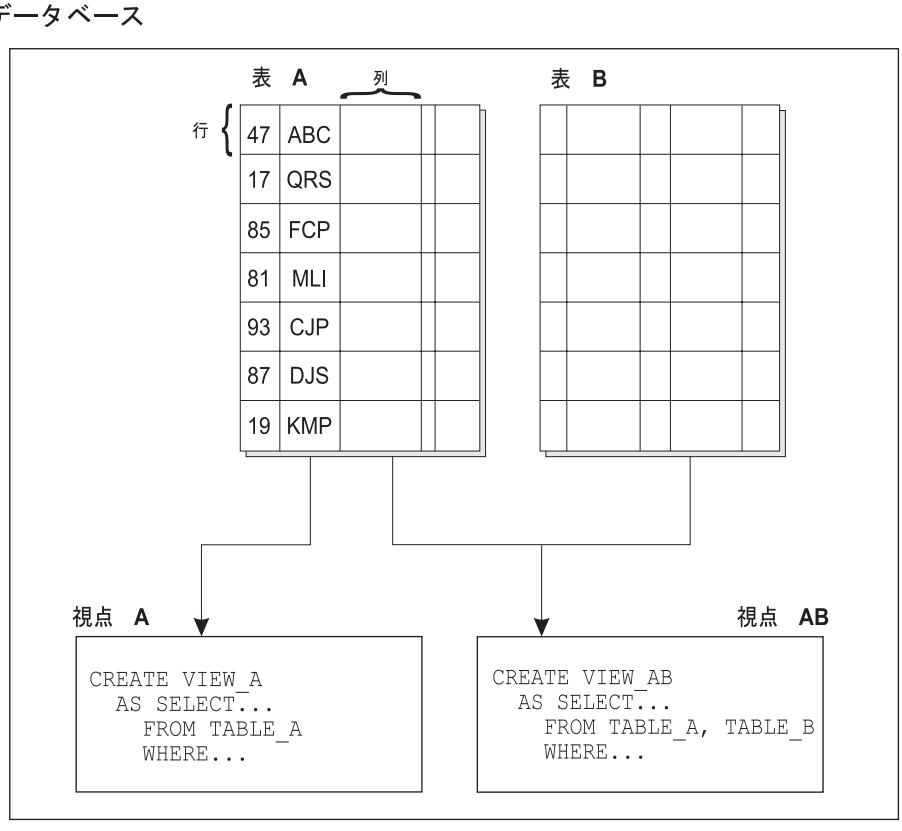

図 2. 表と視点の関係

索引

索引 は一式のキーであり、それぞれのキーは表の行を指しています。たとえば、11ページの図3 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 *最適化プログラム* 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。最適化プログラム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。

固有索引は、索引キーが必ず固有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。固有索引を使用すると、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固有のものとなります。 19ページの『データに関する業務規則』では、キーと索引の詳細が説明されています。

図3 は、索引と表の関係を示しています。

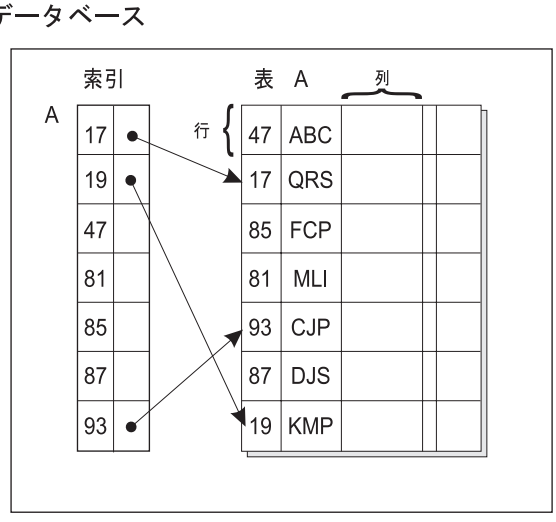

図 3. 索引と表の関係

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・オブジェクトを助ける、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。このオブジェクトは、スキーマ名によって修飾できるものであればなんでもかまいません。たとえば、表、索引、視点、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できます。

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を説明する一連の*システム・カタログ表* が含まれます。 DB2 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、視点、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除去したりすることはできませんが、カタログ視点を使用して内容の照会や表示を行うことは可能です。

ストレージ・オブジェクトの概要

以下のデータベース・オブジェクトにより、システムでデータが保管される方法、および (データへのアクセスと関連する) パフォーマンスを向上させる方法を定義できます。

- 表スペース
- コンテナー
- バッファー・プール

表スペース

データベースは、*表スペース* と呼ばれる部分に編成されています。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データを分散させるときの柔軟性を示しています。

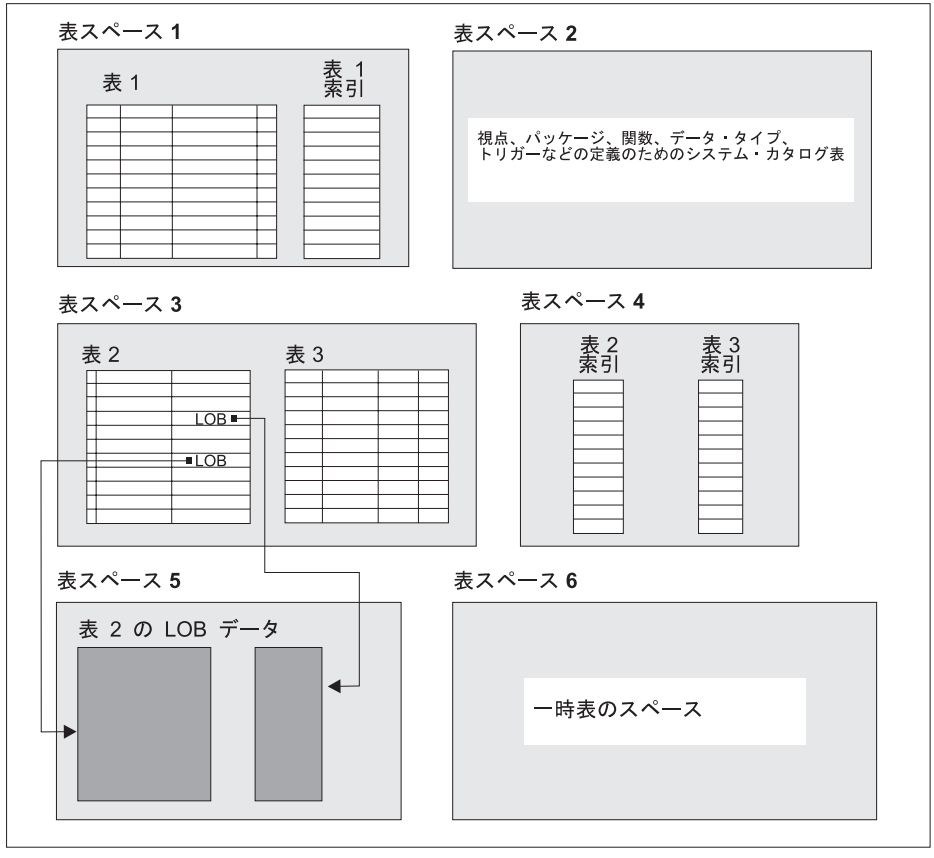

図 4. 表スペース

ノードグループの中に常駐する表スペース (9ページの『ノードグループ』を参照)。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 (12ページの『システム・カタログ表』を参照)。

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 (ファイルまたは装置など) 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管理プログラム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・管理プログラムがストレージを制御します。

図5 は、表、表スペース、および 2 つのタイプのスペース間の関係を示しています。この図はまた、表、索引、長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。

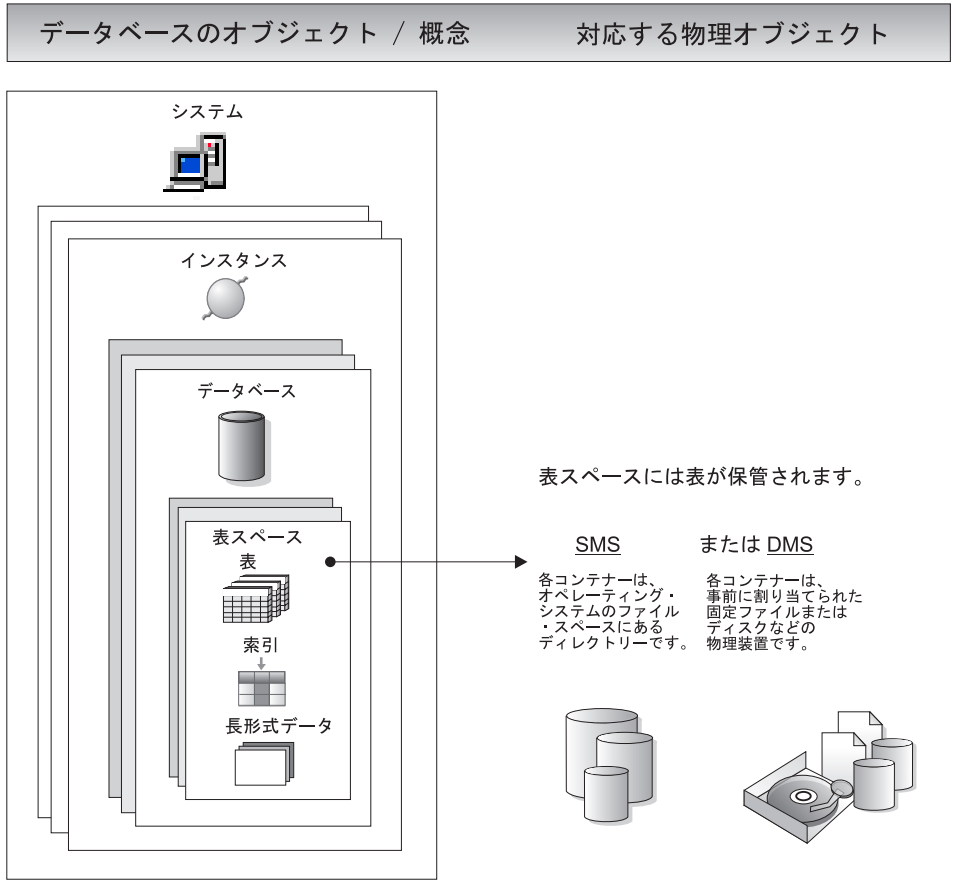

図 5. 表スペースと表

15ページの図6 は、3 つの表スペース・タイプを示しています。これらは*正規*、*一時*、および*長形式* です。

ユーザー・データを含む表は、正規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。索引も正規表スペースに保管されています。システム・カタログ表は、正規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。

長形式フィールド・データまたは長形式オブジェクト・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、長形式表スペースに存在します。

一時表スペース は、システム表またはユーザー表としてソートされます。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グローバル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、デフォルトにはデータベース作成の時点で*作成されません*。

図 6. 3 つの表スペース・タイプ

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、装置名、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。

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関係を示しています。

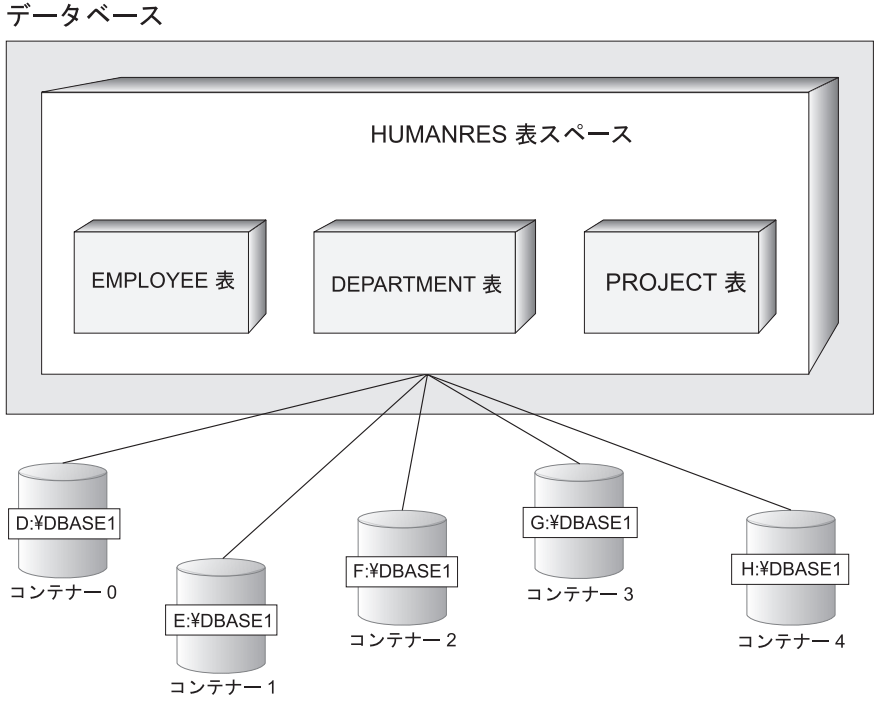

図 7. データベース内の表スペースと表

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、単一の最も重要な調整域になります。

図8 は、バッファー・プールとコンテナーの関係を示しています。

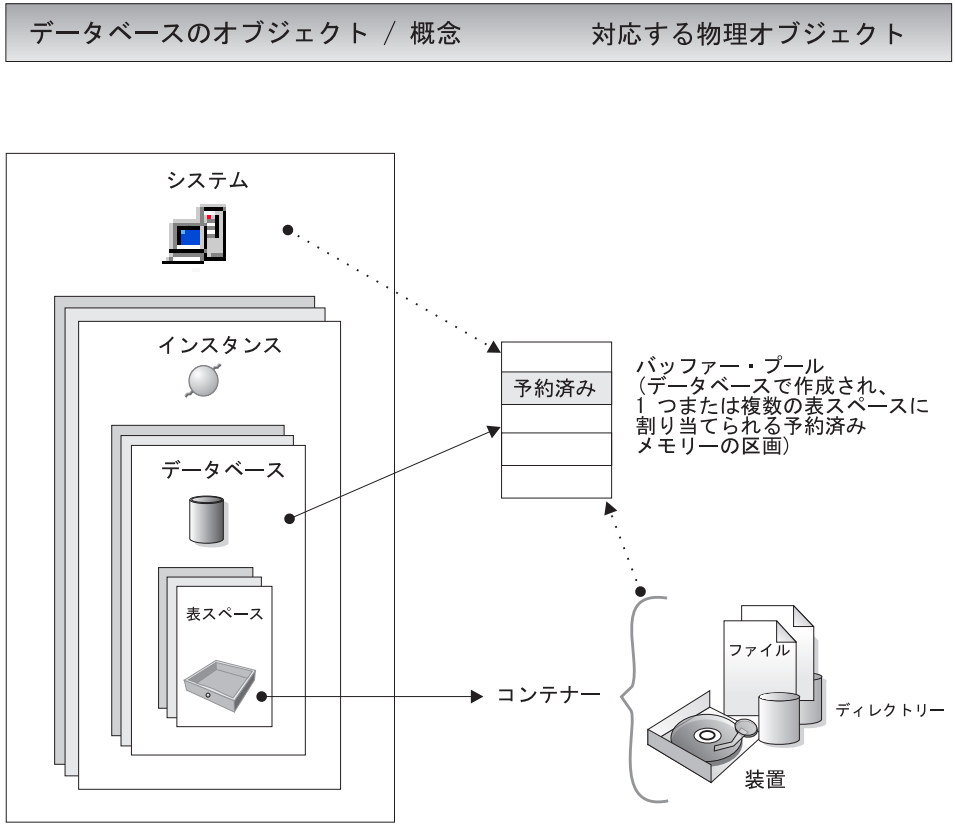

図 8. バッファー・プールとコンテナー

システム・オブジェクトの概要

DB2 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、パフォーマンスを向上させることができます。

構成パラメーター

構成ファイル には、 DB2 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 2 つのタイプがあります。 DB2 インスタンスごとに存在するデータベース・マネージャー構成ファイルと、データベースごとに存在するデータベース構成ファイルです (19ページの図9 を参照)。

データベース・マネージャー構成ファイル は、 DB2 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。

それぞれのクライアント・インストールごとにも、 1 つのデータベース・マネージャー構成ファイルがあります。このファイルには、特定のワークステーションのクライアント・イネーブラーに関する情報があります。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、データベースが常駐している場所に常駐します。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、異なった変更が必要になることがあります。

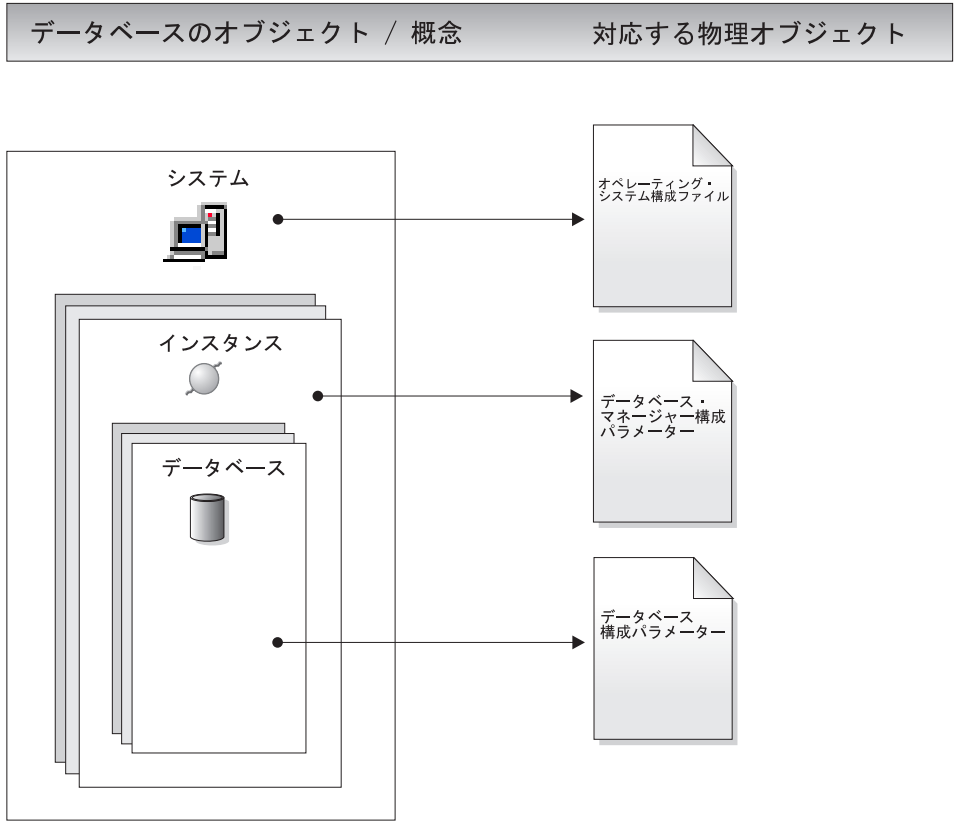

図 9. 構成パラメーター・ファイル

データに関する業務規則

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、従業員番号が固有でなければならない、などです。 DB2 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 は、以下のタイプの制約を提供します。

- NOT NULL 制約
- 固有制約
- 基本キー制約
- 外部キー制約
- 検査制約

NOT NULL 制約

NOT NULL 制約は、ヌル値が列に入力されないようにします。

固有制約

固有制約は、一連の列にある値が固有となり、表のすべての行が非ヌルであるようにします。たとえば、部署表での典型的な固有制約では、部署番号が固有かつ非ヌルでなければなりません。

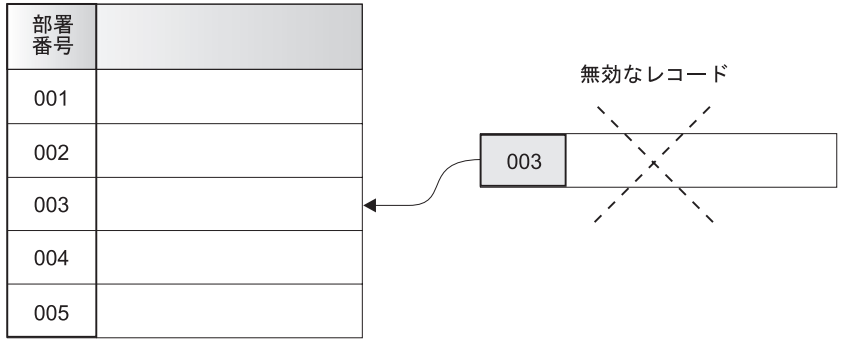

図 10. 固有制約は重複データを防ぐ

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。

基本キー制約

それぞれの表は、1 つの基本キーを持つことができます。基本キーとは、固有制約と同じ特性を持つ、列または列の組み合わせです。基本キー制約と外部キー制約を使用して、表間の関係を定義できます。

基本キーは表の行を識別するために使用するため、固有でなければならず、追加または削除は少なくなければなりません。 1 つの表は複数の基本キーを持つことはできませんが、複数の固有キーを持つことはできます。基本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益もあります。

次の表で、DEPTNO と EMPNO は、それぞれ DEPARTMENT 表と EMPLOYEE 表の基本キーです。

表 1. DEPARTMENT 表

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

表 2. EMPLOYEE 表

EMPNO (基本キー)	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関係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。

この関係を確立するには、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基本キーとして定義できます。

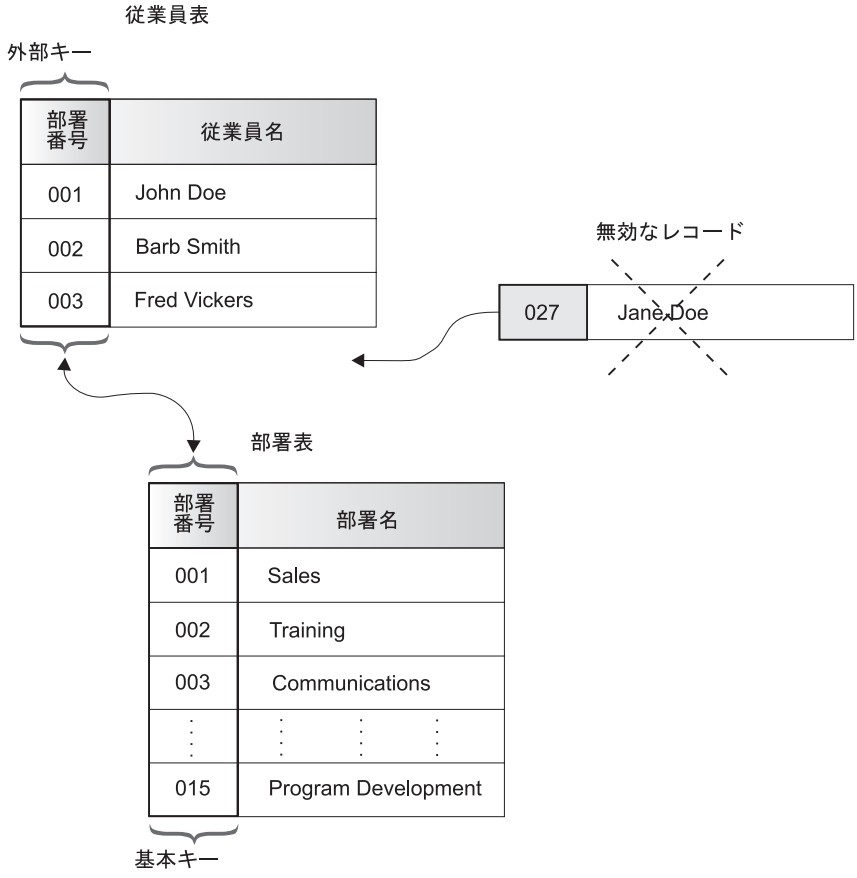

図 11. 外部キー制約および基本キー制約は関係を定義してデータを保護する

検査制約

検査制約は、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、拒否されることにより、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。

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さらに複雑であり、潜在的により強力です。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。このトリガーについての詳細は、*アプリケーション開発の手引き* を参照してください。

データベースのリカバリー

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、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。したがって、障害に対処できるように、データベースを保護するための方策を講じておく必要があります。

バックアップ、リカバリー、および良いバックアップ / リカバリー・ストラテジーの立て方に関する詳細は、 *データ回復と高可用性の手引きと解説書* でご覧になれます。

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。

データベースが損傷した場合や破壊された場合は、バックアップ・イメージからこれを再作成できます。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。

トランザクションの進行中にデータベースが破損した場合、この破損リカバリーというプロセスは、そのデータベースが再始動されたときに行われます。*破損リカバリー* のプロセスでは、データベースを整合された使用可能な状態に戻します。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。

データベースが損傷していたり破壊されていたりする場合は、バージョン・リカバリーとロールフォワード・リカバリーという 2 通りのリカバリー方法があります。

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。データベースのバックアップにより、バックアップをとった時点と同じ状態にデータベースを復元できます。ただし、バックアップ時点から障害発生時点までのすべての作業単位は失われます。

バックアップを作成した時点より後のデータベースを復元したいのであれば、*ロールフォワード・リカバリー* を使用しなければなりません。ロールフォワード・リカバリー方式を使用するためには、データベースのバックアップを作成していることと、ログをアーカイブしていることが必要です (これは、 *logretain* または *userexit* データベース構成パラメーターのいずれかあるいはその両方を使用可能にすることによって可能です)。

すべて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をリカバリーさせることが可能です。ログ・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。ログ・ファイルに直接変更を加えることはできません。

もう 1 つの大切なリカバリー・オブジェクトに、リカバリー・ヒストリー・ファイルがあります。リカバリー・ヒストリー・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要約が含まれています。これはバックアップ、復元、およびロード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。

注: バックアップとリカバリーに関するすべての情報、およびそれに対応するコマンド解説書や API 解説書内の情報は、*データ回復と高可用性の手引きと解説書* に集約されています。 *データ回復と高可用性の手引きと解説書* は、バックアップとリカバリーに関する主要かつ唯一の情報源として提供されています。

データベースでの表の再編成

何度も更新を行うと表はフラグメント化され、パフォーマンスが悪化します。統計を収集したときにパフォーマンスの向上が見られない場合、表を再編成すると役立つことがあります。表データを再編成するとは、データを指定した索引にしたがって物理シーケンスに再配列して、フラグメント化されたデータに伴うフリー・スペースを除去することです。これにより、より速くデータにアクセスできるため、パフォーマンスが向上します。

表を再編成する前に、 REORGCHK コマンドを呼び出して、その表の統計を収集することをお勧めします。このコマンドを実行することは、表データの再編成が適切かどうかを判別するのに役立ちます。 REORGCHK コマンドの詳細については、*コマンド解説書* を参照してください。

DB2 のセキュリティーの概説

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 は、外部セキュリティー・サービスと内部アクセス制御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。

認証

ユーザーの認証は、DB2 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。DCE セキュリティー・サービスは、分散環境に対してセキュリティー機能を提供する別個の製品です。 Windows 95 または Windows 3.1 オペレーティング・システムには、セキュリティー機能はありません。

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 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 *authid* は、 DB2 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。 DCE セキュリティー・サービス製品では、DB2 *authid* は、 DCE レジストリーの中に入っており、認証が正常に完了するとそこから取り出されます。
- 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 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 は、最大 64 グループまでのグループのリストを取得します。 1 ユーザーが 64 グループを超えるグループのメンバーである場合、有効な DB2 許可名にマップされる最初の 64 グループだけが、DB2 グループ・リストに追加されます。エラーは戻されず、最初の 64 グループより後のグループは、すべて DB2 によって無視されます。

DB2 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 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 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 管理者は、プロファイル・レジストリー変数 DB2CHGPWD_EEE を使用して、AIX および Windows NT EEE システム上で、パスワードを変更する許可を他のユーザーに与えることができます。この変数のデフォルト値は NOT SET (使用不可) です。DB2CHGPWD_EEEは、他の DB2 プロファイル変数が使用する標準ブール値を受け入れます。

DB2 管理者は、すべてのノードのパスワードを、 Windows NT ドメイン・コントローラ (Windows NT の場合) または NIS (AIX の場合) を使って集中保守する責任があります。

注: パスワードが集中保守されない場合、 DB2CHGPWD_EEE 変数を使用可能にすることによって、パスワードがすべてのノードで一貫しなくなる可能性が生じます。つまり、ユーザーが「パスワード変更」機能を使用すると、ユーザーのパスワードが、接続しているノードでのみ変更されます。

DB2 UDB (AIX 版) 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。

データベースにアクセスしているリモート・クライアントがある場合に特に関係する、システム項目の妥当性検査についての追加情報に関しては、 *管理の手引き: インプリメンテーション* の『サーバーに対する認証方式の選択』を参照してください。

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 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、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 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。

詳細については、 *管理の手引き: インプリメンテーション* の『特権、権限、および許可』を参照してください。

連合データベース認証および許可の概説

DB2 連合データベース・システム (federated database system: 複数のデータベースから構成されるが、単一のデータベース・イメージを提供するデータベース・システムを意味します) は複数のデータベース管理システムの情報にアクセスできるので、データを保護するために付加的なステップが必要になる場合があります。

認証アプローチの計画時には、ユーザーが DB2 だけではなくデータ・ソースでも、認証検査をパスする必要があるかもしれないということを考慮してください。連合システムでは、DB2 クライアント・ワークステーション、DB2 サーバー、データ・ソース (DB2、DB2 (OS/390 版)、他の DRDA サーバー、Oracle)、または DB2 (クライアントまたは DB2 サーバー) とデータ・ソースの組み合わせにおいて、認証が行われる可能性があります。 DCE 環境でも、データ・ソースにユーザー ID とパスワードが必要な場合、特定のステップが必要になります。詳細については、 *管理の手引き: インプリメンテーション* の『連合データベースの認証処理』を参照してください。

同様に、ユーザーは、データ・ソースおよび DB2 で許可検査をパスする必要があります。各データ・ソース (DB2、Oracle、DB2 (OS/390 版) など) は、その制御下でオブジェクトのセキュリティーを保守します。ユーザーがニックネームに対して操作を実行する場合、そのユーザーはニックネームが参照する表または視点に関する許可検査をパスする必要があります。

第3章 連合システム

連合データベース・システム または *連合システム* とは、アプリケーションまたはユーザーが、1 つのステートメントで 2 つ以上の DBMS またはデータベースを参照する SQL ステートメントを実行依頼することをサポートするデータベース管理システム (DBMS) のことです。一例として、2 つの異なる DB2 データベースにある表を結合することがあります。このタイプのステートメントを、*分散要求* といいます。

DB2 ユニバーサル・データベースの連合システムでは、データベースおよび DBMS 間の分散要求をサポートしています。たとえば、DB2 表と Oracle 視点との間で UNION 操作を実行できます。サポートされている DBMS には、DB2、DB2 ファミリーのメンバー (DB2 (OS/390 版) や DB2 (AS/400 版)) そして Oracle があります。

DB2 連合システムは、データベース・オブジェクトの*位置透過性* を提供します。 (表および視点についての) 情報が移動する場合に、その情報への参照 (ニックネーム と呼ばれる) を、その情報を要求するアプリケーションに変更を加えることなく更新することができます。さらに、DB2 連合システムには、すべての DB2 SQL ダイアレクトをサポートしているわけではない、あるいは特定の最適化能力をサポートしていない DBMS 向けの*代償機能* も備えられています。特定の DBMS で実行できない操作 (再帰的 SQL など) は、DB2 で実行されます。

DB2 連合システムは*半自律的な* 方法で機能します。 Oracle オブジェクトへの参照を含む DB2 照会を実行依頼しながら、 Oracle アプリケーションで同じサーバーにアクセスすることができます。 DB2 連合システムは、(整合制約 / ロック制約を除き) Oracle や他の DBMS オブジェクトへのアクセスを占有したり制約することはありません。

DB2 連合システムは、DB2 UDB のインスタンス、*連合データベース* の役割のデータベース、そして 1 つ以上のデータ・ソース で構成されています。連合データベースには、データ・ソースとその特性を識別するためのカタログ項目が含まれています。データ・ソースは、DBMS とデータで構成されています。アプリケーションは、他の DB2 データベースと同じように、連合データベースに接続します。 30ページの図12 を見れば、連合データベース環境を視覚的に理解できます。

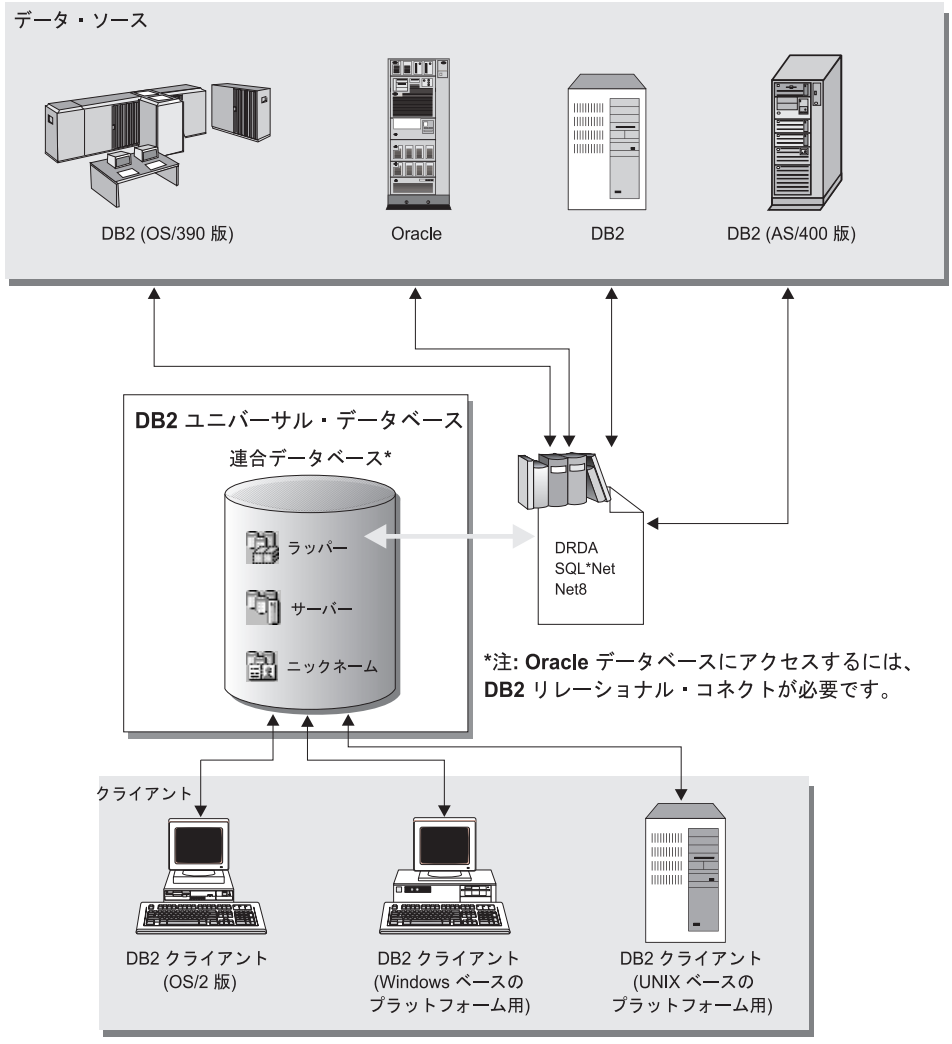

図 12. 連合データベース・システム

DB2 連合データベース・カタログ項目には、データ・ソース・オブジェクトについての情報 (その名前、含まれている情報、使用するときの条件) が載せられています。この DB2 カタログには、いろいろな DBMS に含まれているオブジェクトについての情報が格納されているので、*グローバル・カタログ* と呼ばれます。オブジェクト属性についても、そのカタログに格納されます。参照される実際の DBMS、データ・ソースとのやり取りで使用するモジュール、そしてアクセスされる DBMS データ・オブジェクト (表など) は、データベースの外側にあります。 (1 つの例外として、連合データベースは連合システムのデータ・ソースとすることができます。) 連合オブジェクトは、コントロール・センターまたは SQL DDL ステートメントを使用して作成できます。必要な連合データベース・オブジェクトは以下のとおりです。

ラッパー

データ・ソースの特定のクラスまたは区分にアクセスする際に使用するモジュール (DLL、ライブラリーなど) を識別します。

サーバー

データ・ソースを定義します。サーバー・データには、ラッパー名、サーバー名、サーバー・タイプ、サーバー・バージョン、許可情報、およびサーバー・オプションが含まれています。

ニックネーム

特定のデータ・ソース・オブジェクト (表、別名、視点) を参照する連合データベースに格納されている ID。アプリケーションは照会を使用して、表や視点を参照するときのように、ニックネームを参照します。

それぞれの必要に応じ、以下のオブジェクトを別に作成することができます。

- ユーザー・マッピング (認証問題を扱う)
- データ・タイプ・マッピング (データ・ソースのタイプと DB2 タイプとの間の関連をカスタマイズする)
- 機能マッピング (ローカル機能をデータ・ソース機能へマップする)
- 索引仕様 (パフォーマンスを上げる)

連合システムを設定したら、1 つの大きなデータベースにある情報のようにして、データ・ソース内の情報にアクセスできます。ユーザーとアプリケーションはある連合データベースへ照会を送り、必要に応じて DB2 ファミリーや Oracle システムからデータを取り出します。ユーザーとアプリケーションは照会の中でニックネームを指定します。このようなニックネームにより、データ・ソースにある表や視点を参照することになります。エンド・ユーザー側から見ると、ニックネームは別名によく似ています。

連合システムのパフォーマンスに影響する要因は数多くあります。データ・ソースとそのオブジェクトについての最新情報が、連合データベースのグローバル・カタログに格納されていることを確認することは、最重要な作業です。この情報は DB2 最適化プログラムによって使用されるものであり、データ・ソースでの評価のための操作をプッシュダウンするときの判断に影響する可能性があります。連合システムのパフォーマンスについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

DB2 連合システムの操作には、いくらかの制限があります。分散要求は読み取り専用操作に限定されています。さらに、ニックネームに対するユーティリティー操作 (LOAD、REORG、 REORGCHK、IMPORT、RUNSTATS など) を実行できません。

しかし、パススルー機能を利用すれば、該当のデータ・ソースと関連した SQL ダイアレクトを使用し、データベース・マネージャーに対して直接に DDL および DML ステートメントを実行依頼できます。

連合システムは、並列環境に対応できます。連合データベース照会を意味的にローカル・オブジェクト (表、視点) 参照とニックネーム参照とに細分化できる程度に応じて、それぞれのパフォーマンス向上の度合いが決まります。ニックネーム・データの要求は順番に処理されますが、ローカル・オブジェクトは並列して処理することができます。たとえば、`SELECT * FROM A, B, C, D` という照会があり、 A と B はローカル表を表し、 C と D は Oracle データ・ソースにある表を参照するニックネームであるとします。この場合、1 つの可能なプランは、表 A と B を並列結合で結合することです。そのようにすると、その結果はニックネーム C と D へ順番に結合されます。

連合システムを使用可能にする

DB2 エンタープライズ・エディション (EE) および DB2 エンタープライズ拡張エディション (EEE) は、連合データベースをサポートしています。連合システムを使用可能にするには、次のようにします。

1. DB2 EE または EEE のインストール時に、*「DB2 データベースの分散結合 (Distributed Join for DB2 Databases)」* インストール・オプションを選択します。
2. 連合システムに Oracle データベースを含める場合は、 DB2 リレーショナル・コネクトをインストールします。詳細については、*インストールおよび構成 補足* を参照してください。
3. データベース・マネージャーの構成パラメーター *federated* を「YES」に設定します。
4. ラッパー、サーバー、およびニックネームを作成します (詳細は、 *管理の手引き: インプリメンテーション* の『データベースの作成』を参照)。
5. 別のオブジェクトを作成するか、必要に応じてオプションを設定します (詳細は、*管理の手引き: インプリメンテーション* の『設計のインプリメント』を参照)。

第4章 並列データベース・システム

DB2 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。*データベース区画* 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。データベース区画は、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。 (ノードは、DB2 パラレル・エディション (AIX 版) バージョン 1 の製品で使用された用語です。)

単一区画データベース は、 1 つだけのデータベース区画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、その区画に保管されます。この場合、ノードグループが提供されても (9ページの『ノードグループ』を参照)、追加の機能は提供しません。

区分データベース は、 2 つ以上のデータベース区画を持つデータベースです。表は、1 つ以上のデータベース区画に配置することができます。表が複数区画からなるノードグループ内にある場合、その行のうちの一部が 1 つの区画に保管され、その他の行は他の区画に保管されます。

通常、単一のデータベース区画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区画のデータベース・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区画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副要求に分解され、適用可能なデータベース区画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区画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。

ユーザー対話は、そのユーザーに対する*調整プログラム・ノード*として知られているデータベース区画を介して生じます。調整プログラムは、アプリケーションと同じデータベース区画で実行されるか、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区画で実行されます。任意のデータベース区画を調整プログラム・ノードとして使用することができます。

ノードグループおよびデータ区分化

1 つのデータベースの中に、 1 つ以上のデータベース区画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*ノードグループ* と呼びます。 1 つ以上のデータベース区画が含まれる各サブセットを*複数区画ノードグループ* と呼びます。複数区画ノードグループは、同じインスタンスに属するデータベース区画でのみ定義することができます。

図13 は、5 つの区画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。

- 1 つのノードグループは、1 つのデータベース区画を除き、すべてにまたがっています (ノードグループ 1)。
- 1 つのノードグループには 1 つのデータベース区画が含まれます (ノードグループ 2)。
- 1 つのノードグループには 2 つのデータベース区画が含まれます。(ノードグループ 3)。
- ノードグループ 2 に含まれているデータベース区画は、ノードグループ 1 によって共有 (およびオーバーラップ) されます。
- ノードグループ 3 には 1 つのデータベース区画がありますが、これはノードグループ 1 によって共有 (およびオーバーラップ) されています。

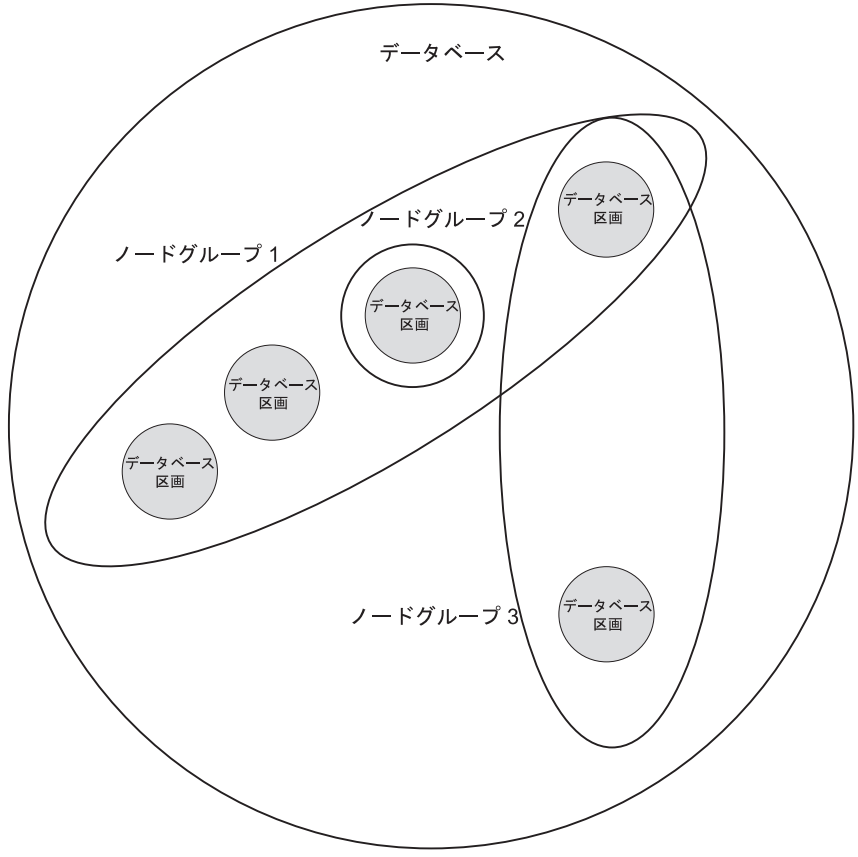

図 13. データベース内のノードグループ

新しいノードグループは、CREATE NODEGROUP ステートメントを使用して作成します。詳しくは、*SQL 解説書* を参照してください。データは、ノードグループ内のすべての区画に渡って分割されます。複数区画ノードグループを使用している場合、ノード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。詳細については、98ページの『ノードグループの設計』を参照してください。

並列化のタイプ

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 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。このセクションでは、 DB2 によってサポートされている次のタイプの並列性について説明します。

- 入出力
- 照会
- ユーティリティー

入出力並列化

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、スループットはかなり向上します。

入出力並列化は、39ページの『ハードウェア環境』で説明する各ハードウェア環境のコンポーネントです。 47ページの表3 は、入出力並列化に最適なハードウェア環境の一覧を示したものです。

照会並列化

照会並列化のタイプには、照会間並列化と照会内並列化の 2 つがあります。

照会間並列化 とは、 1 つのデータベースを同時に複数のアプリケーションで照会する機能のことです。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 は、それらのすべてを同時に実行します。 DB2 は、このタイプの並列化を常にサポートします。

照会内並列化 とは、*区画内並列化* または*区画間並列化* (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。

本書全体を通して、*照会並列化* という用語を使用します。

区画内並列化

区画内並列化 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。 (一部のユーティリティーも、このタイプの並列化を実行します。 38ページの『ユーティリティー並列化』を参照してください。)

区画内並列化では、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区画内で 並列して実行できるようにします。

図14 は、4 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。区画内並列化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。並列化の程度をユーザーが選択するか、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。並列化の程度は、並列に実行する照会のピースの数を表しています。

47ページの表3 は、区画内並列化に最適なハードウェア環境の一覧を示したものです。

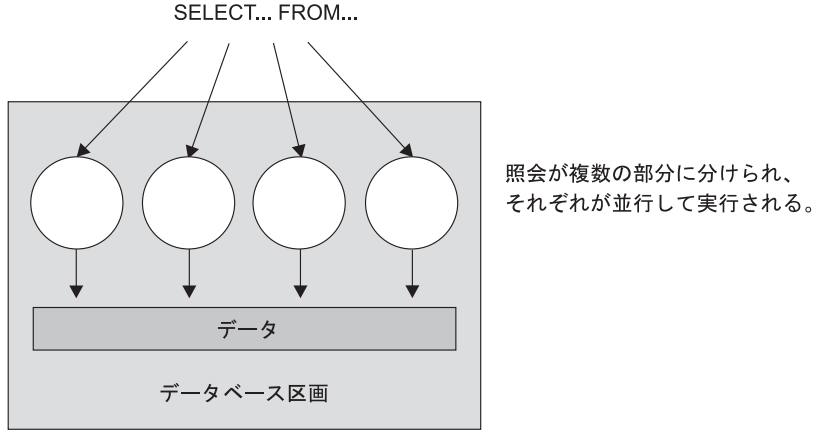

図 14. 区画内並列化

区画間並列化

区画間並列化 とは、1 つのマシンまたは複数のマシン上で、区分データベースの複数の区画に渡って 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。 (一部のユーティリティーも、このタイプの並列化を実行します。 38ページの『ユーティリティー並列化』を参照してください。)

区画間並列化では、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、区分データベースの複数の区画に渡って 並列して実行できるようにします。

図15 は、4 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、照会が単一の区画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。

並列化の程度は、作成した区画の数とノードグループを定義した方法によって大部分決まります。

47ページの表3 は、区画間並列化に最適なハードウェア環境の一覧を示したものです。

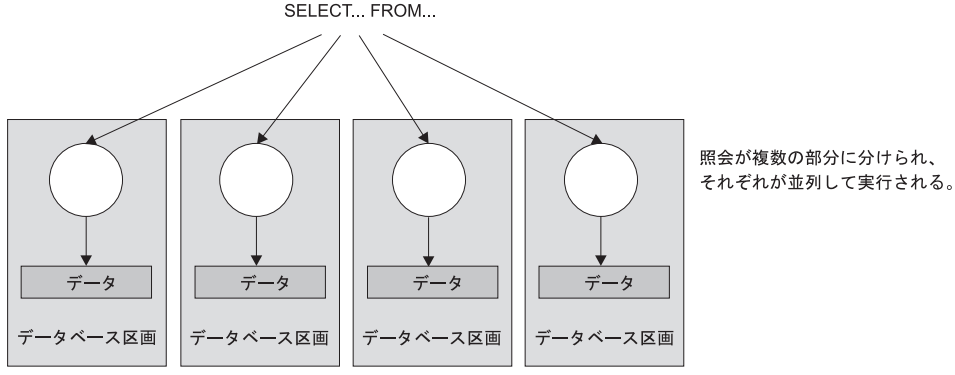

図 15. 区画間並列化

区画内並列化と区画間並列化の同時使用

区画内並列化と区画間並列化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 次元並列化が可能になるため、この結果、照会の処理スピードが劇的に速くなります。

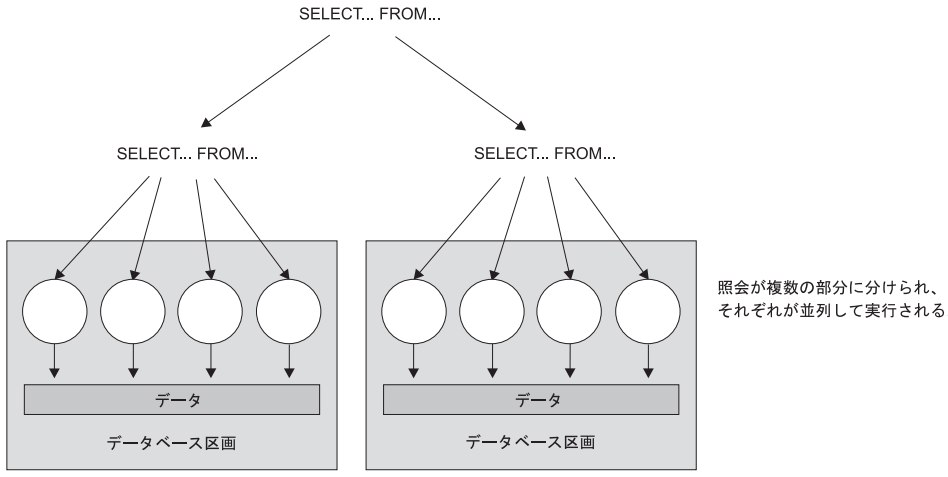

図 16. 区画内並列化と区画間並列化の同時使用

ユーティリティー並列化

DB2 のユーティリティーは、区画内並列化を利用することができます。これらのユーティリティーは区画間並列化も利用しており、複数のデータベース区画が存在していると、ユーティリティーはそれぞれの区画で並列に実行されます。

ロード・ユーティリティーは、区画内並列化と入出力並列化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して書き込むことができます。ロード・ユーティリティーの並列化を可能にする方法については、 *データ移動ユーティリティー手引きおよび解説書* を参照してください。

区分データベース環境では、オートローダー・ユーティリティーは、表が存在する各データベース区画で LOAD コマンドを並列的に呼び出すことによって、区画内、区画間、および入出力並列処理を利用します。オートローダー・ユーティリティーについては、 *データ移動ユーティリティー手引きおよび解説書* を参照してください。

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 では、索引を作成するときに、入出力並列化と区画内並列化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復元は、かなり入出力制約型のタスクです。 DB2 では、バックアップ操作と復元操作を実行するときに、入出力並列化と区画内並列化の両方を利用

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、入出力並列化を利用しています。 BACKUP DATABASE コマンドと RESTORE DATABASE コマンドで並列化を可能にする方法については、 *コマンド解説書* でこれらのユーティリティーを参照してください。

ハードウェア環境

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) 上での単一区画
- 複数プロセッサーを備えた単一区画 (SMP)
- 複数区画構成
  - 1 つのプロセッサーを備えた区画 (MPP)
  - 複数のプロセッサーを備えた区画 (SMP のクラスター)
  - 論理データベース区画 (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されます。*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の単一区画

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17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。

47ページの表3 に、このハードウェア構成を利用するための最適な並列化のタイプの一覧を示します。

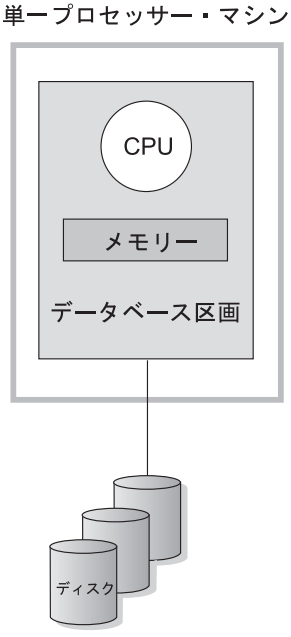

図 17. 単一プロセッサー上の単一区画

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。作業負荷が増加すると、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区画システムにマイグレーションすることを検討することができます。

複数プロセッサーを備えた単一区画

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18 を参照)、*対称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、*共用* 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 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復元、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。

47ページの表3 に、このハードウェア構成を利用するための最適な並列化のタイプの一覧を示します。

SMP マシン

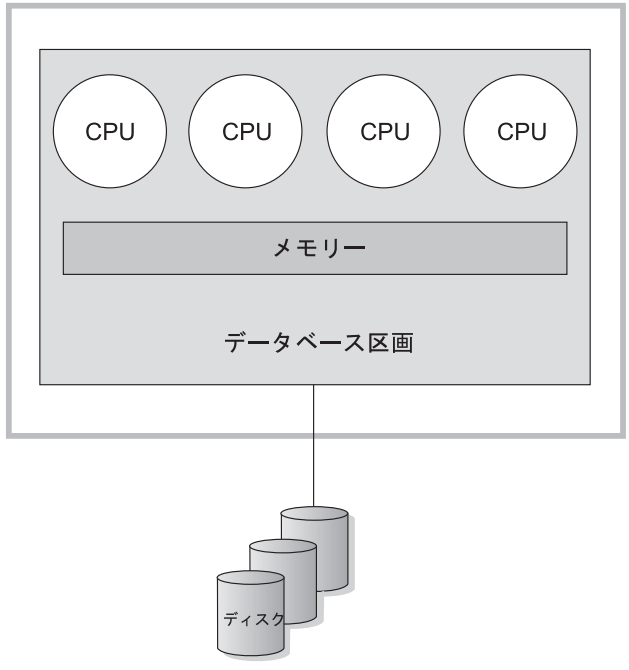

図 18. 単一区画データベースの対称マルチプロセッサー・システム

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。共用メモリーと共用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用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区画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、複数区画を備えたシステムにマイグレーションすることを検討することができます。

複数区画構成

1 つのデータベースを複数の区画に分割して、それぞれの区画が独自のマシン上にあるようにすることができます。複数データベース区画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、以下の区画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区画
- 複数のプロセッサーを備えたシステムの区画
- 論理データベース区画

1 つのプロセッサーを備えた区画

この環境では、多くのデータベース区画があります。それぞれの区画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 (43ページの図19)。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、最大印刷位置 (MPP) 環境、および共用なし構成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 環境には共用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用によって発生する制約を除去します。

区分データベース環境によって、データベースは、物理的には複数の区画に分割されていても、 1 つの完全な論理的なものとして扱えます。データが区分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。各区画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。

47ページの表3 に、このハードウェア構成を利用するための最適な並列化のタイプの一覧を示します。

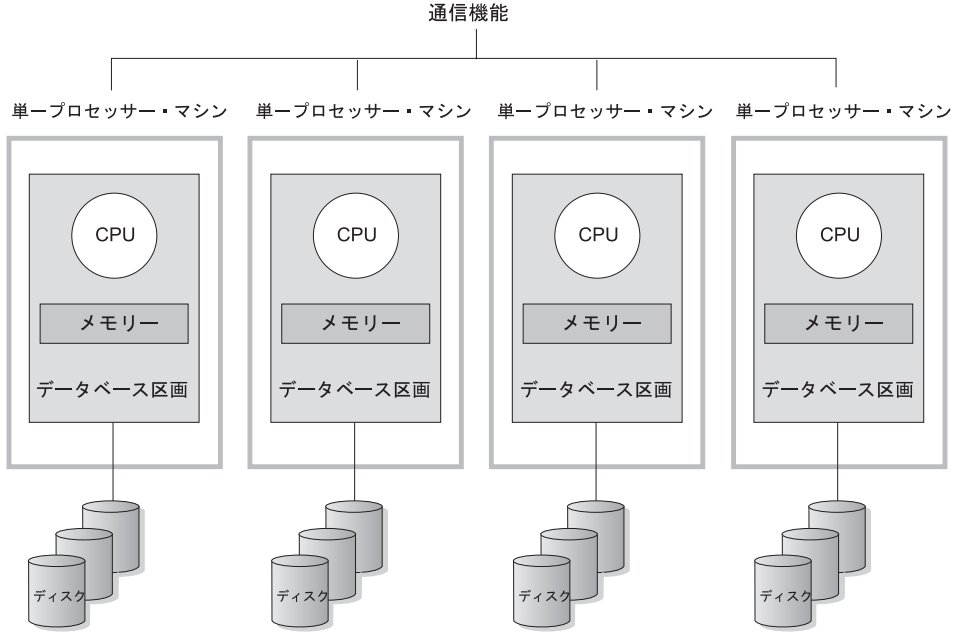

図 19. 大量並列処理システム

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区画 (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、各区画が複数のプロセッサーを備えたシステムにマイグレーションすることを検討することができます。

複数プロセッサーを備えた区画

各区画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 つの区画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
(44ページの図20)。

この構成は、SMP 並列化と MPP 並列化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区画で実行できることを意味します。また、1 つの照会を複数区画にわたって並列に実行できることも意味します。

47ページの表3 に、このハードウェア構成を利用するための最適な並列化のタイプの一覧を示します。

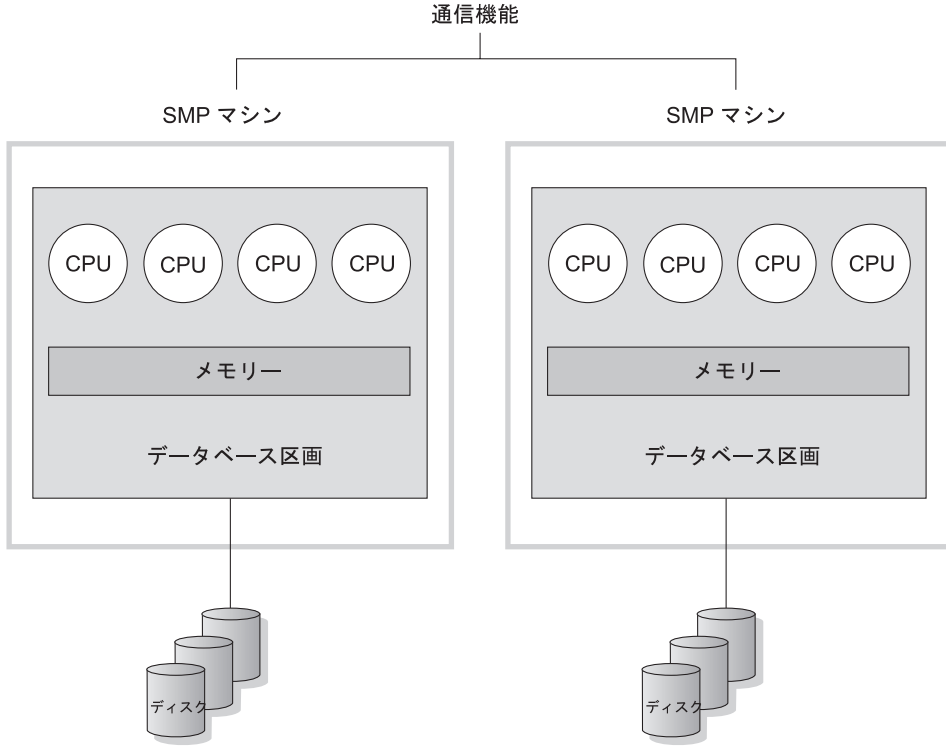

図 20. SMP のクラスター

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区画を追加したり、既存のデータベース区画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。

論理データベース区画

論理データベース区画は、マシン全体の制御権が与えられていないところが物理区画と異なります。マシンは共用リソースを持っていますが、データベース区画はリソースを共用しません。プロセッサーは共用されますが、ディスクとメモリーは共用されません。

論理データベース区画は拡張容易性を提供します。複数の論理区画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・マネージャーが可能なよりも完全に、使用可能なリソースを利用することができる場合があります。 45ページの図21は、特に多くのプロセッサーを備えたマシンについて、さらに区画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。データベースを区分化することによって、それぞれの区画を個別に管理およびリカバリーすることができます。

大規模な **SMP** マシン

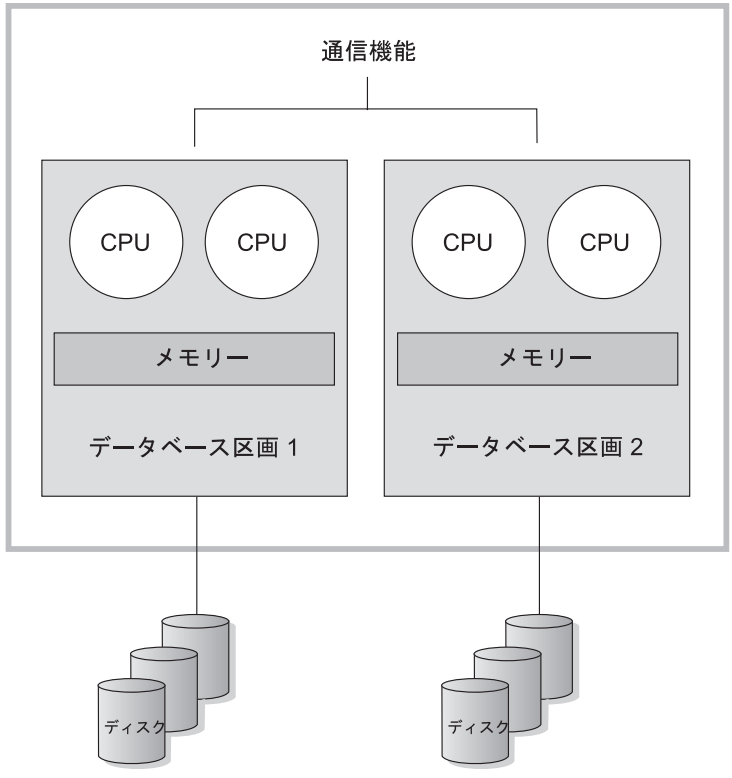

図 21. 区分データベース、対称マルチプロセッサー・システム

46ページの図22 は、処理能力を高めるために、 図21 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。

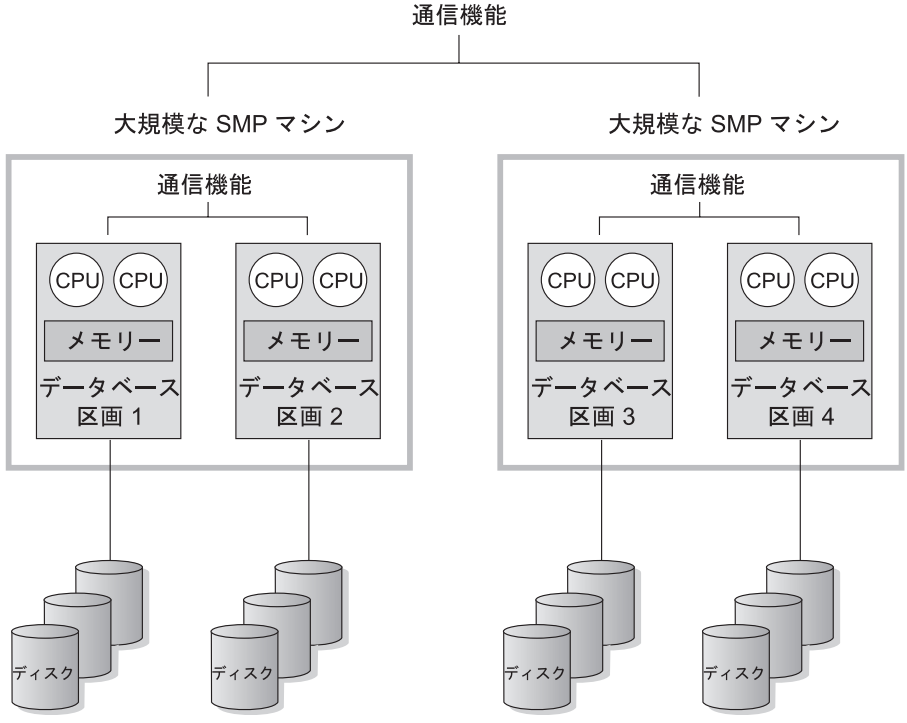

図 22. 区分データベース、一緒にクラスター化された対称マルチプロセッサー・システム

47ページの表3 に、このハードウェア環境を利用するための最適な並列化のタイプの一覧を示します。

注: また、2 つ以上の区画を、(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、高可用性構成とダウン対策を設計する上でより大きい柔軟性が得られます。機械故障の際に、同じデータベースの別の区画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区画を移動して再始動できます。詳細については、179ページの『第11章 高可用性とフェールオーバー・サポートの紹介』を参照してください。

それぞれのハードウェア環境ごとの最適な並列化についてのまとめ

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化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化のタイプ

ハードウェア環境	入出力並列化	照会内並列化	
		区画内並列化	区画間並列化
単一区画、シングル・プロセッサー	可	不可(1)	不可
単一区画、複数プロセッサー (SMP)	可	可	不可
複数区画、1 プロセッサー (MPP)	可	不可(1)	可
複数区画、複数プロセッサー (SMP のクラスター)	可	可	可
論理データベース区画	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)、シングル・プロセッサー・システムの場合でも、並列化の程度を 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。			

第5章 データウェアハウジングについて

DB2 ユニバーサル・データベースは、データウェアハウス処理を自動化するコンポーネントであるデータウェアハウスセンターを提供します。データウェアハウスセンターを使用すれば、ウェアハウスを含めるデータを定義できます。次に、データウェアハウスセンターを使用して、ウェアハウスにあるデータのリフレッシュを自動的にスケジュールできます。

このセクションでは、データウェアハウジングおよびデータウェアハウジング・タスクについて概説します。ウェアハウスの詳細、およびデータウェアハウスセンターについては、 *データウェアハウスセンター 管理の手引き* とデータウェアハウスセンターのオンライン・ヘルプを参照してください。

データウェアハウジングとは

操作可能データ (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、分析者が操作可能データに直接アクセスするには、いくつかの問題があります。

- 操作可能データベースを照会するための専門技術がない可能性がある。たとえば、IMS データベースへの照会には、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、操作可能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全時間業務に携わっています。
- 銀行のデータベースなどの多くの操作可能データベースでは、パフォーマンスが重要である。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。
- 一般に操作可能データは、業務分析者が使用するための最善のフォーマットになっていない。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、生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* (操作可能データから抜き出され、エンド・ユーザーの意思決定用に変換されたデータ) のストアを作成できます。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操作可能データベースからコピーして、データの要約を計算し、操作可能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。エンド・ユーザーは、操作可能データベースに影響を与えることなくこのデータベース (ウェアハウス) に照会できます。

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 (サブジェクト・エリア、ウェアハウス・ソース、ウェアハウス・ターゲット、エージェント、エージェント・サイト、ステップ、およびプロセス) 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、グループ分けします。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・エリアを定義できます。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。

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を識別します。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、データのアクセスと選択を行います。ソースには、ウェアハウス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) 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、エンド・ユーザーが使用できるように変換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。ウェアハウス・ソースのように、ウェアハウス・ターゲットもデータをデータウェアハウスセンター・ステップに提供できます。

ウェアハウス・エージェントおよびエージェント・サイト

データウェアハウスセンターのエージェント は、データ・ソースとターゲット・ウェアハウスの間のデータの流れを管理します。エージェントは、Windows NT、AIX、OS/2、OS/390、OS/400、および SUN Solaris オペレーティング・システムで利用できます。エージェントはオープン・データベース・コネクティビティー (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、ウェアハウスで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・サーバーと同じマシンにイン

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マシンにインストールされているエージェントです。

エージェント・サイト 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・サイト名は、TCP/IP ホスト名と同じではありません。単一の物理マシンが持つことのできる TCP/IP ホスト名は 1 つだけです。しかし、エージェント・サイトは単一のマシンで複数定義することができます。各エージェント・サイトは論理名によって識別されます。

Default VW AgentSite という名前のデフォルト・エージェント・サイト は、ウェアハウス制御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、 Windows NT 上のローカル・エージェントです。

ステップおよびプロセス

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。

- 出力表またはファイルの構造。
- 出力表またはファイルにデータを入れるための機構 (SQL またはプログラム)。
- 出力表またはファイルにデータを入れるスケジュール。

ステップは、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して、データの移動や変換を行います。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・ターゲット間でのデータの転送と、データの変換が行われます。

プロセス には、変換および移動タスクを実行する一連のステップが含まれます。一般に、プロセスは、 1 つまたは複数のウェアハウス・ソース (データベース表またはファイル) からデータを抜き出すことにより、ウェアハウス・データベースにあるウェアハウス・ターゲットにデータを入れます。ただし、ウェアハウス・ソースまたはウェアハウス・ターゲットを指定せずにプログラムを立ち上げるプロセスを定義することもできます。

必要に応じてステップを実行するか、または設定時刻にステップを実行するようにスケジュールすることができます。 1 回だけステップを実行するようにスケジュールするか、または毎週金曜日などのように繰り返し実行するようにスケジュールすることができます。また、ステップを順番に実行するようにスケジュールし、 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ようにすることもできます。ステップは、他のステップの完了時 (成功時または失敗時) に実行されるようにスケジュールできます。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。

ステップまたはプロセスが実行されると、次の方法でデータが保管されます。

- ウェアハウス・ターゲットにあるすべてのデータが新しいデータに置き換えられる。

- 新しいデータが既存のデータに付加される。
- 別の版のデータが付加される。

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. さまざまな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にある表に書き込む。

個々のステップを含むプロセスを作成することができます。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。次に、別のステップを使用して、変換されたデータを含むターゲット表にデータを入れることができます。

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。ステップについての詳細は、データウェアハウスセンター *管理の手引き* を参照してください。

SQL ステップ

SQL ステップは、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。

プログラム・ステップ

プログラム・ステップには、複数のタイプがあります。 DB2 (AS/400 版) プログラム、DB2 (OS/390 版) プログラム、 DB2 UDB プログラム、Visual Warehouse 5.2 DB2 プログラム、 OLAP サーバー・プログラム、ファイル・プログラム、および複製です。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。

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、データを変換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すると、データのクリーニング、逆転、およびピボットを行ったり、基本キーと期間表を生成したり、さまざまな統計を計算したりすることができます。

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アプリケーションを表します。ウェアハウス・エージェントは、次のようにユーザー定義プログラム・ステップを開始できます。

- ウェアハウス・ターゲットへのデータ入力時に。
- ウェアハウス・ターゲットへのデータ入力後に。
- 単独で。

たとえば、次のプロセス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. データを操作する。
3. 一時出力リソースまたはウェアハウス・ターゲットにデータを書き込む。

ウェアハウジング・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。

- ウェアハウスで使用するプロセスを識別してグループ分けするサブジェクト・エリアを定義する。
- ソース・データ (または操作可能データ) を調査し、ウェアハウス・ソースを定義する。
- 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。
- プロセスを定義することにより、ソース・データを移動してウェアハウス・データベースの形式に変換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、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 ウェアハウス・マネージャー・パッケージがある場合は、ウェアハウス内のデータの情報カタログを作成する。情報カタログは、業務メタデータを含むデータベースです。このメタデータは、ユーザーが組織内で利用可能なデータと情報を識別し、見付けるのに役立ちます。ウェアハウスのエンド・ユーザーは、カタログを検索して、照会する表を決定することができます。
- ウェアハウス内のデータのスタースキーマ・モデルを定義する。スタースキーマとは、複数の次元表 (業務の各性質を説明する) および 1 つのファクト表 (業務に関するファクトを含む) によって構成される特殊な設計です。たとえば、ソフト・ドリンクを製造している場合、いくつかの次元表は、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り引き情報が含まれます。

- ファクト表と次元表を結合して、次元表からの詳細を発注情報と組み合わせる。たとえば、製品次元をファクト表と結合して、発注に対してそれぞれの製品がどのようにパッケージされたかに関する情報を追加できます。

これらのタスクおよび他の詳細については、 *ビジネス・インテリジェンス・チュートリアル* を使用するか、 *DB2 ユニバーサル・データベース クイックツアー* を表示するか、 *データウェアハウスセンター 管理の手引き* を参照してください。

第6章 地理情報エクステンダーについて

このセクションでは、地理情報エクステンダーを紹介し、この目的とこれが処理するデータについて説明します。地理情報エクステンダーの使用に関する詳細については、 *地理情報エクステンダー 使用者の手引きおよび解説書* を参照してください。

地理情報エクステンダーの目的

地理情報エクステンダーを使用すると、*地理情報システム* (GIS) を作成できます。 GIS はオブジェクト、データ、およびアプリケーションの複合であり、これにより地理的な地理情報を生成することが可能になります。*地勢* には、地面を形成するオブジェクトと、それを占有しているオブジェクトが含まれます。これらは自然環境 (たとえば川、森、丘、および砂漠) と文化環境 (都市、住宅、オフィスビル、陸標など) の両方を構成します。

地理情報 には次のような事実が含まれます。

- 周囲に対する地勢の位置 (たとえば、都市の中で病院やクリニックが位置している地点、または都市の住宅と地元の地震地帯との接近性など)。
- 地勢が互いにどのように関係しているか (たとえば、特定の河川系が特定の地域に含まれていること、またはその地域にある特定の橋が河川系の支流にかかっているなどの情報)。
- 1 つまたは複数の地勢に適用される尺度 (たとえば、オフィスビルとその敷地を示す線との距離、または猟鳥獣保護の境界線の長さなど)。

地理情報を、単独で使用するか、または従来のデータベース出力と組み合わせるなら、プロジェクトを設計したり、業務決定や戦略決定を行うのに役立ちます。たとえば、地域の福祉管理者が、その地域のサービスを受ける区域に福祉サービス申込者および受領者が実際どれほど住んでいるかを検査する必要があるとします。地理情報エクステンダーにより、サービスを受けている区域の位置と、申込者および受領者の住所から、この情報を引き出すことができます。

または、レストランのチェーン店のオーナーが近隣都市で業務を広げようと思っているとします。新しいレストランをどこに開くかを決定するには、オーナーは次のような質問に答える必要があります。これらの都市のどのあたりに、このレストランを頻繁に利用するタイプの人が集中しているか。主要な高速道路はどこにあるか。犯罪の割合が最も低いのはどこか。競争相手のレストランはどこにあるか。地理情報エクステンダーは、ビジュアルに地理情報を生成して、これらの質問に答えることができます。さらに、基盤となっているデータベース管理システムは、その表示を説明するためのラベルとテキストを生成できます。

地勢を表すデータ

このセクションでは、地理情報を取得するために生成、保管、および操作するデータの概要を提供します。以下のトピックが扱われています。

- データが地勢を表す方法
- 地理情報データの特性
- 地理情報データを生成する方法

データが地勢を表す方法

地理情報エクステンダーでは、地形を表または視点の行、またはそのような行の一部によって表すことができます。たとえば、2 つの地勢、オフィスビルと住宅について考えてみます。 図23 では、 BRANCHES 表の各行は、銀行の支店を表しています。このバリエーションとして、 CUSTOMERS 表の各行は全体として銀行の顧客を表しています。しかし、各行の各部分 (特に顧客の住所を含むセル) は、顧客の住宅を表しているものとして表示できます。

これらの表には、銀行の支店と顧客を識別するデータが含まれています。これらのデータは*属性データ* と呼ばれています。

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	A	A

図 23. 地勢を表している表行、住所データが地勢を表している表行. BRANCHES 表のデータ行は、銀行の支店を表しています。 CUSTOMERS 表の住所データのセルは、顧客の住所を表しています。

属性データのサブセット (支店および顧客アドレスを表す値) は、地理情報のもとになる値に変換できます。たとえば、図に示されているように、ある支店の住所は 92467 Airzone Blvd., San Jose CA 95141 です。顧客の住所は 9 Concourt Circle, San Jose CA 95141 です。地理情報エクステンダーはこれらの住所を、周囲の環境に対して支店と顧客の家が存在する位置を示す値に変換できます。 57ページの図24 は、そのような値を含む新しい行のある BRANCHES 表および CUSTOMERS 表を示します。

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141		A	A

図 24. 地理情報列が追加されている表. 各表で、LOCATION 列には、住所に対応する座標を含まれます。

住所およびそれと同様の ID を地理情報の開始点として使用するとき、これらはソース・データ と呼ばれます。これらから引き出された値が地理情報のもとになるため、これらの引き出された値は*地理情報データ* と呼ばれます。次のセクションでは、地理情報データについて説明し、さらにそれと関連するデータ・タイプを紹介します。

地理情報データの特性

多くの地理情報データは座標によって構成されています。*座標* とは、参照点に対する相対的な位置を示す数値です。たとえば、緯度は赤道に対する相対的な位置を示す座標です。経度はグリニッジ子午線に対する相対的な位置を示す座標です。したがって、Yellowstone National Park の位置は、その緯度 (北緯 44.45 度) と経度 (西経 110.40 度) によって定義されます。

緯度、経度、それらの参照点、および他の関連しているパラメーターは、まとめて*座標系* と呼ばれています。緯度と経度以外の値に基づいた座標系も存在します。これらの座標系それぞれには、独自の位置の尺度、参照点、および区別するための付加的なパラメーターがあります。

最も単純な地理情報データ項目は、単一の地勢の位置を定義する 2 つの座標によって構成されています。*データ項目* という用語は、リレーショナル表のセルを占有する値を指しています。さらに広範囲な地理情報データ項目は、道または川などの線形経路を定義する複数の座標から構成されます。 3 番目の種類のものは、区域の周辺 (たとえば、1 区画の土地や氾濫 (はんらん) 原の縁) を定義する座標によって構成されます。

それぞれの地理情報データ項目は、地理情報データ・タイプのインスタンスです。位置を示す 2 つの座標のデータ・タイプは ST_Point です。線形経路を定義する座標のデータ・タイプは ST_LineString であり、周辺を定義する座標のデータ・タイプは ST_Polygon です。これらのタイプは、地理情報データ用の他のデータ・タイプと共に、単一の階層に属する構造型となっています。

地理情報データの取得方法

地理情報データを得る方法は、次のとおりです。

- 属性データから導出する
- 他の地理情報データから導出する
- インポートする

属性データのソース・データとしての使用

地理情報データを属性データ (住所など) から導出することをジオコーディング と言います。 57ページの図24 は、地理情報データ用に指定されている 2 つの列 (1 つは BRANCHES 表にあり、もう 1 つは CUSTOMERS 表にある) を示しています。地理情報エクステンダーはこれらの表にある住所をジオコーディングし、結果出力 (住所に対応する座標) を列に入れることができます。 図25 は、その結果を図示したものです。

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094

CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	953 1527	A	A

図 25. ソース・データから導出された地理情報データを含む表. CUSTOMERS 表の LOCATION 列には、ジオコーダーが ADDRESS、CITY、STATE、および ZIP 列にある住所から導出した座標が含まれます。同様にBRANCHES 表の LOCATION 列 には、ジオコーダーが ADDRESS、 CITY、STATE、および ZIP 列にある住所から導出した座標が含まれます。

地理情報エクステンダーはジオコーダー と呼ばれる機能を使用して、属性データをジオコーディングし、結果の地理情報データを列に配置します。

他の地理情報データのソース・データとしての使用

地理情報データは属性データからだけでなく、他の地理情報データからも生成できます。たとえば、 BRANCHES 表に支店が定義されている銀行が、各支店から 5 マイル以内にどれだけの顧客がいるかを知りたいとします。この情報をデータベースから得る前に、データベースに、各支店から半径 5 マイルの地域に関する定義を提供しなければなりません。地理情報エクステンダーの ST_Buffer 関数により、このような定義を作成できます。各支店の座標を入力として使用して、この関数は、望みの地域の境界線を区別する座標を生成します。 59ページの図26 は、ST_Buffer によって提供された情報を含む BRANCHES 表を示しています。

BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	SALES_AREA
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094	1002 2001, 1192 3564, 2502 3415, 1915 3394, 1002 2001

図 26. 既存の地理情報データから導出された新しい地理情報データを含む表. SALES_AREA 列の座標は、ST_Buffer 関数によって LOCATION 列の座標から導出されます。

ST_Buffer に加えて、地理情報エクステンダーは既存の空間的データから新しい空間的データを導出する他のいくつかの関数を備えています。

地理情報データのインポート

地理情報データを取得する 3 番目の方法は、地理情報エクステンダーがサポートしている形式になっているファイルからインポートすることです。これらのファイルには、通常は地図で提供されるデータ、たとえば、人口調査トラック、氾濫 (はんらん) 原、地震障害などが含まれます。これらのデータを、生成した地理情報データと組み合わせて使用することにより、利用可能な地理情報を増大させることができます。たとえば、公共の作業部署で、自治体が影響を受けやすい災害を判別する必要がある場合は、ST_Buffer を使用して自治体を囲む地域を定義し、氾濫 (はんらん) 原および地震災害のデータをインポートすることにより、これらの問題区域が地域と重なり合っているかどうかを調べることができます。

第3部 データベースの設計

第7章 論理データベースの設計

このセクションでは、データベースの設計に含まれる次のような手順について説明します。

- 『データベースにどんなデータを記録するか』
- 65ページの『各タイプの関係の表を定義する』
- 67ページの『すべての表の列を定義する』
- 69ページの『1 つまたは複数の列を基本キーとして指定する』
- 73ページの『等しい値が必ず同じエンティティーを表すようにする』
- 73ページの『表の正規化について』
- 78ページの『制約の強制について計画する』
- 84ページの『データベース設計に関するその他の考慮事項』

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生じ得る異常をなくす必要があります。

この手順は、*論理* データベースの設計に関するものです。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。

データベース設計の*物理的な* 実装については、 87ページの『第8章 物理データベースの設計』、および *管理の手引き: インプリメンテーション* の『設計のインプリメント』に説明があります。

データベースにどんなデータを記録するか

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関係を含めます。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・データベースについては、*SQL 解説書* を参照してください。)

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*、または特性が含まれています。こうした特性は、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。各従業員には固有の従業員番号 (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関係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関係には、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。
- 部署は他の部署に対して報告を行う
  - C01 部は A00 部に対して報告を行います。
  - B01 部は A00 部に対して報告を行います。
- 従業員はいくつかのプロジェクトで働いている
  - Dolores と Heather は、IF1000 プロジェクトで働いています。
- 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関係が適用されます。たとえば、表のある行は Sally Kwan が C01 部を管理する関係を表し、別の行は、Sean O'Connell が A00 部の事務員 (clerk) である関係を表しています。

表にどんな情報が入れられるかは、表記される関係、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関係の識別に加えて、データに適用する業務規則など、他のタイプの情報も識別する必要があります。

各タイプの関係の表を定義する

データベースでは、いくつかのタイプの関係が定義できます。従業員と部署との間のいろいろな関係について考えてみましょう。従業員が 1 つの部署だけでしか働かない場合、この関係は従業員にとって*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、この関係は部署にとって*複数値* になります。従業員 (単数値) と部署 (複数値) との間の関係は、 *1 対多* の関係になります。このセクションでは、次のタイプの関係について説明します。

- 『1 対多および多対 1 の関係』
- 66ページの『多対多の関係』
- 67ページの『1 対 1 の関係』

1 対多および多対 1 の関係

1 対多および多対 1 の関係ごとに、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関係をすべて 1 つのグループにまとめます。
2. グループ内のすべての関係をまとめて 1 つの表を定義します。

次の例では、1 番目と 2 番目の関係の「多」の側が「従業員」なので、従業員の表「EMPLOYEE」を定義します。

表 5. 多対 1 の関係

エンティティー	関係	エンティティー
従業員が (Employees)	割り当てられる (are assigned to)	部門に (departments)
従業員が (Employees)	働く (work at)	仕事を (jobs)
部門が (Departments)	報告する (report to)	(管理) 部門に ((administrative) departments)

3 番目の関係の場合、「部署」が「多」の側なので、部署の表「DEPARTMENT」を定義します。

次の表は、これらの関係が表でどのように表されるかを示しています。

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

多対多の関係

多対多の関係は、双方向に複数値の関係です。 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。多対多の関係は、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。

多対多の関係 (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 の関係

1 対 1 の関係は、双方向に単一値の関係です。 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。この関係は、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。

次に、1 対 1 関係がどのように表で表されるかを示します。

DEPARTMENT 表:

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

すべての表の列を定義する

関係表の列を定義するには、次のようにします。

1. 列の名前を選択します。

   表の各列の名前は、その表で固有であることが必要です。列名の選択については、187ページの『付録A. 命名規則』で詳しく説明されています。

2. その列に対してどん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。 DB2 によって提供されるデータ・タイプおよびユーザー定義のタイプについては、 *SQL 解説書* を参照してください。

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 (すなわちグラフィック) 文字ストリング、日時、および 2 進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。

- *2 進ラージ・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 2 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。

ラージ・オブジェクト・サポートをもっと理解するためには、*SQL 解説書* を参照してください。

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、表または視点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*タイプ付き表* または*タイプ付き視点* の列の名前とデータ・タイプになります。タイプ付き表またはタイプ付き視点の行は、構造型のインスタンスの表示として考えることができます。

構造型は、表または視点の列のデータ・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、構造型のインスタンス全体をリカバリーさせることもサポートされていません。

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用されます。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、常にタイプ付き表または視点の行です。参照タイプを使用する時は、*効力範囲* を定義します。効力範囲は、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)、または視点 (*ターゲット視点* と呼ばれる) を識別します。ターゲット表または視点は、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。効力範囲付きの参照タイプのインスタンスは、タイプ付き表またはタイプ付き視点の行 (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。

ユーザー定義タイプ、構造型、参照タイプ、およびユーザー定義関数をもっと理解するためには、*SQL 解説書* を参照してください。

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、ヌル値 (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ヌル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、のどちらかを選択できます。

   ヌル値およびデフォルト値についての詳細は、*SQL 解説書* に説明があります。

1 つまたは複数の列を基本キーとして指定する

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。

固有キー は、 2 つの値が同じにならないように強制されているキーです。固有キーの列にヌル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、これを固有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有することはできません。

キーの固有性を強制させるために使用する機構を*固有索引* と呼びます。表の固有索引とは、それぞれの値が固有の行を識別する (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。固有索引にはヌル値を含めることができます。

基本キー は、 1 つの表上に定義された固有キーの 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 つだけの基本キーが可能です。

基本キーに対する *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・マネージャーが基本キーを固有のものにするためのものです。 (さらに、非基本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″ 固有キーがない場合や、到着順によって固有の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。 (71ページの『識別列の定義』も参照。)

サンプル表の一部に使用されている基本キーは、次のとおりです。

表	キー列
従業員表	EMPNO
部署表	DEPTNO
プロジェクト表	PROJNO

PROJECT 表の一部に基本キーを示した例を次に示します。

表 6. PROJECT 表の基本キー

PROJNO(基本キー)	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 つの列だけに基本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、固有の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、固有の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、複数の列を含む基本キー (複合キー) を示すものです。

表 7. EMP_ACT 表の複合基本キー

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

キー列の候補を識別する

キーの候補を識別するには、固有の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。 21ページの表2 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。

いくつかのキーの候補から基本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、それぞれの行の基本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、基本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、ある時点で固有であっても、そのことが保証されているわけではありません。基本キーとしては従業員番号列が最もよいと言えます。従業員には一度だけ固有の番号が与えられ、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。

識別列の定義

識別列 は、DB2 が自動的に表の各行に固有の数値を生成する手段となります。表には、識別属性が定義されている単一列を入れることができます。識別列の例には、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。

識別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 (generated always)* として定義されている識別列は、 DB2 によって固有であることが保証されています。この値は常に DB2 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 (generated by default)* として定義されている識別列は、アプリケーションが識別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 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 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。

識別列は、固有の基本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識別列を使用することにより、データベースの外で独自の固有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性と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、カウンターが識別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性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。

識別列のカウンターは、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識別カウンターを 2 回増分した場合、同時に他のトランザクションが同じ識別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識別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、結果として並列性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、識別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、特定の識別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。

識別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、スケールがゼロの正確な数値データ・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。
- 識別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、固有の値が引き続き生成されることが保証されます。
- 識別列値をキャッシュすると、パフォーマンスが向上します。

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。接続する列の名前は互いに違っていてもかまいません (前の例では「WORKDEPT」と「DEPTNO」)。また、同じ名前であってもかまいません (部署表とプロジェクト表では「DEPTNO」という列)。

表の正規化について

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、キー列以外の列が基本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。そうでないなら、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 (74ページの『第 1 正規形』を参照)。

第 2　キーの一部ではない各列は、キーに従属しています (74ページの『第 2 正規形』を参照)。

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 (76ページの『第 3 正規形』を参照)。

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 (77ページの『第 4 正規形』を参照)。

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 8. 第 1 正規形に違反する表

PART (基本キー)	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 9. 第 1 正規形に適合する表

PART (基本キー)	WAREHOUSE (基本キー)	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 10. 第 2 正規形に違反している表

PART (基本キー)	WAREHOUSE (基本キー)	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

表 10. 第 2 正規形に違反している表 (続き)

PART (基本キー)	WAREHOUSE (基本キー)	QUANTITY	WAREHOUSE_ADDRESS
P0020	Warehouse D	278	800 Massey Street

基本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 11. 第 2 正規形に適合する部品在庫表

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

表 12. 第 2 正規形に適合する倉庫表

WAREHOUSE (基本キー)	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。

パフォーマンスの面での考慮事項については、 *管理の手引き: パフォーマンス* の『アプリケーション・パフォーマンスのチューニング』を参照してください。

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 (表14を参照)。新しい列は WORKDEPT に依存しますが、基本キーは 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ます。更新後の表に結果として矛盾が生じます。

表 13. 更新前の正規化されていない EMPLOYEE_DEPARTMENT 表

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

表 14. 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 (基本キー)	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 15. 「従業員 - 部署」表を正規化した後の従業員表

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

表 16. 「従業員 - 部署」表を正規化した後の部署表

DEPTNO (基本キー)	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複数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 通りの関係があります。 1 つの表でその両方の関係を表すとすれば、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 17. 第 4 正規形に違反する表

EMPNO (基本キー)	SKILL (基本キー)	LANGUAGE (基本キー)
000130	Data Modelling	英語
000130	Database Design	英語
000130	Application Design	英語
000130	Data Modelling	スペイン語
000130	Database Design	スペイン語
000130	Application Design	スペイン語

むしろ、これらの関係は 2 つの表で表すべきです。

表 18. 第 4 正規形に適合している EMPLOYEE_SKILL 表

EMPNO (基本キー)	SKILL (基本キー)
000130	Data Modelling

表 18. 第 4 正規形に適合している EMPLOYEE_SKILL 表 (続き)

EMPNO (基本キー)	SKILL (基本キー)
000130	Database Design
000130	Application Design

表 19. 第 4 正規形に適合している EMPLOYEE_LANGUAGE 表

EMPNO (基本キー)	LANGUAGE (基本キー)
000130	英語
000130	スペイン語

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。

制約の強制について計画する

制約 とは、データベース・マネージャーが実施する規則の 1 つのことです。このセクションでは、4 つのタイプの制約処理について説明します。

固有制約
表のキー値を必ず固有にします。基本キーを構成する列に対するすべての変更については、固有かどうかが検査されます。

参照保全
挿入、更新、削除の操作時に参照制約を施行します。この場合、データベースは、すべての外部キーのすべての値が有効になります。

表検査制約
変更されたデータが、表の作成または変更時に指定された条件に違反していないかどうかを検査します。

トリガー
指定された表に対する更新、削除、または挿入の操作によって呼び出されたときに実行される 1 組のアクションを定義します。

固有制約

固有制約 とは、1 つのキーの値が、表内で必ず固有になるという規則のことです。固有制約内のキーを構成する各列は、NOT NULL として定義されていなければなりませ

ん。固有制約は、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されます。

1 つの表は任意の数の固有制約を持つことができますが、 1 つの表に対する基本キーとして定義できるのは 1 つの固有制約だけです。また、1 つの表は、同じ列のセット上では、複数の固有制約を持つことはできません。

固有制約が定義されると、データベース・マネージャーは、 (必要ならば) 固有索引を作成し、それをシステム必須の 1 次索引または固有索引のいずれかとして指定します。この制約は、固有索引を通して施行されます。固有制約が 1 つの列でいったん確立されると、複数行の更新中における固有性の検査は、更新の終わりまで延期させられます。

固有制約は、参照制約の中の親キーとしても使用することができます。

参照保全

データベース・マネージャーは、*参照制約* を介して参照制約を保守します。この関係においては、表の中の特定の属性や列の値が別の表や列にも存在していることが必要となります。たとえば、ある参照制約では、従業員表の全従業員が部署表の中に含まれているいずれかの部署に所属している必要があります。従業員が、存在しない部署に所属していてはなりません。

データベースの中に参照制約を作成すると、参照保全が必ず維持されるようになり、照会をより能率良く処理するために、最適化プログラムがこれらの特殊な関係の情報を活用できるようになります。参照保全を計画する時は、データベースの表と表の間の関係をすべて識別することが必要です。基本キーと参照制約を定義すれば、関係を識別できるようになります。

次の関連表を考慮してください。

表 20. DEPARTMENT 表

DEPTNO (基本キー)	DEPTNAME	MGRNO
A00	Spiffy Computer Service Div.	000010
B01	Planning	000020
C01	Information Center	000030
D11	Manufacturing Systems	000060

表 21. EMPLOYEE 表

EMPNO (基本キー)	FIRSTNAME	LASTNAME	WORKDEPT (外部キー)	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423

表 21. EMPLOYEE 表 (続き)

EMPNO (基本キー)	FIRSTNAME	LASTNAME	WORKDEPT (外部キー)	PHONENO
000120	Sean	O'Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

次に示す概念の多くは、参照保全を理解するのに役立つものであり、これらの表と関連して説明されています。

固有キー とは、値が他のどの行とも重複していない 1 つの列または 1 組の列のことです。 1 つの固有キーを表の基本キーとして定義できます。固有キーは、外部キーによって参照された場合、*親キー* とも呼ばれます。

基本キー とは、表の定義の一部である固有キーのことです。それぞれの表は、1 つの基本キーのみを持つことができます。前述の表では、DEPTNO と EMPNO が、それぞれ DEPARTMENT 表と EMPLOYEE 表の基本キーです。

外部キー とは、同じ表または別の表の固有キーまたは基本キーを参照する、表内の 1 つの列または 1 組の列のことです。外部キーは、表と表の間の参照保全を実施するために、固有キーまたは基本キーとの関係を確立するために使用されます。 EMPLOYEE 表の WORKDEPT 列は、 DEPARTMENT 表の DEPTNO という基本キーを参照しているため、外部キーになっています。

複合キー とは、複数の列を持つキーのことです。基本キーと外部キーの両方が複合キーになることができます。たとえば、部署が部番号と課番号の組み合わせによって一意的に識別される場合、 DEPARTMENT 表のキーを作成するには 2 つの列が必要になります。

親キー とは、参照制約の基本キーまたは固有キーのことです。*基本キー制約* は、親キー列のセットが指定されていないときの参照制約のデフォルト親キーです。

親表 とは、同じ表または別の表の少なくとも 1 つの外部キーに関連づけられた親キーが含まれる表のことです。 1 つの表が、任意の数の関係において親となることが可能です。たとえば、DEPTNO が基本キーの DEPARTMENT 表は、 WORKDEPT という外部キーを含む EMPLOYEE 表の親表です。

親行 とは、その親キーの値が従属表の中の少なくとも 1 つの外部キーと一致する、親表の 1 つの行のことです。親表の行がすべて親行であるとは限りません。
DEPARTMENT 表の第 4 行 (D11) は、EMPLOYEE 表の第 3 行と第 6 行の親行です。 DEPARTMENT 表の第 2 行 (B01) は、他のどの行に対しても親行になっていません。

従属表 とは、外部キーを含んだ表のことです。従属表が親表でもある場合もあります。表は、任意の数の関係において従属表となることが可能です。 EMPLOYEE 表には WORKDEPT という外部キーが含まれており、基本キー DEPTNO のある DEPARTMENT 表に対する従属となっています。

従属行 とは、親キーの値と一致する非ヌルの外部キー値を持つ従属表の 1 つの行のことです。外部キーの値は、従属行から親行への参照を表します。外部キーがヌル値のこともあるため、従属表の行がすべて従属行であるとは限りません。

表が *子孫* であるとは、それが従属表であるか、または従属表の子孫であるということです。子孫表には、ある表の親キーまでトレースバックできる外部キーが含まれています。

参照循環 とは、表をその表自体に接続する経路のことです。表がその表自体に直接接続されている場合、それは*自己参照* 表です。 EMPLOYEE 表の中に各従業員のマネージャーの EMPNO の含まれる MGRID という別の列がある場合、 EMPLOYEE 表は自己参照表となります。その MGRID は、EMPLOYEE 表の外部キーとなります。

自己参照表は、同一の関係において親表でもあり従属表でもあります。自己参照行とは、その行自体の親行でもあり従属行でもある行のことです。この状態で存在する制約は、自己参照制約と呼ばれます。たとえば、自己参照表の 1 つの行の中の外部キーの値がその行の中の固有キーの値と一致する場合、その行は自己参照になります。

参照制約 とは、指定された外部キーの非ヌル値が、指定表の固有キーの値としても現れる場合にのみ有効であることを表明することです。参照制約の目的は、データベースの関係を維持し、データ入力規則に従うようにすることです。

SQL 操作に及ぼす影響

参照制約を施行すると、表が親表であるか従属表であるかに依存する一部の SQL 操作に特別な影響が及びます。ここでは、参照保全の保守が SQL の INSERT、DELETE、UPDATE、および DROP の操作に及ぼす影響について説明します。

DB2 は、自動的にはシステムでの参照制約を適用しません。そのため、システム間で参照制約を施行したい場合は、アプリケーションの中に必要な論理が含まれていなければなりません。

この部分では、次の点について説明します。

- 『INSERT 規則』
- 82ページの『DELETE 規則』
- 83ページの『UPDATE 規則』

INSERT 規則: 親表への行の挿入は、従属表に対するアクションを何も行うことなく、いつでも実行できます。ただし、外部キーがヌル値である場合以外は、親キー値を

持つ親表の中に、挿入される行の外部キーの値と等しい行がないかぎり、行を従属表に挿入することはできません。複合外部キーの値は、値のいずれかのエレメントがヌル値であれば、ヌル値になります。

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。

参照制約を持っている表に行を挿入することを試みる場合、親キーに何らかの非ヌルの外部キー値が存在しないと、INSERT 操作は許されません。複数の行を挿入しようとした時に 1 つの行に関して INSERT 操作が失敗した場合、行はどれも挿入されません。

DELETE 規則: 親表から行を削除する場合、DB2 は、従属表の中に、外部キーの値と一致する従属行があるかどうかを調べます。従属行が見つかると、いくつかのアクションが実行されることがあります。どんなアクションを実行させるかは、従属表の作成時に*削除* 規則を指定することによって指定できます。

基本キー削除時の従属表 (外部キーを含む表) の削除規則は、次のとおりです。

RESTRICT
従属行が見つかると、親表の行の削除は禁止されます。親行と従属行の両方を削除する必要がある場合は、従属行の方をまず削除してください。最初に親行を削除しようとしても、参照制約に違反するため実行されません。

NO ACTION
すべての参照制約が適用された後、それぞれの子に対して、親行の存在を強制します。

CASCADE
親表の行を削除すると、従属表の中の関連する行が自動的に削除されます。この規則は、親表の行がなければ従属表の行が無意味になってしまう場合に役立ちます。

最初に親行を削除すると、基本キーを参照する従属行が自動的に削除されます。したがって、従属行を最初に削除する必要はありません。従属行自体に従属行がある場合にも、こうした関係についての削除規則が当てはまります。連鎖削除は DB2 によって管理されます。

SET NULL
親表から行を削除すると、従属行の外部キーの値が必ずヌル値に設定されます。行の他の部分は変更されません。

表の作成時に削除規則を明示的に定義しない場合は、NO ACTION 規則が適用されます。

削除操作に関係する表は、連結削除表と呼ばれます。連結削除関係には、次の制限が適用されます。

- 複数の表の参照循環において、表がその表自体に対して連結削除されることはありません。
- 複数の従属関係によって表が別の表に連結削除されている場合、それらの関係の削除規則は、CASCADE か NO ACTION のどちらか同じものでなければなりません。
- 自己参照表が CASCADE 関係で別の表の従属表になっている場合、自己参照関係の削除規則も CASCADE でなければなりません。

従属表の行の削除は、親表に対するアクションなしにいつでも実行できます。たとえば、部署 - 従業員の関係で、従業員が退職する場合、部署表には影響を与えずに、従業員表からその従業員の行を削除することができます。 (従業員 - 部署の逆関係は、部署のマネージャー ID が従業員表の親キーを参照する外部キーになります。マネージャーが退職する場合は、部署表に影響します。)

UPDATE 規則: DB2 では、親行の固有キーの更新は禁止されています。従属表の外部キーを更新する場合で、外部キーがヌル値でない場合は、その外部キーは、その関係の親表の基本キーのいずれかの値に一致しなければなりません。参照制約の中に UPDATE 操作で違反しているものがあれば、エラーになり、どの行も更新されません。

親キーの列の値が更新される場合、以下のようになります。

- 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 であると更新は拒否されます。
- 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 であると更新は拒否されます。

親行の中にある親キーの値を更新するためには、以下のいずれかを行うことによって、まず従属表内のすべての子行との関係を除去しなければなりません。

- 子行を削除する。
- 従属表の中の外部キーを更新して、別の有効なキー値が含まれるようにする。

行の中のキー値との従属関係がない場合、行は参照関係の親ではなくなり、更新することができます。

外部キーの一部が更新され、しかも外部キーにヌル値の部分がない場合、外部キーの新しい値は、親表の中の固有キーの値を示していなければなりません。特定の固有キーに従属する外部キーがない場合、つまり、固有キーを含んでいる行が親行ではない 場合、固有キーの一部は更新できます。ただし、固有キーの作業を行っており、重複行が許されないので、この場合は、更新する行として 1 つの行しか選択できません。

表検査制約

設計の中で指定する業務上の規則は、表検査制約によって施行できます。*表検査制約*は、表の行ごとに適用される探索条件を指定します。これらの制約は、表に対して更新

または挿入ステートメントを適用したときに自動的に活動化されます。また、CREATE TABLE または ALTER TABLE ステートメントを介して定義されます。

表検査制約は、妥当性検査のために使用できます。たとえば、部署番号の値は 10 ～ 100 の範囲になければならず、従業員の職務名は、″Sales″、″Manager″、または ″Clerk″ だけが可能であり、入社歴 8 年を超える従業員は $40,500 を超える収入がなければならないなどです。

IMPORT および LOAD コマンドへの表検査制約の影響についての詳細は、 *データ移動ユーティリティー手引きおよび解説書* を参照してください。

トリガー

トリガー とは、指定された表に対して、削除、挿入、または更新の操作が行われたときにいつでも実行される、定義済みのアクションのセットのことです。業務規則をサポートするため、トリガーを定義できます。トリガーは、要約や監査データを自動更新する時にも使用できます。トリガーはデータベースに格納されるため、どのアプリケーション・プログラムでもアクションをコーディングする必要がありません。トリガーは 1 度だけコーディングして、データベースに保管します。アプリケーションがそのデータベースを使用すると、トリガーは、必要に応じて自動的に DB2 から呼び出されます。これにより、データに関連した業務規則は必ず適用されます。業務規則を変更した場合は、トリガーのみを変更する必要があります。

トリガー起動される SQL ステートメントからは、ユーザー定義関数 (UDF) を呼び出すことができます。これによって、トリガー起動時に、トリガー起動されるアクションによって非 SQL 操作を実行できます。たとえば、警報機構として電子メールを送信することができます。トリガーの詳細については、 *管理の手引き: インプリメンテーション* の『トリガーの作成』と、 *アプリケーション開発の手引き* を参照してください。

データベース設計に関するその他の考慮事項

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。

設計時に考慮しなければならないその他の要件としては、*監査活動*、ヒストリー・データ、*要約表*、セキュリティー、*データ・タイプ* および*並列処理能力* などがあります。

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 監査機能でも行えます。詳細については、 *管理の手引き: インプリメンテーション* の『DB2 アクティビティーの監査』を参照してください。

パフォーマンス上の理由で、*ヒストリー* 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、ヒストリー・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、*要約* 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要約表が役立ちます。要約表について詳しくは、 *管理の手引き: インプリメンテーション* の『要約表の作成』 を参照してください。

セキュリティー の影響についても、設計中に識別すべきです。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。セキュリティーと許可の詳細については、 *管理の手引き: インプリメンテーション* の『データベース・アクセスの制御』を参照してください。

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。そのようなタイプ付き表については、*タイプ階層* と呼ばれる表の間で定義された関係を使用して、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、常にタイプ付き表または視点の行です。

タイプ付き行および表を含む設計の実装の詳細については、 *管理の手引き: インプリメンテーション* の『設計のインプリメント』を参照してください。階層構造になっている表の間でのデータの移動の詳細については、 *データ移動ユーティリティー手引きおよび解説書* を参照してください。

業務が拡大するにつれて、DB2 エンタープライズ拡張エディションによって提供される追加の容量とパフォーマンス能力を必要とする場合があります。この環境では、データベースは複数のマシンまたはシステムにわたって区分化され、それぞれが、全体のデータベースの一部分の記憶と検索についての責任を持ちます。それぞれの区画 (またはノード) は並列に働いて、 SQL またはユーティリティーの操作を処理します。

並列操作に関する問題点と考慮事項については、本書を通じて説明されています。

第8章 物理データベースの設計

論理データベースの設計 (63ページの『第7章 論理データベースの設計』を参照) 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について考慮す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めに作成されるファイルについて、データを格納するのに必要なスペースの大きさについて、またデータの格納に必要な表スペースの使用法について考慮しなければなりません。

この部分では、次の点について説明します。

- 『データベース・ディレクトリー』
- 90ページの『表スペース所要量の見積もり』
- 97ページの『さらに必要となるスペース量』
- 98ページの『ノードグループの設計』
- 106ページの『表スペースの設計と選択』
- 128ページの『連合データベースの設計についての考慮事項』

データベース・ディレクトリー

データベースを作成すると、 DB2 は制御ファイル (ログ・ヘッダー・ファイルなど) を格納したり、デフォルトの表スペースにコンテナーを割り振ったりするためのサブディレクトリーを作成します。データベースに関連するオブジェクトは、常にデータベース・ディレクトリーの中に保管されるとは限りません。これらのオブジェクトは、装置上も含めて、さまざまな場所に保管される可能性があります。

データベースは、DB2INSTANCE 環境変数で定義されたインスタンス、または (ATTACH コマンドを使って) 明示的に付加したインスタンスに作成されます。インスタンスについての概要は、 *管理の手引き: インプリメンテーション* の『データベース・マネージャーの複数インスタンスを使用する』を参照してください。

UNIX ベースのシステムで使用される命名方式は、以下のとおりです。

```
specified_path/$DB2INSTANCE/NODEnnnn/SQL00001
```

OS/2 および Windows オペレーティング・システムで使用される命名方式は、以下のとおりです。

```
D:¥$DB2INSTANCE¥NODEnnnn¥SQL00001
```

説明

- specified_path は、インスタンスをインストールするためのオプションのユーザー指定の位置です。
- NODEnnnn は、区分データベース環境でのノード ID です。最初のノードは、NODE0000 です。
- D: は、ルート・ディレクトリーのあるボリュームを識別するドライブ文字です。

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。

サブディレクトリーは、データベースを作成しているときに生成されたデータベース・マネージャーのインスタンスと同じ名前のディレクトリー内に作成されます。 (OS/2 および Windows オペレーティング・システムでは、サブディレクトリーは、「ドライブ文字」によって識別されるボリュームのルート・ディレクトリーに作成されます。) これらのインスタンスおよびデータベース・サブディレクトリーは、 CREATE DATABASE コマンドで指定されたパス内に作成されます。それらの保守は、データベース・マネージャーが自動的に行います。プラットフォームによっては、それぞれのインスタンスが、そのインスタンスが属しているデータベースに対するシステム管理者 (SYSADM) 権限を持ったインスタンス所有者によって所有されている場合があります。

問題の発生を避けるため、同じ命名方式でディレクトリーを作成することがないようにしてください。また、データベース・マネージャーによってすでに作成されているディレクトリーを操作することもしないでください。

データベース・ファイル

データベースには、以下のようなファイルが関連付けられます。

ファイル名	説明
SQLDBCON	このファイルには、データベースの調整パラメーターとフラグが入れられます。データベースの構成パラメーターの変更については、 *管理の手引き: パフォーマンス* を参照してください。
SQLOGCTL.LFH	このファイルは、データベースのすべてのログ・ファイルの記録と制御に使用されるものです。
Syyyyyyy.LOG	データベースのログ・ファイル。番号は 0000000～9999999。これらのファイルの番号は、データベース構成パラメーター *logprimary* および *logsecond* によって制御されます。個々のファイルのサイズは、データベース構成パラメーター *logfilsiz* によって制御されます。 循環ログの場合、ファイルは再使用され、同じ番号のままになります。アーカイブ・ログの場合、ログがアーカイブされたり新しいログが割り振られたりするたびに、ファイル番号が順に増えてゆきます。9999999 まで達すると、番号は折り返して 0000000 になります。

デフォルトには、これらのログ・ファイルは SQLOGDIR というディレクトリーに格納されます。 SQLOGDIR は、SQL*nnnnn* サブディレクトリーの中にあります。

SQLINSLK このファイルは、各データベースが常にただ 1 つのデータベース・マネージャー・インスタンスによって使用されるようにします。

SQLTMPLK このファイルは、各データベースが常にただ 1 つのデータベース・マネージャー・インスタンスによって使用されるようにします。

SQLSPCS.1 このファイルには、データベース内のすべての表スペースの定義と現行の状態が入れられます。

SQLSPCS.2 このファイルは、SQLSPCS.1 のバックアップ・コピーです。これらのファイルのうちのいずれかがないと、データベースにアクセスできません。

SQLBP.1 このファイルには、データベース内で使用されるすべてのバッファー・プールの定義が含まれます。

SQLBP.2 このファイルは、SQLBP.1 のバックアップ・コピーです。これらのファイルのうちのいずれかがないと、データベースにアクセスできません。

DB2RHIST.ASC このファイルは、データベースのヒストリー・ファイルです。このファイルには、バックアップ操作や復元操作など、データベースに対する管理操作のヒストリーが保管されます。

DB2RHIST.BAK
このファイルは、DB2RHIST.ASC のバックアップ・コピーです。

注:

1. これらのファイルは、直接変更しないでください。これらのファイルにアクセスできるのは、文書 API およびその API を実現するためのツール (コマンド行プロセッサーやコントロール・センターなど) によって間接的にアクセスする場合のみです。
2. これらのファイルは移動しないでください。
3. これらのファイルは削除しないでください。
4. データベースや表スペースのバックアップ用にサポートされている唯一の手段は、**sqlubkp** (データベースのバックアップ) API です。これには、コマンド行プロセッサーおよびその API を実装するコントロール・センターが含まれます。

表スペース所要量の見積もり

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、さまざまなオブジェクト間の関係を判別できます。
- 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。このユーティリティーについての詳細は、*コマンド解説書* を参照してください。

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。

- 91ページの『システム・カタログ表』
- 91ページの『ユーザー表データ』
- 93ページの『長形式フィールドのデータ』
- 93ページの『ラージ・オブジェクト (LOB) データ』
- 94ページの『索引スペース』

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

システム・カタログ表

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。詳しくは、106ページの『表スペースの設計と選択』を参照してください。

注: 複数区画を持つデータベースの場合、カタログ表は CREATE DATABASE の発行元の区画上にのみ存在します。カタログ表のディスク・スペースは、その区画のためだけに必要です。

ユーザー表データ

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、76 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 4020 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含まれます。 (これらのデータ・タイプを含む表オブジェクトに必要なスペースの見積もりについては、93ページの『長形式フィールドのデータ』および 93ページの『ラージ・オブジェクト (LOB) データ』を参照してください。)

通常、行は先頭一致順で表に挿入されます。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、4 KB 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、オリジナルの行位置にレコードが 1 つ作成され、更新された行の表ファイルの中の新しい位置を指し示します。

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。このステートメントについての詳細は、*SQL 解説書* を参照してください。

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(4020/(平均行サイズ + 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 (各列のサイズについては、 *SQL 解説書* で CREATE TABLE ステートメントを参照)。また 1.1 はオーバーヘッド因数です。

注: この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ふさわしいでしょう。

バックアップを異なるページ・サイズに復元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。表へのデータのインポートについて、および IXF データ・ファイルについて詳しくは、 *データ移動ユーティリティー手引きおよび解説書* を参照してください。

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除去されます。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。

長形式フィールドのデータ

長形式フィールド・データは、他のデータ・タイプとは構造が異なる、別個の表オブジェクトに格納します (91ページの『ユーザー表データ』および 『ラージ・オブジェクト (LOB) データ』を参照してください)。

データが格納される 32 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。)

長形式フィールド・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、長形式フィールド・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貫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、長形式フィールド・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。

ラージ・オブジェクト (LOB) データ

ラージ・オブジェクト (LOB) データは、他のデータ・タイプとは構造が異なる、 2 つの別個の表オブジェクトに格納されます (91ページの『ユーザー表データ』および 『長形式フィールドのデータ』を参照してください)。

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、最少量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、更新および削除活動の量と挿入する 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。

索引スペース

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。

```
(平均索引キー・サイズ + 8) × 行数 × 2
```

ここで、

- 「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。異なるデータ・タイプを持つ列のバイト・カウントを計算する方法については、 *SQL 解説書* の CREATE TABLE ステートメントの説明を参照してください。 (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 1 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、非葉ページやフリー・スペースなどのオーバーヘッドのためです。

注: ヌル値が可能な列の場合は、ヌル値標識のために 1 バイトをさらに追加してください。

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。

(平均索引キー・サイズ + 8) × 行数 × 3.2

ここで 3.2 は索引オーバーヘッドの因数、および索引の作成時のソートに必要なスペースの因数です。

注: 非固有索引の場合、重複キー項目を保管するために、4 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、上記の式では余分に見積もってしまう場合があります。

葉ページの数を見積もるために、以下の 2 つの式を使用できます (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。

- 葉ページ当たりのキーの数は、おおよそ以下のように見積もることができます。

$$\frac{(.9 * (U - (M*2))) * (D + 1)}{K + 6 + (4 * D)}$$

  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 / (8 + *minimumKeySize*)

  – D = キー値当たりの重複の平均数

  – K = *averageKeySize*

  minimumKeySize と *averageKeysize* は、それぞれヌル可能なキー部、および各可変長キー部のために余分に 1 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。

- 葉ページ当たりのキーの数をより正確に見積もるには、以下のようにします。

  L = 葉ページの数 = X / (葉ページのキーの平均数)

  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/(葉ページのキーの平均数)) * ページ・サイズ

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 / 更新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。) 非葉ページ当たりのキーの数は、おおよそ以下のようになります。

```
 (.9 * (U - (M*2))) * (D + 1)
----------------------------
     K + 12 + (8 * 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 は非葉ページ上のキー値あたりの平均重複数 (この数は葉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 / (8 + 非葉ページの *minimumKeySize*)
- K = 非葉ページの *averageKeySize*

非葉ページの *minimumKeySize* および *averageKeySize* は、組み込み列が存在する場合を除いて、葉ページのものと同じです。組み込み列は、非葉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・スペースが非葉ページに残されるからです。

非葉ページの数は、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 は葉ページの数。
- N は各非葉ページのキーの数。
- 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
```

さらに必要となるスペース量

さらに、次のようなスペースも必要になります。

- 『ログ・ファイル・スペース』
- 98ページの『一時ワークスペース』

ログ・ファイル・スペース

ログ・ファイルに必要なスペース量 (バイト) の範囲は、

```
( logprimary * (logfilsiz + 2 ) * 4096 ) + 8192
```

から、

```
( (logprimary + logsecond) * (logfilsiz + 2 ) * 4096 ) + 8192
```

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、データベースの構成ファイルに定義されている 2 次ログ・ファイルの数です。
- *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。
- 8192 はログ制御ファイルのサイズ (バイト数) です。

logprimary、*logsecond*、および *logfilsiz* 構成パラメーターについての詳細は、 *管理の手引き: パフォーマンス* を参照してください。

注: アクティブ・ログ・スペースの合計は 32 GB を超えてはなりません。

ログ・ファイル・スペースの上限は、データベース・マネージャーがランタイムに必要とする 2 次ログ・ファイルの実際の数によって変わります。上限が必要となるのは、時々発生する、ボリュームが大きい活動のときだけです。全く必要ないこともあります。

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・ファイルを別の位置に移動しない限り、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、ユーザー出口プログラムによって、アーカイブ・ログ・ファイルが別の位置に移動されます。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。
  - ユーザー出口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。

一時ワークスペース

一部の SQL ステートメントには、処理のための一時スペースが必要です (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および戻される表のサイズに依存するため、見積もることはできません。

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。

ノードグループの設計

ノードグループ とは、 1 つのデータベースに属するものとして定義されている 1 つ以上のノードのセットに名前を付けたものです。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区画は、 *db2nodes.cfg* と呼ばれる**区画構成ファイル**の中にあらかじめ定義されていなければなりません。 1 つのノードグループには、最小で 1 つのデータベース区画を、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区画を含めることができます。

新しいノードグループを作成するには CREATE NODEGROUP ステートメントを使用し、変更するには ALTER NODEGROUP ステートメントを使用します。ノードグループ内では、1 つまたは複数のデータベース区画を追加または除去できます。ノードグループを変更する前に、データベース区画が `db2nodes.cfg` ファイルの中に定義されていなければなりません。表スペースはノードグループ内に存在します。表は、表スペース内に存在します。

ノードグループが作成または修正されるときには、*区分化マップ* がそのノードグループに関連付けられます。区分化マップは、*区分化キー* 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、ノードグループ内のどのデータベース区画が特定のデータの行を保管するかが決められます。区分化マップについて詳しくは、101ページの『区分化マップ』を参照してください。区分化キーについて詳しくは、102ページの『区分化キー』を参照してください。

非区分データベースでは、区分化キーおよび区分化マップは必要ありません。非区分データベースを使用している場合には、ノードグループについての設計上の考慮事項はありません。*データベース区画* 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ノードグループは、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・ノードグループです。 IBMTEMPGROUP は、システム一時表スペース用のデフォルト・ノードグループです。 IBMDEFAULTGROUP は、ユーザーがそこに書き込むことを選択できる、ユーザー定義の表が入る表スペース用のデフォルト・ノードグループです。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ノード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。

複数区画のノード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。

- 複数区画のノードグループでは、区分化キーのスーパーセットである場合にのみ、固有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区画の数によっては、 1 つまたは複数の単一区画ノードグループ、および 1 つまたは複数の複数区画ノードグループを作成できます。
- 各データベース区画には固有の区分番号を割り当てる必要があります。同じデータベース区画が 1 つまたは複数のノード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区画を速やかにリカバリーさせるためには、同じデータベース区画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 ノードグループのデータベース区画を含まないノード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 *併置* を利用したい場合を除き、単一区画ノードグループの中に置いてください。併置とは、同じデータベース区画にある、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。併置された表を使用して、DB2 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。併置された表は、1 つの単一区画ノードグループの中に存在することができます。複数の表が 1 つの複数区画ノードグループの中に存在し、区分化キーの中に同数の列を持ち、対応する列のデータ・タイプが区画互換である場合、それらの表は併置されていると見なされます。同じ区分化キー値を持つ

併置された表の中の行は、同じデータベース区画に置かれます。それぞれの表が同じノードグループの中の別個の表スペースの中に入っていても、併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区画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 区画ノードグループ上よりも、 16 区画ノードグループ上のほうがパフォーマンスが良くなります。

ノード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。

ノードグループ設計についての考慮事項

データベースを区分化する必要があるかどうかは、論理データベースの設計内容、および処理するデータの量によって判断が可能です。このセクションでは、データベースの区分化について次のようなトピックを扱います。

- 『データの区分化』
- 101ページの『区分化マップ』
- 102ページの『区分化キー』
- 104ページの『表の併置』
- 105ページの『区画の互換性』
- 105ページの『複製要約表』

データの区分化

DB2 は、データベース内の複数のデータベース区画にわたってデータを保管できるようにする、区分化ストレージ・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区画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。区分化された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、区分化キーを宣言することによって、自分のデータを区分化する方法を選択することができます。ユーザーはまた、データを保管すべき表スペースと関連するノード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区画を指定したりデータベース区画の数を決定することもできます。さらに、更新可能な区分化マップをハッシュ・アルゴリズムとともに使用して、データベース区画への区分化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、大きな表のための作業負荷を区分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区画に保管することができます。それぞれのデータベース区画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区画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 は*部分デクラスター* 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区画のサブセット (つまりノードグループ) に分割できます。

それぞれのデータベース区画に表を置きたいときに考慮できる別の方法は、要約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要約表を作成して、それを各ノードに複製します。詳しくは、105ページの『複製要約表』を参照してください。

区分化マップ

区分データベース環境では、どのデータベース区画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*区分化マップ* というマップを使用します。

区分化マップは内部で生成された配列であり、複数区画ノードグループの場合は 4 096 項目が、単一区画ノードグループの場合は単一の項目が入っています。単一区画ノードグループの場合、区分化マップの項目は 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区画の区画番号が入っています。複数区画ノードグループの場合、ノードグループの区画番号は、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、*区分化キー* を使用して、データが保管されている場所 (データベース区画) を判別します。

たとえば、 4 つのデータベース区画 (0 ～ 3 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 ノードグループの区分化マップは、次のようになります。

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

データベース区画の 1 および 2 を使用してノードグループがデータベース内に作成されている場合、そのノードグループの区分化マップは、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区分化キーが 1 と 500 000 の間の可能値を持つ整数である場合、区分化キーは、0 と 4 095 の間の区分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区画を選択するための区分化マップの索引として使用されます。

102ページの図27 は、区分化キー値 (c1, c2, c3) を持つ行が、区画 2 にマップされ、次に区画 2 がデータベース区画 n5 を参照する方法を示しています。

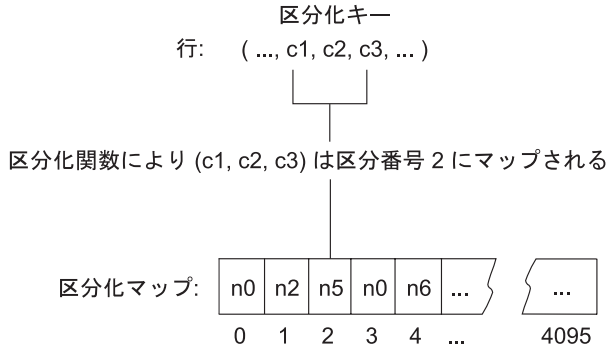

図 27. 区分化マップを使用したデータ配分

区分化マップは、区分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区画にわたるデータ配分を将来変更する必要がある場合、データ再配分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、データ配分のバランスをとり直したり、データ配分にスキューを導入したりすることができます。このユーティリティーについての詳細は、 *管理の手引き: パフォーマンス* の『データベース区画間でのデータの再配分』を参照してください。

表区分情報入手 (**sqlugtpi**) API を使用して、見ることができる区分化マップのコピーを入手することができます。この API についての詳細は、*管理 API 解説書* を参照してください。

区分化キー

区分化キー とは、特定のデータの行が保管される区画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。区分化キーは、CREATE TABLE ステートメントを使用して表の上に定義されます。ノードグループ内の複数のデータベース区画にわたって分割された表スペース内の表に対して区分化キーが定義されていない場合、デフォルトによって、区分化キーが基本キーの最初の列から作成されます。基本キーが指定されていない場合は、デフォルトの区分化キーは、その表に定義された最初の長形式フィールド列以外の列になります。 (*長形式* には、すべての長形式データ・タイプとすべてのラージ・オブジェクト・データ・タイプが含まれます。) 単一区画ノード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、区分化キーが必要であれば、区分化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、区分化キーは作成されません。

デフォルトの区分化キーの要件を満たす列がない場合、表は区分化キーなしで作成されます。区分化キーのない表は、単一区画ノード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区分化キーを追加または除去できます。区分化キーの変更は、単一区画ノード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。

適切な区分化キーを選択することが重要です。以下のような事柄を考慮してください。

- 表がアクセスされる方法
- 照会の作業負荷の性質
- 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

併置が主な考慮事項ではない場合、表に対する適切な区分化キーは、ノードグループ内のすべてのデータベース区画に均等にデータが分散するような区分化キーです。ノード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区分化キーによって、その表が併置されているかどうかが判別されます。表は、以下の場合に併置されていると考えられます。

- 表が同じノードグループ内にある表スペースに置かれている。
- それぞれの表の区分化キーが同じ数の列を持っている。
- 対応する列のデータ・タイプが区画互換である。

これらの特性により、同じ区分化キー値を持つ併置された表の各行は、確実に同じ区画に配置されるようになります。区画互換について詳しくは、105ページの『区画の互換性』を参照してください。表の併置について詳しくは、104ページの『表の併置』を参照してください。

区分化キーが不適切であると、データの分配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、区分化キーとして選択すべきではありません。異なる値の数は、ノードグループ内のすべてのデータベース区画にわたって行を均等に分配するのに十分な大きさでなければなりません。区分化ハッシュ・アルゴリズムを適用するためのコストは、区分化キーのサイズに比例します。区分化キーは 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、区分化キーの中に含めるべきではありません。

区分化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。

- 長形式データ・タイプ (LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、または DBCLOB) のみを含む複数区画の表を作成することはできません。
- 区分化キーの定義は変更できません。
- 区分化キーには、最も頻繁に結合される列を含める必要があります。
- 区分化キーは、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。
- どの固有キーまたは基本キーにも、すべての区分化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、区分化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 の適切な単一列区分化キーとして使用できるでしょう。

DB2_UPDATE_PART_KEY レジストリー変数が NO に設定されている場合は、表内の行ごとに区分化キー列の値を更新することはできません。この場合は、区分化キー列の値の削除と挿入のみが可能です。

ハッシュ区分化 とは、区分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、区分化キーの値に適用され、ゼロと 4095 の間の区分番号を生成します。
2. ノードグループが作成されるときに、区分化マップが作成されます。区分番号のそれぞれは、区分化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。区分化マップについて詳しくは、101ページの『区分化マップ』を参照してください。
3. 区分番号は、区分化マップへの索引として使用されます。区分化マップの中のその場所にある番号は、その行が保管されているデータベース区画の番号になります。

表の併置

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区画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*表の併置* と呼びます。

表は、同じノードグループ内に保管される場合、およびそれらの区分化キーが互換性がある場合に併置されます。両方の表を同じノードグループに置くことによって、共通の区分化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の中にあってもかまいませんが、その表スペースは同じノードグループに関連付けられていなければなりません。各区分化キーの中の対応する列のデータ・タイプは、*区画互換* でなければなりません。区画互換性の詳細については、105ページの『区画の互換性』を参照してください。

DB2 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、結合すべきデータが同じデータベース区画に配置されていることを認識する機能があります。同じデータベース区画内に配置されている場合、 DB2 では、データをデータベース区画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区画で結合または副照会を実行できます。データベース区画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。詳細については、 *管理の手引き: パフォーマンス* の『併置結合』を参照してください。

区画の互換性

区分化キーの対応する列の基本データ・タイプを比較して、*区画互換* として宣言することができます。区画互換データ・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、同じ区分化アルゴリズムによって同じ区分番号にマップされるという特性を持っています。

区画の互換性は、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。
- 区画の互換性は、 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、区画の互換性を分析するために使用されます。
- 区分化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・タイプは区分化キーとしてサポートされないため、区画の互換性はこれらには適用されません。

複製要約表

要約表 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。要約表を使って、照会のパフォーマンスを向上させることができます。照会の一部は要約表を使って解決できると DB2 が判断した場合、データベース・マネージャーは、その照会が要約表を使用するように書き換えます。

区分データベース環境では、要約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、*複製要約表* を使用できます。複製要約表には単一区画ノードグループで作成された表に基づくものもありますが、これをノードグループ内のすべてのデ

ータベース区画にわたって複製することもできます。複製要約表を作成するには、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。

要約表について詳しくは、 *管理の手引き: インプリメンテーション* の『要約表の作成』 を参照してください。

複製要約表を使用することによって、一般的には連結されていない表の間で連結を行うことができます。複製要約表は、 1 つの大きいファクト表と小さい次元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、連結によって得られるパフォーマンス上の益によって相殺されます。

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。

複製要約表についての詳細は、 *SQL 解説書* の CREATE TABLE ステートメントを参照してください。併置結合について詳しくは、 *管理の手引き: インプリメンテーション* の『併置結合』を参照してください。

表スペースの設計と選択

表スペースは、データベースとそのデータベース内に格納されている表との間に入るストレージ・モデルです。表スペースは、ノード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位置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、構成の融通性が大きくなり、整合性が向上します。

表スペースの作成と変更について、詳しくは *管理の手引き: インプリメンテーション* の『表スペースの作成』、または『表スペースの変更』を参照してください。

表スペースはノードグループの中に常駐するので、表の保持のために選択された表スペースは、その表のデータがノードグループ内の複数のデータベース区画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。 107ページの図28 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関係について示しています。

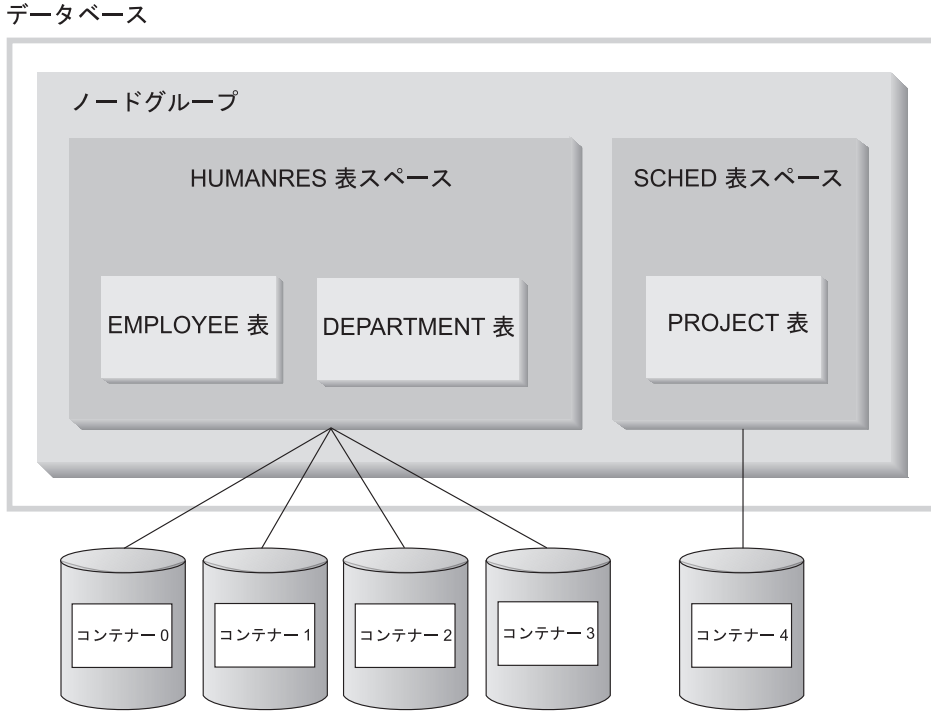

図 28. データベース内の表スペースと表

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・ロードのバランスを取ろうとします。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データを格納し始めるとは限りません。

108ページの図29 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。

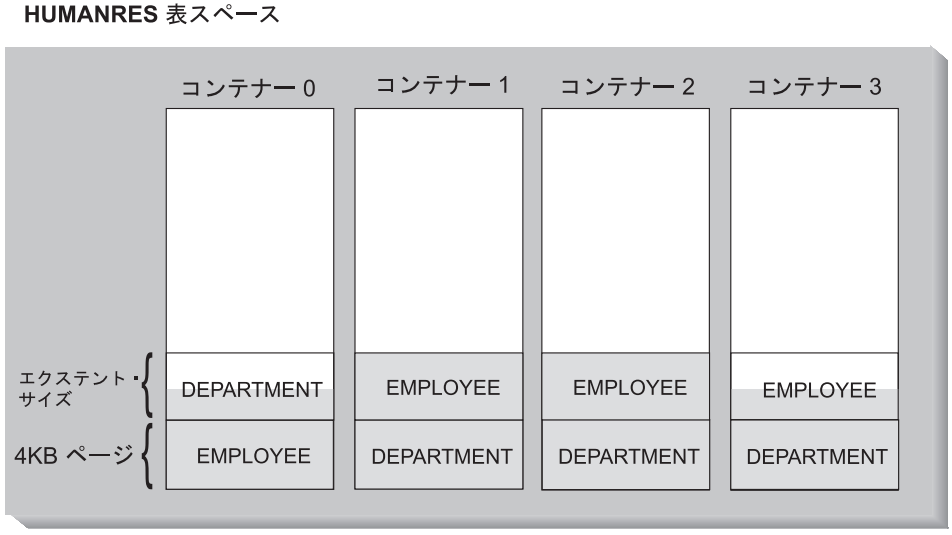

図 29. コンテナーとエクステント

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・ノードグループです。
- 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・ノードグループです。

  表スペース名は、表の作成時に指定してください。そうしないと、望みどおりの結果が得られないかもしれません。表スペース名を指定しない場合、表は、以下の規則にしたがって配置されます。ユーザー作成の表スペースが存在する場合、この表に十分な大きさの表スペースのうち、最もページ・サイズが小さいものを選択してください。存在しない場合は、USERSPACE1 のページ・サイズがこの表に十分な大きさであれば、この表スペースを使用してください。ページ・サイズが十分な大きさの表スペースが 1 つも存在しない場合、表は作成されません。

  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。基礎表には 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。) 列の数またはページ・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・ノードグループです。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、最適化プログラムが、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。

  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、一般的な環境と作業負荷に適したサイズが得られます。 121ページの『一時表スペースについての推奨事項』も参照してください。

区分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区画はそれぞれ 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。
- 113ページの『データベース管理スペース表スペース』。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。

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場所を決定し、DB2 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。SMS 表スペースは、デフォルトの表スペースです。

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。それらのファイルのリストと内容の説明については、112ページの『SMS 物理ファイル』を参照してください。

SMS 表スペースでは、オブジェクトが増大するにつれて、ファイルが一度に 1 ページずつ拡張されます。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ことができます。これによって、システムは、一度に複数ページ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。複数ページのファイル割り振りを使用可能にするためには、 **db2empfa** を実行しなければなりません。区分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区画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。 **db2empfa** についての詳細は、 *コマンド解説書* を参照してください。

SMS 表スペースの定義は、 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明示的に行う必要があります。 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。区分データベース環境では、 SMS 表スペースのノードグループに新しい区画が追加されたときに、ALTER TABLESPACE ステートメントを使用して、新しい区画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。

  ```
  コンテナー数 * (オペレーティング・システムによって
    サポートされるファイル・システムの最大サイズ)
  ```

  この式は、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。

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。詳しくは、120ページの『エクステント・サイズの選択』を参照してください。

  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 (このパラメーターについての詳細は、*管理の手引き: パフォーマンス* を参照)。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 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)、長形式フィールド (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どれか 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配分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 1) とコンテナーの番号のモジュロをとることによって、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。

SMS 表スペースで使われるファイルの詳細については、 『SMS 物理ファイル』を参照してください。

SMS 物理ファイル

SMS 表スペース・ディレクトリー・コンテナー内には次のファイルが含まれています。

ファイル名	記述
SQLTAG.NAM	各コンテナー・サブディレクトリーにはこれらのファイルのいずれか 1 つがあります。データベースに接続すると、データベース・マネージャーはこれらのファイルを用いて、データベースが完全で一貫しているか検証します。
SQLxxxxx.DAT	表ファイル。表のすべての行が格納されます。ただし、 LONG VARCHAR、 LONG VARGRAPHIC、 BLOB、 CLOB、 DBCLOB のデータは除きます。
SQLxxxxx.LF	LONG VARCHAR または LONG VARGRAPHIC のデータ (「長形式フィールド・データ」ともいう) が入るファイル。このファイルが作成されるのは、 LONG VARCHAR または LONG VARGRAPHIC の列が表内に存在する場合だけです。
SQLxxxxx.LB	BLOB、CLOB、DBCLOB のデータ (「LOB データ」ともいう) が入るファイル。これらのファイルが作成されるのは、BLOB、CLOB、または DBCLOB の列が表内に存在する場合だけです。
SQLxxxxx.LBA	SQL*xxxxx*.LB ファイルの割り振りおよびフリー・スペース情報が入るファイル。
SQLxxxxx.INX	表の索引ファイル。対応する表のすべての索引は、この 1 つのファイルに格納されます。このファイルが作成されるのは、索引が定義されている場合だけです。 **注:** 索引が除去されても、索引ファイルが削除される時点まで、スペースは索引 (.INX) ファイルから物理的には解放されません。索引ファイルは、表のすべての索引が除去 (およびコミット) されるか、表が再構成されるかした場合に削除されます。索引ファイルが削除されていない場合、スペースは除去がコミットされた時点で解放されたものとしてマークされ、将来の索引作成または索引維持のために再使用できるようになります。
SQLxxxxx.DTR	DAT ファイルの再編成のための一時データ・ファイル。表の再編成中に、 reorg (再編成) ユーティリティーは (REORG TABLE コマンド

を介して) システム一時表スペースの 1 つに表を作成します。これらの一時表スペースは、ユーザー定義の表のために使われるコンテナーとは別のコンテナーを使用するよう定義することができます。

SQLxxxxx.LFR LF ファイルの再編成のための一時データ・ファイル。表の再編成中に、 reorg (再編成) ユーティリティーは (REORG TABLE コマンドを介して) システム一時表スペースの 1 つに表を作成します。これらの一時表スペースは、ユーザー定義の表のために使われるコンテナーとは別のコンテナーを使用するよう定義することができます。

SQLxxxxx.RLB LB ファイルの再編成のための一時データ・ファイル。表の再編成中に、 reorg (再編成) ユーティリティーは (REORG TABLE コマンドを介して) システム一時表スペースの 1 つに表を作成します。これらの一時表スペースは、ユーザー定義の表のために使われるコンテナーとは別のコンテナーを使用するよう定義することができます。

SQLxxxxx.RBA LBA ファイルの再編成のための一時データ・ファイル。表の再編成中に、 reorg (再編成) ユーティリティーは (REORG TABLE コマンドを介して) システム一時表スペースの 1 つに表を作成します。これらの一時表スペースは、ユーザー定義の表のために使われるコンテナーとは別のコンテナーを使用するよう定義することができます。

注:

1. これらのファイルは、直接変更しないでください。これらのファイルにアクセスできるのは、文書 API およびその API を実現するためのツール (コマンド行プロセッサーやコントロール・センターなど) によって間接的にアクセスする場合のみです。
2. これらのファイルは移動しないでください。
3. これらのファイルは削除しないでください。
4. データベースや表スペースのバックアップ用にサポートされている唯一の手段は、**sqlubkp** (データベースのバックアップ) API です。これには、コマンド行プロセッサーおよびその API を実装するコントロール・センターが含まれます。

データベース管理スペース表スペース

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 によってスペースが管理される、限定された数の装置からなります。管理者がどの装置を使用するかを決定し、 DB2 がそれらの装置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。表スペース定義には、データ格納用の表スペースに属する装置やファイルのリストが含まれます。

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。

- 正規表データと索引データを格納する*正規* 表スペース

- 長形式フィールドや LOB データを格納する長形式 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。
- 正規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。長形式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なら、その装置の 2 000 ページは使用できなくなります。
- どのコンテナーでも 1 ページはオーバーヘッドのために確保されており、残りのページが一度に 1 エクステントずつ使用されます。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。
- 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、長形式フィールド、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、「特殊文字インターフェース」付きの論理ボリュームを使用します。
- 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実行時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

- 実際の作業負荷に 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 のバッファー・プールには入れられません。
- 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。物理装置を複数の論理装置に区分化して、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。

DMS 表スペースへのコンテナーの追加

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て、記憶容量を増やすことができます。その後、すべてのコンテナーにわたって表スペースの内容の配分バランスが再調整されます。バランスの再調整中も、表スペースへのアクセスは制限されません。複数のコンテナーを追加する必要がある場合、 1 つの ALTER TABLESPACE ステートメント内か、または同じトランザクション内で同時にそれらを追加して、データベース・マネージャーがコンテナーの再バランスを何度も行わなくても済むようにすべきです。

LIST TABLESPACE CONTAINERS または LIST TABLESPACES コマンドを使用することによって、表スペース用のコンテナーがどの程度満杯かを検査する必要があります。新しいコンテナーの追加は、既存のコンテナーがほとんど満杯か、または完全に満杯になる前に行う必要があります。すべてのコンテナーにわたる新しいスペースは、再バランスが完了するまで使用可能になりません。

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、データの分配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。

表スペース設計時の考慮事項

このセクションでは、次のようなトピックを扱います。

- 116ページの『表スペースの入出力 (I/O) についての考慮事項』
- 118ページの『バッファー・プールへの表スペースのマッピング』
- 118ページの『ノードグループへの表スペースのマッピング』
- 119ページの『表スペースへの表のマッピング』
- 120ページの『エクステント・サイズの選択』
- 121ページの『一時表スペースについての推奨事項』

- 123ページの『カタログ表スペースについての推奨事項』
- 123ページの『作業負荷についての考慮事項』
- 125ページの『SMS 表スペースか DMS 表スペースかの選択』
- 126ページの『データが RAID 装置にある場合のパフォーマンスの最適化』

表スペースの入出力 (I/O) についての考慮事項

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。

事前取り出し
: 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。ページの事前取り出しが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 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。

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場所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 ステートメントの PREFETCHSIZE パラメーターを調整することによって、事前取り出し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、事前取り出し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 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 は各装置から大ブロックを並列的に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 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、ランタイムに事前取り出し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、独自の事前取り出し方式を使用します。場合によっては、DB2 事前取り出しよりもファイル・システムの事前取り出し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 表スペースの事前取り出しの方が、装置を使用する DMS 表スペースの事前取り出し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事前取り出しが行われているためにそう見せかけ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。

事前取り出し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事前取り出し要求が存在するとします。この並列事前取り出し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、事前取り出し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。データベースによって使用されるそれぞれの実ディスクごとに、少なくとも 1 つのページ・クリーナーを定義する必要があります。これらのトピックについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

バッファー・プールへの表スペースのマッピング

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) いなければならず、 (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。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。このトピックについての詳細は、 *管理の手引き: パフォーマンス* の『データベース・バッファー・プールの管理』を参照してください。

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 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。

区分データベース環境では、データベース内のすべての区画について、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。異なる区画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。 CREATE BUFFERPOOL ステートメントの詳細については、*SQL 解説書* を参照してください。

ノードグループへの表スペースのマッピング

区分データベース環境では、各表スペースが特定のノード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ノードグループ内の各ノードに適用することができます。ノードグループはすでに存在していなければならず (CREATE NODEGROUP ステートメントを使用して定義される)、表スペースとノードグループの関連は、 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ノードグループの間の関連を変更することはできません。ノードグループ内の個々の区画に対する表スペース仕様を変更できるだけです。単一区画データベース環境では、各表スペースはデフォルト・ノード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ノードグループは 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。 CREATE NODEGROUP ステートメントの詳細については、*SQL 解説書* を参照してください。ノードグループと物理データベースの設計についての詳細は、98ページの『ノードグループの設計』を参照してください。

表スペースへの表のマッピング

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。

- 表の区分化。

  最低でも、選択する表スペースが、希望する区分化を使用するノード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 表スペースを検討すべきです。

  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。 (さらに詳しい情報については、120ページの『エクステント・サイズの選択』を参照してください。)

- 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ヒストリー・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、ヒストリー・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 および長形式フィールド・データ用、残る 1 つは正規表データ用です。これにより、データに最も適した

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 つの表スペースを分割する場合、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復元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配分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復元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復元して、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除去および再定義する必要がある場合、表を除去するよりは DMS 表スペースを除去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。

エクステント・サイズの選択

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。

  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 つの長形式フィールド・オブジェクト。表に 1 つまたは複数の LONG 列があれば、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。

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。

このため、1 つの表のためのスペースの初期割り振りは、それぞれの表オブジェクトごとに 2 エクステントになります。このため、1 つの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・サイズを指定するか、または一度に 1 ページを割り振る SMS 表スペースを使用する必要があります。

反対に、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 表スペースを使用している場合は、エクステントの割り振り追加が頻繁になされることに関連した不必要なオーバーヘッドが生じることにな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かもしれません。 (データの事前取り出しおよびエクステント・サイズとの関係については、 *管理の手引き: パフォーマンス* を参照してください。)

- 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。

一時表スペースについての推奨事項

単一の SMS 一時表スペースの定義では、大半の正規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、一般的な環境と作業負荷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作業負荷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、バッチを使って順次アクセスされます。つまり、行のバッチが挿入されたり、順次行のバッチ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。

- データベース内の正規表スペースの 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。というのも、そのような指定を行えば、一時表スペースは、正規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「インプレース」で、つまり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「インプレース」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。表の再編成についての追加情報は、*管理の手引き: パフォーマンス* を参照してください。
- 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、最適化プログラム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 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 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。
  - SMS ではディスク・スペースをオンデマンドで割り当てますが、 DMS では事前に割り当て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、普段のストレージ要件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。
– SMS コンテナーはファイル・システムによるバッファリングを利用できますが、DMS コンテナーは利用できません。

カタログ表スペースについての推奨事項

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 ～ 4 ページ) を選択すべきであり、そうでなければ、SMS 表スペースを使用すべきです。
- 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、データを保管 (またはキャッシュ) するための独自の場所があるため、 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、カタログ用にはいくぶん優れた選択です。

考慮すべき別の要因は、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復元 を使って SMS 表スペースを拡張ことも可能ですが、 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。

作業負荷についての考慮事項

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 が管理している基本的な作業負荷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) の作業負荷の特徴は、データに対してランダム・アクセスを行う必要がある、通常は小さなデータ・セットを戻す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、事前取り出しは不可能です。

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 作業負荷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。

照会作業負荷の特徴は、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、並列的事前取り出し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掛けた値に設定すべきです。これによって、DB2 は、すべてのコンテナーから並列に事前取り出しすることができます。

照会の作業負荷の代わりの方法として、ファイル・システムが独自の事前取り出し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、入出力並列化を達成するために、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ります。

作業負荷が混在している場合には、 OLTP 作業負荷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、照会の作業負荷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・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 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、

  ページ・サイズ / 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ふさわしいでしょう。
- 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。 *SQL 解説書* には制限事項が記されています。

SMS 表スペースか DMS 表スペースかの選択

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、いくつかの選択事項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、データベ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再バランスが行われます。
- 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。
  - 長形式フィールドおよび LOB データ
  - 索引
  - 正規表データ

  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 つの表に正規表データ 64 GB、索引データ 64 GB、および長形式データ 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位置の制御は、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、表を除去および再定義するよりも少ないオーバーヘッドで、表スペースを除去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 および DYNIX/ptx (IBM NUMA-Q) では、パフォーマンス重視の作業負荷用にはロー・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、長形式フィールド・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。詳細については、*管理の手引き: パフォーマンス* の『DMS 装置のパフォーマンスに関する考慮事項』を参照してください。

データが RAID 装置にある場合の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。一般に、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 (DB2_PARALLEL_IO を参照) を使って、表スペースの並列入出力を使用可能にする。
- DB2_STRIPED_CONTAINERS レジストリー変数 (127ページの DB2_STRIPED_CONTAINERSを参照) を使って、エクステント境界が表スペース内に位置合わせされるようにする。

DB2_PARALLEL_IO

DB2 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思える状況もあります。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できます。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、次のようにします。

```
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 を停止 (**db2stop**) した後、再始動 (**db2start**) する必要があります。

DB2_STRIPED_CONTAINERS

現時点では、DMS 表スペース・コンテナー (装置またはファイル) を作成するときに、1 ページのタグがコンテナーの先頭に保管されます。残りのページは DB2 がデータ記憶用に使用するためのもので、エクステント・サイズのブロックにグループ化されます。

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。しかし、1 ページのコンテナー・タグがあるため、エクステントは RAID ストライプと同列にはならず、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。

DMS 表スペースのコンテナーは、自身の (フル) エクステント内にタグを含めて作成できるようになりました。これにより、上記の問題は避けられますが、コンテナー内にオーバーヘッドによる余分のエクステントが必要になります。この方法でコンテナーを作成するには、次のように DB2 レジストリー変数 DB2_STRIPED_CONTAINERS を ″ON″ に設定してから、インスタンスを停止し、再始動する必要があります。

```
db2set DB2_STRIPED_CONTAINERS=ON
db2stop
db2start
```

(CREATE TABLESPACE または ALTER TABLESPACE ステートメントを使用して) DMS コンテナーを作成すると、フル・エクステントを使用するタグを含むコンテナーになります。既存のコンテナーは未変更のままです。

この属性を持つコンテナーの作成を停止するには、次のように変数 (デフォルト値は NULL) をリセットしてから、インスタンスを停止し、再始動します。

```
db2set DB2_STRIPED_CONTAINERS=
db2stop
db2start
```

コントロール・センターおよび LIST TABLESPACE CONTAINERS コマンドは、作成されたコンテナーがストライピングされたもの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。ストライピングされたコンテナーと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。

連合データベースの設計についての考慮事項

連合データベース (federated database: 複数のデータベースから構成されるが、単一のデータベース・イメージを提供するデータベース・システムを意味します) の設計時には、設計に関する次のトピックを考慮してください。

- スペース要件
- ネットワークの優先順位

普通、連合データベースからアクセス可能なデータは、そのデータベースには保管されていません。データ・ソース表および視点の参照はシステム・カタログ内に保管されますが、実際のデータは別のデータ・ソースにあります。このため、多くの場合、連合データベースで必要なストレージは一般的なデータベースよりも少なくて済みます。ただし、併置システムの差異やデータ・ソースでの機能不足により、照合をローカルで実行する必要がある場合には、このような一般規則が当てはまらないことがあります。その場合、表は DB2 で具体化されて処理されます。

連合システム・データの大部分は、通常、ネットワーク全体にわたる 1 つまたは複数のデータ・ソースに置かれているため、 DB2 およびネットワーク・システムに割り当てられたリソースの変更を検討してください。データベース・マネージャーそのものよりも、 DB2 システムにあるネットワークにさらに多くのリソースを割り当てることによって、パフォーマンスの向上が見られる場合があります。

第9章 分散データベースの設計

通常 DB2 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、一貫性ポイントに達します。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。

以下の部分で、さらに詳しく説明します。

分散データベースを使用するアプリケーションの作成については、 *アプリケーション開発の手引き* および *コール・レベル・インターフェースの手引きおよび解説書* を参照してください。

1 つのトランザクションで単一のデータベースを使用する場合

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。

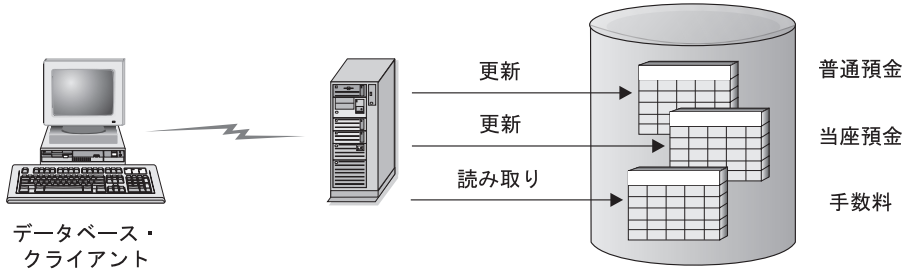

図 30. 1 つのトランザクションで単一のデータベースを使用する場合

図30 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。

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 (*管理の手引き: インプリメンテーション* の『設計のインプリメント』を参照)。
2. 物理的にリモートの場合、 *インストールおよび構成 補足* で説明しているように、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、*概説およびインストール* で説明しているように、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、*アプリケーション開発の手引き* で説明しているように、 PRECOMPILE PROGRAM コマンドで CONNECT 1 (デフォルト) を指定する。

単一のトランザクションで複数のデータベースを使用する場合

単一の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。

単一の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。

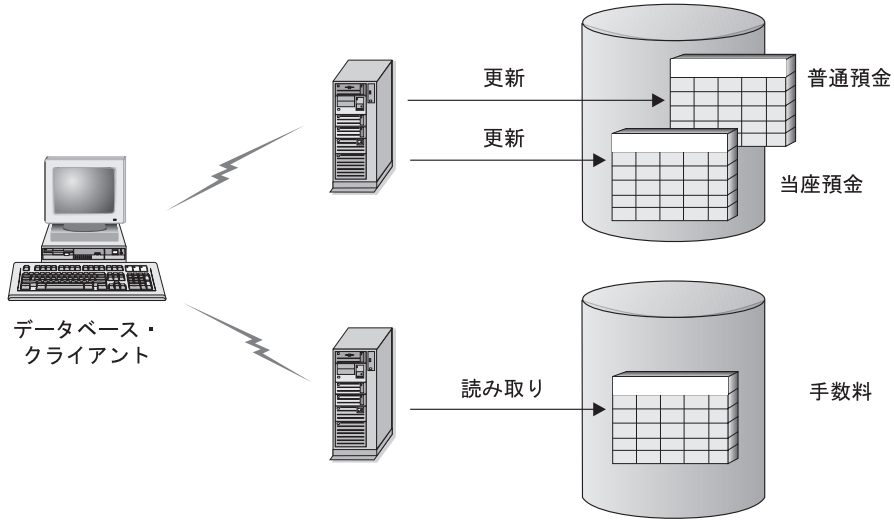

図 31. 単一のトランザクションで複数のデータベースを使用する場合

図31 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。この例は、130ページの図30に示される例と類似していますが、データベースの数と表の位置が違っています。

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 (*管理の手引き: インプリメンテーション* の『設計のインプリメント』を参照)。

2. 物理的にリモートの場合、*インストールおよび構成 補足* で説明しているように、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、*概説およびインストール* で説明しているように、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. *アプリケーション開発の手引き* で説明しているように、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 AS/400 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 コネクトが必要です。セットアップについては、いずれかの DB2 コネクト *概説およびインストール* を参照してください。 DB2 コネクトの使用については、*DB2 コネクト 使用者の手引き* を参照してください。

複数の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。

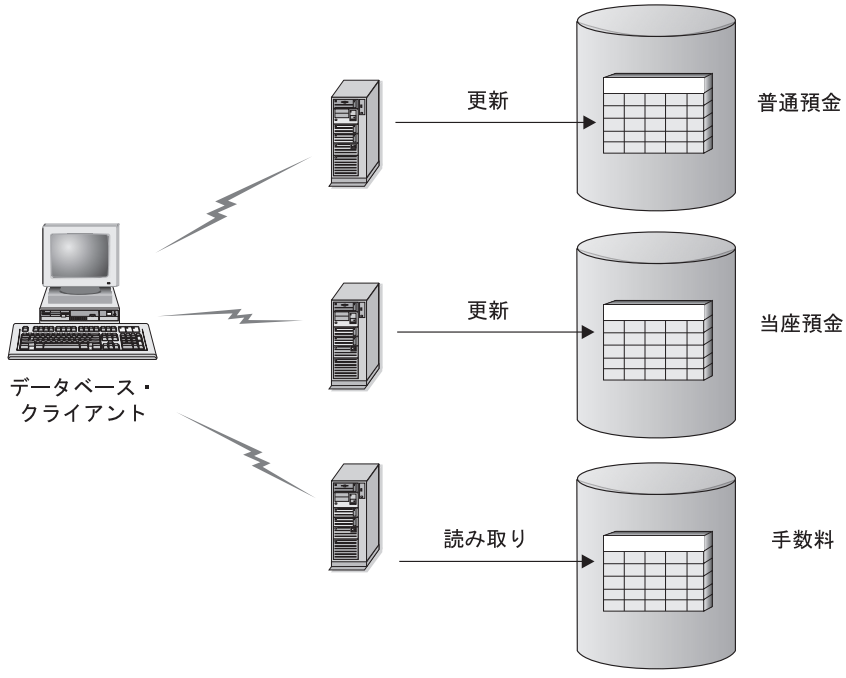

図 32. 単一のトランザクションで複数のデータベースを更新する

図32 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。

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. 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 (*管理の手引き: インプリメンテーション* の『設計のインプリメント』を参照)。
   b. 物理的にリモートの場合、*インストールおよび構成 補足* で説明しているように、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、*概説およびインストール* で説明しているように、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. *アプリケーション開発の手引き* で説明しているように、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) および 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 トランザクション・マネージャー (TM) を構成する (『DB2 トランザクション・マネージャーの使用方法』を参照)。

2. トランザクション・マネージャーを使用しない方法:
   a. 該当するデータベースの中に必要な表を作成する (*管理の手引き: インプリメンテーション* の『設計のインプリメント』を参照)。
   b. 物理的にリモートの場合、*インストールおよび構成 補足* で説明しているように、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、*概説およびインストール* で説明しているように、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. *アプリケーション開発の手引き* で説明しているように、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。

DB2 トランザクション・マネージャーの使用方法

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。

IBM TXSeries、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、統合に関する説明について 147 ページの『第10章 トランザクション・マネージャーの設計』を参照してください。

UNIX ベースのシステム、Windows オペレーティング・システム、または OS/2 版の DB2 UDB 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 と DB2 (OS/390 版) だけである場合は、単純な構成を使用できます。

TCP/IP 接続性を使用した DB2 UDB および DB2 (OS/390 版): 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 (OS/390 版) 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・サーバーが、 UNIX ベースのシステム、Windows オペレーティング・システム、OS/2、または OS/390 版の UDB DB2 のみである。
- DB2 コネクト同期点マネージャー (SPM) が構成されていない。

DB2 コネクト同期点マネージャーは DB2 インスタンスの作成時に自動的に構成されるもので、以下の場合に必要です。

– マルチサイト更新に、 SNA 接続性がホストまたは AS/400 データベース・サーバーで使用されている。

– XA 準拠のトランザクション・マネージャー (IBM TXSeries CICS など) が、 2 フェーズ・コミットを調整している。

これは、ホストまたは AS/400 データベース・サーバーでの SNA 接続性と TCP/IP 接続性の両方に当てはまります。詳細については、147ページの『第10章 トランザクション・マネージャーの設計』を参照してください。 DB2 コネクト同期点マネージャーが必要でない環境は、 DB2 コネクト・サーバーでコマンド `db2 update dbm cfg using spm_name NULL` を発行してこれをオフにすることができます。その後、DB2 を停止して再始動してください。

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。この構成パラメーターについて詳しくは、 *管理の手引き: パフォーマンス*の『DB2 の構成』を参照してください。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 (UNIX ベースのシステム版、Windows オペレーティング・システム版、または OS/2 版) のデータベース
  – DB2 (OS/390 版) バージョン 5 以降のデータベース

  このデータベース・サーバーをトランザクション・マネージャー・データベースとして使用することをお勧めします。一般的に、OS/390 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のみ、つまり、以下のような場合のみにしてください。
  – 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。
  – DCE ディレクトリー・サービスを使用して、データベースのカタログ作成およびデータベースへのアクセス管理を行っている。

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。

その他の環境: 以下のような環境では、マルチサイト更新の構成ステップは複雑になります。

- リモート・データベースとの通信に使用されているのが TCP/IP だけではない (たとえば、 NETBIOS が使用されている)
- DB2 (MVS 版) バージョン 3 または バージョン 4、DB2 (AS/400 版)、または DB2 (VM および VSE 版) にアクセスする
- SNA を使用して DB2 (OS/390 版) にアクセスする
- ホストまたは AS/400 データベース・サーバーにアクセスするのに、 DB2 コネクト同期点マネージャーが使われる

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。この構成パラメーターについて詳しくは、 *管理の手引き: パフォーマンス*の『DB2 の構成』を参照してください。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、 DB2 UDB (UNIX ベースのシステム版、Windows オペレーティング・システム版、または OS/2 版) のデータベースを使用できます。
- 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のみ、つまり、以下のような場合のみにしてください。

  – 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。
  – DCE ディレクトリー・サービスを使用して、データベースのカタログ作成およびデータベースへのアクセス管理を行っている。

  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。

構成についてのその他の考慮事項

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。これらのパラメーターの設定について詳しくは、*DB2 コネクト 使用者の手引き* を参照してください。

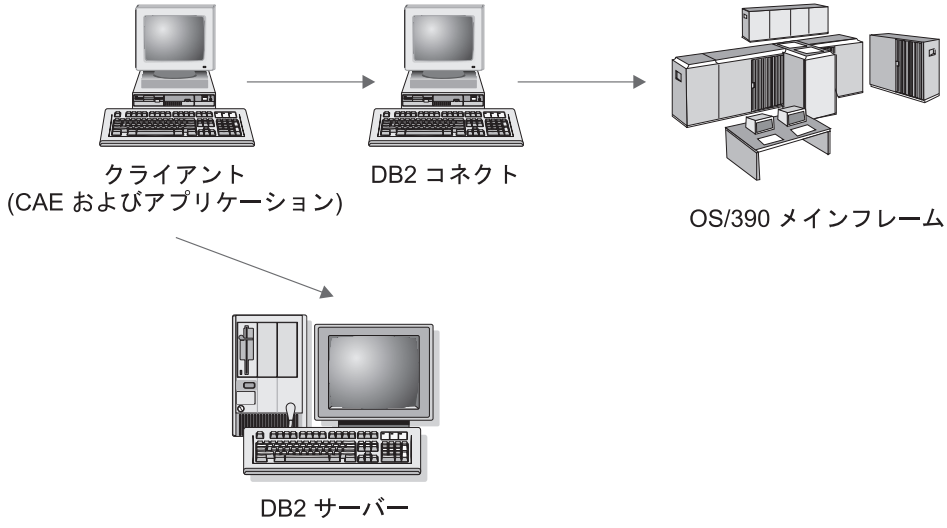

図 33. 構成についての考慮事項

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 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 コネクト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固有の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 サーバー・データベース・マネージャー、および DB2 コネクト (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、活動状態の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。未確定トランザクションについて詳しくは、143ページの『2 フェーズ・コミット時の問題をリカバリーする』を参照してください。

- *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除去されるときに *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発見的手法の操作によって、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。

これらの構成パラメーターについての詳細は、*管理の手引き: パフォーマンス* を参照してください。

マルチサイト更新で LAN ベースの DB2 ユニバーサル・データベース・サーバーにアクセスするホストまたは AS/400 アプリケーション

DB2 ユニバーサル・データベースは、TCP/IP 接続を使用した、ホストまたは AS/400 データベース・クライアントからのマルチサイト更新をサポートしません。このような状況では、SNA (システム・ネットワーク体系) 接続がサポートされています。マルチサイト更新には DB2 同期点マネージャーが必要です。ここでは DB2 コネクトは使用されません。

ホストまたは AS/400 データベース・クライアントからアクセスされるデータベース・サーバーは、 DB2 同期点マネージャーのあるワークステーションから見てローカルで

ある必要はありません。ホストまたは AS/400 データベース・クライアントは、 DB2 同期点マネージャー・ワークステーションを暫定ゲートウェイとして使用して、 DB2 UDB サーバーに接続できます。これにより、DB2 同期点マネージャー・ワークステーションを分離して、セキュリティーされた環境に置くことができます。このとき、実際の編成では DB2 UDB サーバーがリモートになっています。また、これによって DB2 共通サーバー バージョン 2 データベースを、ホストまたは AS/400 データベース・クライアントに起因するマルチサイト更新に参加させることができます。

手順は以下のとおりです。

- ホストまたは AS/400 アプリケーションから直接アクセスされるワークステーション上で、以下のようにします。
  1. ホストまたは AS/400 データベース・クライアントを使ったマルチサイト更新をサポートできるようにするために、 DB2 ユニバーサル・データベース エンタープライズ・エディションまたはエンタープライズ拡張 (EE) エディションをインストールします。
  2. 同じシステムにデータベース・インスタンスを作成します。たとえば、デフォルト・インスタンス DB2 を使用するか、または以下のコマンドを使用して新しいインスタンスを作成できます。

     ```
     db2icrt myinstance
     ```

  3. 必要に応じてライセンス情報を提供します。
  4. レジストリー値 DB2COMM に値 APPC が含まれていることを確認します。
  5. 必要に応じて SNA 通信を構成します。サポートされている IBM SNA 製品を使用している場合、 DB2 同期点マネージャーに必要な SNA プロファイルは、データベース・マネージャー構成パラメーター *spm_name* の値に基づいて自動的に作成されます。サポートされているその他の SNA スタックについては、手動で構成する必要があります。詳しくは、*インストールおよび構成 補足* を参照してください。
  6. データベース・マネージャー構成パラメーター *spm_name* に指定する値を決定します。このパラメーターは、DB2 インスタンス作成時に、マシンの TCP/IP ホスト名の派生名としてあらかじめ構成されています。この値が適切で、環境の中で固有のものであれば、変更しないでください。
  7. 必要に応じて、 UPDATE DATABASE MANAGER CONFIGURATION コマンドを使用して、 DB2 ユニバーサル・データベース・サーバー上で *spm_name* を更新します。
  8. この DB2 ワークステーションがリモート DB2 UDB サーバーに接続するために必要な通信が存在すれば、それを構成します。
  9. リモート DB2 UDB サーバーがこの DB2 ワークステーションに接続するために必要な通信を構成します。
  10. SPM を初めて開始するために、 DB2 ユニバーサル・データベース・サーバーでデータベース・マネージャーを停止および再始動します。

この DB2 ワークステーションからリモート DB2 UDB サーバーに接続できるはずです。

- ホストまたは AS/400 データ・クライアントからアクセスされるそれぞれのリモート DB2 UDB サーバー上で、次のようにします。
  1. DB2 同期点マネージャーのあるリモート DB2 UDB ワークステーションがこの DB2 UDB サーバーに接続するために必要な通信を構成します。
  2. データベース・マネージャーを停止および再始動します。

2 フェーズ・コミット・プロセスについて

141ページの図34 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。

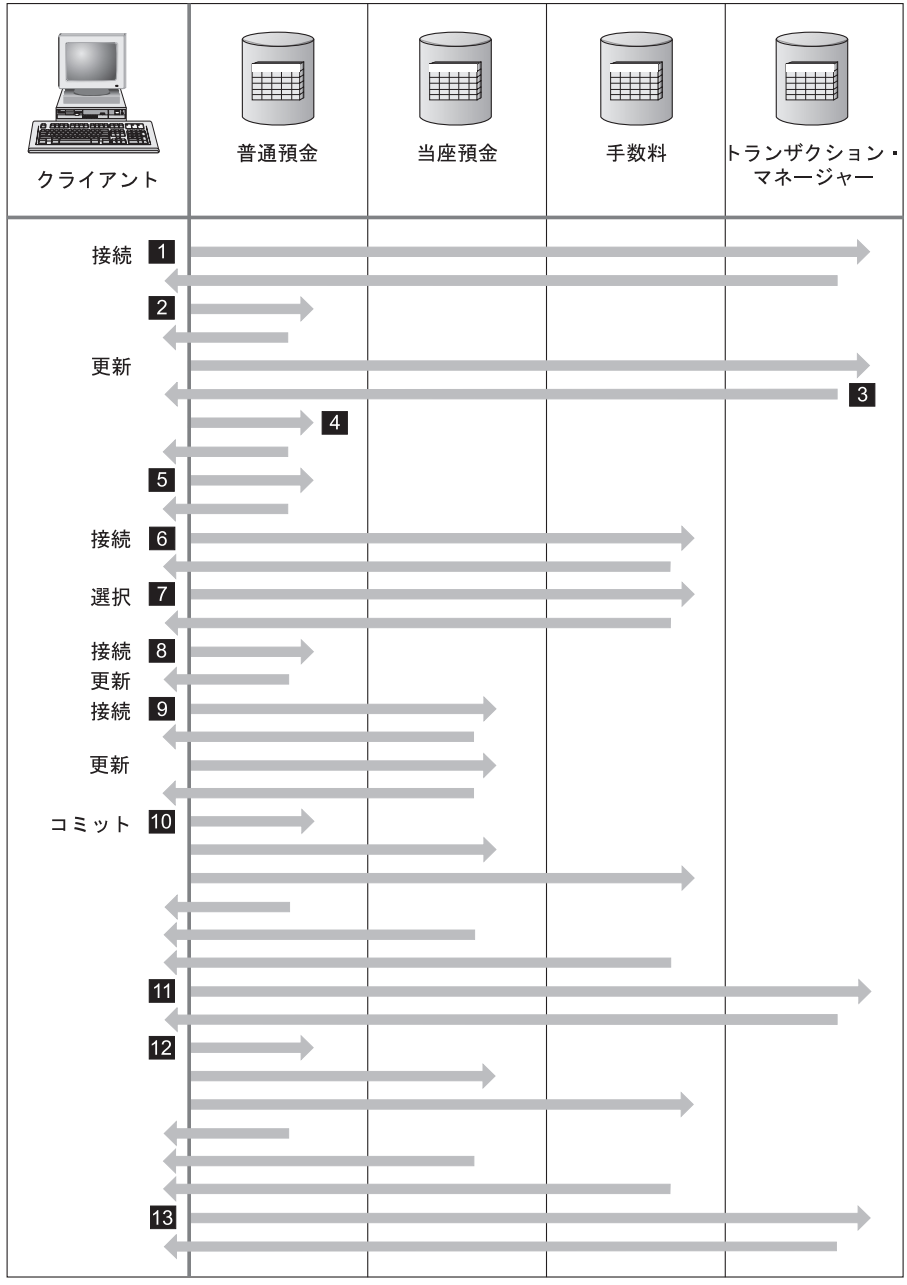

図 34. 複数のデータベースの更新

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 (詳細については、*アプリケーション開発の手引き* を参照してください)。また、DB2 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 (詳細については、*コール・レベル・インターフェースの手引きおよび解説書* を参照してください)。

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。 (SQLCA については、*アプリケーション開発の手引き* および *SQL 解説書* で説明されています。)

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。

2 フェーズ・コミット時の問題をリカバリーする

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おり、この点については、140ページの『2 フェーズ・コミット・プロセスについて』 (**10** 、 **11** 、および **12**) に示されています。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 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間隔は、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 (*管理の手引き: パフォーマンス* の『DB2 の構成』を参照)。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行います。

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、「発見的手法の決定」と呼ばれることもあります。未確定トランザクションの手動リカバリーについて詳しくは、 159ページの『発見的手法の決定』を参照してください。

AUTORESTART=OFF の場合の未確定トランザクションの再同期化

DB2 ユニバーサル・データベースの 2 フェーズ・コミット環境における構成の考慮事項については、 137ページの『構成についてのその他の考慮事項』を参照してください。

特に、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除去され、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除去します。

第10章 トランザクション・マネージャーの設計

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、 132ページの『複数のデータベースの更新』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 TXSeries CICS、IBM TXSeries Encina、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。

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/library/`

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)』
- 150ページの『トランザクション・マネージャー (TM)』
- 151ページの『リソース・マネージャー (RM)』

図35 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。

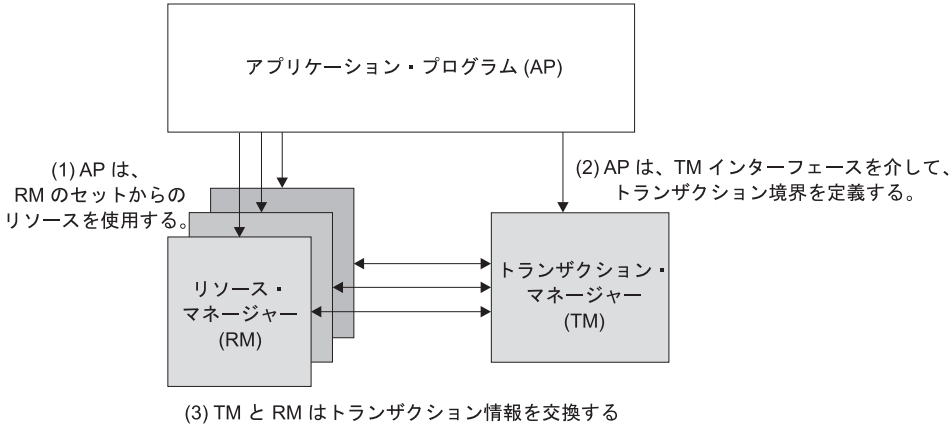

図 35. X/Open 分散トランザクション処理 (DTP) のモデル

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、プログラミング論理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 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 事前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当て、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。詳しくは、162 ページの『セキュリティーについての考慮事項』を参照してください。

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。トランザクション分岐 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 1 つまたは複数のアプリケーション・プロセスによって複数の RM がアクセスされる場合は、 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分岐が存在するようになる可能性があります。

  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分岐があります。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、トランザクション分岐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 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。 132ページの『複数のデータベースの更新』を参照してください。)

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分岐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、トランザクション分岐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。トランザクション分岐 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の際に、システム始動時の*再同期化* 操作 (*resync* ともいう) を行ったり、または必要に応じて、管理者が*発見的手法* の操作 (*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、この始動プロシージャーの一環として再同期化を実行し、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 に通知すべきかを判断します。管理者の発見的手法の操作により、 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 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 インターフェースに必要なものとして、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。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。データベース・マネージャーでサポートされている XA 関数については、164ページの『サポートされている XA 関数』を参照してください。

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 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 2 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 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。 X/Open XA インターフェース・プログラミングの考慮事項については、 *アプリケーション開発の手引き* を参照してください。

XA インターフェースは一定ですが、XA 準拠の各 TM では、 RM が製品特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 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。一般的な TP モニターに関する組み込み情報は、 167 ページの『DB2 UDB を使用するよう XA トランザクション・マネージャーを構成する』を参照してください。

データベースをリソース・マネージャーとして設定する

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。 DB2 の xa_open ストリングの形式については、『xa_open および xa_close ストリングの使用法』を参照してください。

xa_open および xa_close ストリングの使用法

データベース・マネージャーの xa_open ストリングには、2 つの形式があります。 1 つは DB2 バージョン 7 で新しく使用される形式です。 2 番目の形式は DB2 の以前のバージョンで使われるもので、バックレベル互換性のために保持されています。新しく実装する際には新しい形式を使用すべきであり、以前に実装したものは、可能であれば新しい形式にマイグレーションしてください。 DB2 の将来のバージョンでは、古い形式の xa_open ストリングをサポートしない可能性があります。以前の xa_open ストリング形式については、158ページの『以前のバージョンの DB2 の xa_open ストリング形式』を参照してください。

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。

DB2 バージョン 7 での新しい xa_open ストリング形式

以下の xa_open ストリング形式は DB2 バージョン 7 で新しく使用されるものです。

```
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 22. parm_id の有効値

パラメーター名	値	必須かどうか	大文字小文字の区別	デフォルト値
DB	データベース別名	あり	なし	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	なし	あり	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。				

表 22. parm_id の有効値 (続き)

パラメーター名	値	必須かどうか	大文字小文字の区別	デフォルト値
PWD	パスワード	なし	あり	なし
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	なし	なし	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、154ページの『TPM 値および TP_MON_NAME 値』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 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。	なし	あり	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 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 TPM モニターがこの値を使用することもできます。				
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。	なし	なし	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 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。また、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。	なし	なし	F

表 22. *parm_id* の有効値 (続き)

パラメーター名	値	必須かどうか	大文字小文字の区別	デフォルト値
トランザクション分岐を中断する 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。このような状況では、新しいトランザクションがカーソルを継承しないようにするために、カーソルを閉じる必要があります。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 がオンであれば、開いたカーソルはいずれも閉じられませんが、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することはできません。中断されたトランザクションの再開だけが可能です。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。また、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。	なし	なし	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_CURSOR がオンであれば、トランザクションのコミット後も次のコミットまでカーソルを保持します。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。また、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 に示すために使われます。 *tp_mon_name* 値は DB2 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 23. 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 (Windows の場合) =/usr/lpp/encina/lib/libEncServer (UNIX ベースのシステムの場合) HOLD_CURSOR=T CHAIN_END=T SUSPEND_CURSOR=F
ENCINA	IBM TxSeries Encina Monitor	AXLIB=libEncServer (Windows の場合) =/usr/lpp/encina/lib/libEncServer (UNIX ベースのシステムの場合) HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
MQ	IBM MQSeries	AXLIB=mqmax (Windows の場合) =/usr/mqm/lib/libmqmax.a (AIX の場合) =/opt/mqm/lib/libmqmax.a (Solaris の場合) HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
CB	IBM Component Broker	AXLIB=somtrx1i (Windows の場合) =libsomtrx1 (UNIX ベースのシステムの場合) 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 を構成する必要はありません。 MTS は DB2 の ODBC ドライバーによって自動的に検出されます。

表 23. TPM および tp_mon_name の有効値 (続き)

TPM 値	TP モニター製品	内部設定
JTA	Java Transaction API	IBM WebSphere などの Enterprise Java Server (EJS) 用に DB2 を構成する必要はありません。 DB2 の JDBC ドライバーは、この環境を自動的に検出します。

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 コネクトのバージョン 7 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 コネクトのバージョン 7 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 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 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 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
  ```

以前のバージョンの DB2 の xa_open ストリング形式

以前のバージョンの DB2 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 (152ページの『DB2 バージョン 7 での新しい xa_open ストリング形式』を参照) にマイグレーションしてください。

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のでない限り、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のに使用します。

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。

ホストまたは AS/400 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 AS/400 データベース・サーバーを更新することができます。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 コネクト・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 コネクト EE 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_logicagents* 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 コネクト EE コンセントレーターがサポートするためには、 DB2 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。この環境で使用できる SQL ステートメントについては、*アプリケーション開発の手引き* を参照してください。コンセントレーターについては、*DB2 コネクト 使用者の手引き* を参照してください。

ホストまたは AS/400 データベース・サーバーを更新する場合は、 DB2 同期点マネージャー (SPM) が構成されている DB2 コネクトが必要です。詳しくは、*概説およびインストール* を参照してください。

データベース接続の考慮事項

このセクションでは、次のトピックについて説明します。

- 『RELEASE ステートメント』
- 『区分データベースにアクセスするトランザクション』

RELEASE ステートメント

RELEASE ステートメントを用いてデータベースへの接続を解放する場合は、 SET CONNECTION ではなく CONNECT ステートメントを用いてそのデータベースに再接続してください。

区分データベースにアクセスするトランザクション

区分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区画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、データベース区画 (調整プログラム・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区画に接続することや、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区画を選択することができます。

区分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* データベース区画から、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)、同じデータベース区画を使用しなければならないということです。

区分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。

発見的手法の決定

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、140ページの『2 フェーズ・コミット・プロセスについて』で説明されているように、DB2 トランザクション・マネージャーが使用するのと同様な 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 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・データベースの機能の代わりに、 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 トランザクション・マネージャーについて論じられたエラー (143ページの『2 フェーズ・コミット時の問題をリカバリーする』参照) 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 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、「発見的手法の決定」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、発見的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「忘れる」こともできます。 UNIX ベースのシステム、Windows オペレーティング・システム、または OS/2 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。このコマンドまたは関連する管理 API については、 *コマンド解説書* または *管理 API 解説書* を参照してください。

ホストまたは AS/400 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 AS/400 サーバーから未確定情報を直接入手する。

  DB2 (OS/390 版) 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。発見的手法を使用するには、RECOVER コマンドを使用します。 DB2 (OS/400 版) 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 AS/400 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 コネクト・サーバーから、未確定情報を取得する。

  DB2 コネクト・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 インスタンスに接続することによって、 DB2 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発見的手法を使用するために、 LIST DRDA INDOUBT TRANSACTIONS WITH PROMPTING コマンドを発行します。

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コミットまたはロールバックを行い、別の参加データベースでは正反対の処置を取るならば、データ保全の問題が生じかねません。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、次いで所定時間内にデータベースをリカバリーするか、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、発見的手法の操作が必要です。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。

発見的手法の操作を行う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操作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発見的手法でコミットするか、または発見的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発見的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発見的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発見的手法でロールバックしてください。

– 他のリソース・マネージャーが全く利用不能であれば、そのトランザクションを発見的手法でロールバックしてください。

発見的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) を除き、発見的手法の forget 関数は実行しないでください。発見的手法の 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。

セキュリティーについての考慮事項

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確認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視点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 カタログ視点から、その表または視点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。

構成についての考慮事項

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。詳細については、*概説およびインストール* を参照してください。

- *tm_database*

  DB2 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、活動状態の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。未確定トランザクションについて詳しくは、143ページの『2 フェーズ・コミット時の問題をリカバリーする』を参照してください。

  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除去されるときに *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発見的手法の操作によって、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。

サポートされている XA 関数

DB2 ユニバーサル・データベースは、 *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 ユニバーサル・データベースは、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。これら 2 つの方法についての詳細は、151ページの『リソース・マネージャー (RM)』 を参照してください。

- 関連のマイグレーション

  DB2 ユニバーサル・データベースは、制御スレッド間のトランザクション・マイグレーションをサポートしていません。

xa_open ストリングおよび xa_close ストリングの使用法については、 152ページの『xa_open および xa_close ストリングの使用法』を参照してください。

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 (AIX 版)。
flags	TMREGISTER | TMNOMIGRATE DB2 ユニバーサル・データベースが動的登録を使用し、 TM は関連のマイグレーション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 ユニバーサル・データベース 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 非空ポインター、フラグ、およびバージョン番号が含まれます。

UNIX ベースのシステムおよび OS/2: 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えます。

  ```
  #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** を呼び出すことによって。

  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 (UNIX ベースのシステム) または db2api.lib (OS/2) 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えます。

  ```
  #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** を呼び出すことによって。

  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 ;
   }
```

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。基本障害保守ログの詳細について、*問題判別の手引き* を参照してください。

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 (トランザクションまたは発見的手法のエラー) 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。
- 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しなかった。
  - 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

以下の例は、 (xa_open ストリングがないために) AIX で生成される xa_open エラーのエラー・ログを示しています。

```
Tue Apr  4 15:59:08 1995
toop pid(83378) process (xatest) XA DTP Support      sqlxa_open Probe:101
DIA4701E Database "" could not be opened for distributed transaction
processing.
String Title : XA Interface SQLCA  pid(83378)
SQLCODE = -998  REASON CODE: 4  SUBCODE: 1
Dump File : /u/toop/diagnostics/83378.dmp Data : SQLCA
```

DB2 UDB を使用するよう XA トランザクション・マネージャーを構成する

以下のセクションでは、 DB2 をリソース・マネージャーとして使用するように特定の製品を構成する方法を説明します。以下のいずれかを使用することができます。

- 『IBM TXSeries CICS の構成』
- 『IBM TXSeries Encina の構成』
- 170ページの『BEA Tuxedo の構成』
- 172ページの『Microsoft Transaction Server の構成』

IBM TXSeries CICS の構成

DB2 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 Administration Guide* 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 AS/400 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 ユニバーサル・データベース・サーバーの統合に必要とされるもの、および (DB2 コネクトを使ってアクセスする場合) DB2 (MVS 版)、DB2 (OS/390 版)、 DB2 (AS/400 版)、または DB2 (VSE および VM 版) 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 AS/400 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 の構成

DB2 を構成するには、次のようにします。

1. すべてのデータベース名を DB2 データベース・ディレクトリーで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・ディレクトリー入り口も定義する必要があります。構成を行うには、クライアント構成アシスタント (CCA) または DB2 コマンド行プロセッサー (CLP) を使用できます。たとえば、次のようにします。

   ```
   DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER
   DB2 CATALOG TCPIP NODE host1 REMOTE hostname1 SERVER svcname1
   ```

2. DB2 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 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 全画面インターフェース、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、次のようにします。

```
monadmin create rm inventdb -open "db=inventdb,uid=user1,pwd=password1"
```

各 DB2 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、rm 名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 の xa_open ストリングについては、152ページの『データベースをリソース・マネージャーとして設定する』を参照してください。

DB2 は、xa_close ストリングを使用しません。

制御の取り決めのスレッドは、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 UDB は、デフォルトの TMXA_SERIALIZE_ALL_OPERATIONS をサポートしていますが、この場合、トランザクションが完了した後にしかスレッドを再使用できません。

DB2 (OS/390 版)、DB2 (MVS 版)、DB2 (AS/400 版)、または DB2 (VSE および VM 版) にアクセスする場合は、 DB2 同期点マネージャーを使用する必要があります。構成の指示については、*DB2 コネクト エンタープライズ・エディション (OS/2 および Windows 版) 概説およびインストール* を参照してください。

Encina アプリケーションからの DB2 データベースの参照

Encina アプリケーションから DB2 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、次のようにします。

   ```
   rc = mon_SetSchedulingPolicy (MON_EXCLUSIVE)
   ```

   DB2 (DB2 ユニバーサル・データベース、ホスト、または AS/400 データベース・サーバー) については、デフォルト設定の MON_EXCLUSIVE を使用する必要があります。そのようにすることによって、次のことが保証されます。

   - トランザクションの存続時間中は、アプリケーション・プロセスがロックされる。
   - アプリケーションが単一スレッドで動作する。

   注: ODBC または DB2 コール・レベル・インターフェースを使用している場合は、マルチスレッド・サポートを使用不可にしなければなりません。これは、CLI 構成パラメーター DISABLEMULTITHREAD = 1 (マルチスレッドを使用不可にする) を設定することによって行えます。 DB2 ユニバーサル・データベースのデフォルトは、 DISABLEMULTITHREAD = 0 (マルチスレッドを使用可能にする) です。詳しくは、*コール・レベル・インターフェースの手引きおよび解説書* を参照してください。

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、次のようにします。

```
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 ユニバーサル・データベースの XA スイッチは、db2xa_switch で、これは DB2 クライアント・ライブラリー (Windows オペレーティング・システムおよび OS/2 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。

注: TM-XA インターフェースを介した DB2 のトランザクション処理に Encina を使用する際には、 Encina のネストされたトランザクションが、現在 DB2 XA インターフェースではサポートされていないことに注意してください。可能であれば、このようなトランザクションは使用しないでください。使用を避けられない場合は、SQL の作業を必ず Encina トランザクション・ファミリーの 1 メンバーだけで行ってください。

BEA Tuxedo の構成

DB2 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 アプリケーション開発クライアントも必要です。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 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 データベースに接続できることを確認します。
3. TUXEDOの値でデータベース・マネージャー構成パラメーター *tp_mon_name* を更新します。
4. DB2 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 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 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 インスタンス・ディレクトリーです。

5. 次のようにして、DB2 の Tuxedo トランザクション・モニターを構築する。
   - 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 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、次のようなエントリーを追加します。

```
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 ユーザー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 (OS/390 版)、DB2 (AS/400 版)、または DB2 (VM および VSE 版) 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 コネクト XA コンセントレーターが必要です。構成の詳細と制限事項については、*DB2 コネクト 使用者の手引き* を参照してください。
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 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。

Microsoft Transaction Server の構成

DB2 UDB V5.2 以降のバージョンを、 Microsoft Transaction Server (MTS) バージョン 2.0 に完全に統合できます。 Windows 32 ビット・オペレーティング・システム上で MTS の下で実行されているアプリケーションは、DB2 UDB、ホスト、および AS/400 サーバーと他の MTS 準拠のリソース・マネージャーとの 2 フェーズ・コミットを調整するために MTS を使用できます。

注: 付加的な技術情報が IBM の Web サイトで提供されており、 DB2 MTS サポートのインストールと構成について詳しく知ることができます。 URL を http://www.ibm.com/software/data/db2/library/ に設定し、キーワード ″MTS″ で DB2 Universal Database の ″Technote″ を検索します。

DB2 での MTS サポートの使用可能化

Microsoft Transaction Server サポートは、自動的に使用可能になります。 *tp_mon_name* データベース・マネージャー構成パラメーターを MTS に設定できるものの、これは必要ではなく、無視されます。

MTS ソフトウェア前提条件

MTS サポートでは、DB2 クライアント・アプリケーション (CAE) バージョン 5.2 以降が必要で、 MTS は Hotfix 0772 のバージョン 2.0 以降でなければなりません。

DB2 ODBC ドライバーを Windows 32 ビット・オペレーティング・システムにインストールすると、次のように、新しいキーワードがレジストリーに自動的に追加されます。

```
HKEY_LOCAL_MACHINE¥software¥ODBC¥odbcinit.ini¥IBM DB2 ODBC Driver:
Keyword Value Name: CPTimeout
Data Type: REG_SZ
Value: 60
```

インストールと構成

以下に、MTS のインストールと構成についての考慮事項を要約します。 DB2 の MTS サポートを使用するには、以下のようにする必要があります。

1. MTS および DB2 クライアントを MTS アプリケーションが実行されている同じマシンにインストールします。
2. マルチサイト更新でホストまたは AS/400 データベース・サーバーが関係する場合:
   a. ご使用のローカル・マシンまたはリモート・マシンのいずれかに、 DB2 コネクト エンタープライズ・エディション (EE) をインストールします。 DB2 コネクト EE を使用すると、ホストまたは AS/400 データベース・サーバーは、マルチサイト更新トランザクションに参加できるようになります。
   b. DB2 コネクト EE サーバーがマルチサイト更新で使用可能になっていることを確認してください。マルチサイト更新で DB2 コネクトを使用可能にすることについての詳細は、ご使用のプラットフォームの DB2 コネクト エンタープライズ・エディション 概説およびインストールを参照してください。

DB2 CLI/ODBC アプリケーションを実行している場合には、以下の構成キーワード (db2cli.ini ファイルに設定されている) は、デフォルト値から変更してはなりません。

- CONNECTYPE キーワード (デフォルト 1)
- MULTICONNECT キーワード (デフォルト 1)
- DISABLEMULTITHREAD キーワード (デフォルト 0)
- CONNECTIONPOOLING キーワード (デフォルト 0)
- KEEPCONNECTION キーワード (デフォルト 0)

MTS サポートを利用するために作成される DB2 CLI アプリケーションは、上記のキーワードに対応する属性値を変更してはなりません。加えて、アプリケーションは以下の属性のデフォルト値を変更してはなりません。

- SQL_ATTR_CONNECT_TYPE 属性 (デフォルト SQL_CONCURRENT_TRANS)
- SQL_ATTR_CONNECTON_POOLING 属性 (デフォルト SQL_CP_OFF)

インストールの検査

1. DB2 クライアントと DB2 コネクト EE を、 DB2 UDB、ホスト、または AS/400 サーバーにアクセスするよう構成します。
2. DB2 CAE マシンから DB2 UDB データベース・サーバーへの接続を検査します。
3. DB2 コネクト・マシンからホストまたは AS/400 データベース・サーバーへの接続を DB2 CLP で検査して、いくつかの照会を出します。

4. DB2 CAE マシンから DB2 コネクト・ゲートウェイを介してホストまたは AS/400 データベース・サーバーに至る接続を検査して、いくつかの照会を出します。

サポートされている DB2 データベース・サーバー

MTS 調整トランザクションを使用したマルチサイト更新用に、以下のサーバーがサポートされています。

- DB2 ユニバーサル・データベース エンタープライズ・エディション バージョン 6 以降
- DB2 ユニバーサル・データベース エンタープライズ拡張エディション バージョン 6 以降
- DB2 (OS/390 版)
- DB2 (MVS 版)
- DB2 (AS/400 版) (ただし、以前の名称は DB2 AS/400 用)
- DB2 (VSE および VM 版)

MTS トランザクション・タイムアウトと DB2 接続動作

MTS Explorer ツールで、トランザクション・タイムアウト値を設定できます。詳しくは、オンラインの *MTS Administrator Guide* を参照してください。

トランザクションにトランザクション・タイムアウト値 (デフォルト値は 60 秒) 以上の時間がかかる場合は、 MTS は関係するすべてのリソース・マネージャーに非同期に打ち切りを出し、トランザクション全体が打ち切られます。

DB2 サーバーへの接続については、打ち切りは DB2 ロールバック要求に変換されます。他のすべてのデータベース要求と同じように、データベース・サーバー上のデータの保全性を保証するために、ロールバック要求は接続に基づいてシリアル化されます。

その結果、次のようになります。

- 接続がアイドルの場合は、ロールバックは即時に実行されます。
- 長期間実行される SQL が処理している場合は、ロールバック要求はその SQL ステートメントが完了するのを待ちます。

接続のプール

接続のプールによって、アプリケーションは接続のプールから接続を使用できるので、接続を使用するたびに再確立する必要はありません。接続が作成されてプールに置かれると、アプリケーションは完全な接続処理を実行せずに、その接続を再使用できます。接続は、アプリケーションが ODBC データ・ソースから切断されるときにプールされ、属性が同じである新しい接続に与えられます。

接続のプールは、ODBC ドライバー・マネージャー 2.x 以降の機能です。 MTS と共に出荷された最新の ODBC ドライバー・マネージャー (バージョン 3.5) では、接続のプールの構成が変更され、トランザクション MTS COM オブジェクトの ODBC 接続

の動作が新しくなりました (176ページの『同一トランザクションに参加している COM オブジェクト間での ODBC 接続の再使用』を参照)。

ODBC ドライバー・マネージャー 3.5 では、接続のプールを活動化させる前に、ODBC ドライバーは新しいキーワードをレジストリーに登録する必要があります。このキーワードは次のとおりです。

```
Key Name: SOFTWARE¥ODBC¥ODBCINST.INI¥IBM DB2 ODBC DRIVER
Name: CPTimeout
Type: REG_SZ
Data: 60
```

32 ビット Windows オペレーティング・システム用の DB2 ODBC ドライバー バージョン 6 およびそれ以降は、接続プールを完全にサポートします。したがって、このキーワードは登録されています。バージョン 5.2 クライアントでは、 FixPack 3 (WR09024) またはそれ以降のバージョンをインストールしなければなりません。

デフォルト値 60 は、接続が切断されるまで 60 秒間プールされることを意味します。

接続が多い環境では、CPTimeout 値を大きな値にして (Microsoft は特定の環境について 10 分を推奨しています)、物理的な接続と切断を多くし過ぎないようにするのがよいでしょう。なぜなら、物理的な接続と切断が多いと、システム・メモリーと通信スタック・リソースも含めたシステム・リソースが大量に使用されるためです。

さらに、複数のプロセッサーがあるマシン上でオブジェクトが同じトランザクションに必ず同じ接続を使うようにするためには、 1 つのプロセッサーに対する複数のプールのサポートをオフにする必要があります。これを行うには、以下のレジストリー設定を `odbcpool.reg` というファイルにコピーし、それをプレーン・テキスト・ファイルとして保管してから、コマンド **odbcpool.reg** を発行します。これによって Windows オペレーティング・システムにこのレジストリー設定がインポートされます。

```
REGEDIT4

[HKEY_LOCAL_MACHINE¥SOFTWARE¥ODBC¥ODBCINST.INI¥ODBC Connection Pooling]
"NumberOfPools"="1"
```

このキーワードが 1 に設定されていないと、 MTS は複数の異なるプールに接続をプールする可能性があり、その場合には同じ接続が再使用されません。

ADO 2.1 以降を使用した MTS 接続のプール

MTS COM オブジェクトが ADO を使用してデータベースにアクセスする場合は、OLEDB リソースのプールをオフにして、 Microsoft の ODBC 用 OLEDB プロバイダー (MSDASQL) が ODBC の接続プールを妨害しないようにする必要があります。この機能は ADO 2.0 では `OFF` に初期化されていましたが、 ADO 2.1 では `ON` に初期化されています。 OLEDB リソースのプールをオフにするには、以下の行を `oledb.reg` というファイルにコピーし、それをプレーン・テキスト・ファイルとして保管してから、

コマンド **oledb.reg** を発行します。これによって、Windows オペレーティング・システムにこれらのレジストリー設定がインポートされます。

```
REGEDIT4

[HKEY_CLASSES_ROOT¥CLSID¥{c8b522cb-5cf3-11ce-ade5-00aa0044773d}]
@="MSDASQL"
"OLEDB_SERVICES"=dword:fffffffc
```

同一トランザクションに参加している COM オブジェクト間での ODBC 接続の再使用

MTS COM オブジェクトでの ODBC 接続では、接続プールが自動的にオンにされます (COM オブジェクトでトランザクションを行えるかどうかにかかわらず)。

同一のトランザクションに参加している複数の MTS COM オブジェクトについては、次のような方法で、2 つ以上の COM オブジェクトで接続を再使用できます。

COM1 および COM2 という 2 つの COM オブジェクトがあり、どちらも同じ ODBC データ・ソースに接続され、同じトランザクションに参加するとします。

COM1 が接続されて作業を終えると、切断されて接続はプールされます。しかし、この接続はトランザクションが同じである他の COM オブジェクトが使用するために予約されます。他のトランザクションでこの接続を使用できるようになるのは、現在のトランザクションが終了した後です。

COM2 が同じトランザクションで呼び出されると、プールされている接続が与えられます。 MTS は、同じトランザクションに参加している COM オブジェクトにのみ、接続が与えられるようにします。

他方、COM1 が明示的に切断されない場合は、トランザクションが終了するまで接続が独占されます。 COM2 が同じトランザクションで呼び出されると、別の接続が獲得されます。したがって、このトランザクションでは、1 つではなく 2 つの接続が独占されます。

同一のトランザクションに参加している COM オブジェクトの接続機能をこのように再使用することは、次のような理由により、望ましいものです。

- クライアントとサーバーの両方で使用するリソースが少ない。必要なのは 1 つの接続だけである。
- DB2 サーバーは MTS COM からの異なる接続を別々のトランザクションとして扱うので、同一のトランザクション (同一のデータベース・サーバーと同一のデータにアクセスする) に参加している 2 つの接続が互いにロックし合う可能性がなくなる。

TCP/IP 接続の調整

多くの物理的な接続と切断が同時に生じるような作業負荷が大きい環境で、低い CPTimeout 値が使用される場合は、 TCP/IP スタックにリソース上の制約が生じることがあります。

この問題を解決するには、TCP/IP レジストリー・エントリーを使用します。これは、*Windows NT Resource Guide*, Volume 1 で説明されています。レジストリー・キー値は、 HKEY_LOCAL_MACHINE—> SYSTEM—> CurrentControlSet—> Services—> TCPIP—> Parameters に保管されています。

デフォルト値と推奨されている値は次のとおりです。

名前	デフォルト値	推奨値
KeepAlive time	7200000 (2 時間)	同じ
KeepAlive interval	1000 (1 秒)	10000 (10 秒)
TcpKeepCnt	120 (2 分)	240 (4 分)
TcpKeepTries	20 (20 回再試行)	同じ
TcpMaxConnectAttempts	3	6
TcpMaxConnectRetransmission	3	6
TcpMaxDataRetransmission	5	8
TcpMaxRetransmissionAttempts	7	10
レジストリー値が定義されていない場合は、作成してください。		

MTS ″BANK″ サンプル・アプリケーションを使用した DB2 のテスト

MTS に付属の ″BANK″ サンプル・プログラムを使用して、クライアント製品と MTS のセットアップをテストできます。

以下のステップに従ってください。

1. ファイル ¥Program Files¥Common Files¥ODBC¥Data Sources¥MTSSamples.dsn を、以下のように変更します。

   ```
   [ODBC]
   DRIVER=IBM DB2 ODBC DRIVER
   UID=your_user_id
   PWD=your_password
   DSN=your_database_alias
   Description=MTS Samples
   ```

   ここで、

   - *your_user_id* と *your_password* は、ホストに接続するために使用されるユーザー ID とパスワードです。

- *your_database_alias* は、データベース・サーバーに接続するために使用されるデータベース別名です。

2. コントロール パネルの「ODBC」に移り、**「システム DSN」**タブを選択して、以下のようにデータ・ソースを追加します。
   a. 「IBM ODBC ドライバー (IBM ODBC Driver)」を選択して、**「完了」**を選択します。
   b. データベース別名のリストが表示されたら、以前に指定されたものを選択します。
   c. **「OK」**を選択します。
3. DB2 CLP を使用して、上記のとおりに *your_user_id* という ID で DB2 データベースに接続します。
   a. 次のように db2cli.lst ファイルをバインドします。

      ```
      db2 bind @C:¥sqllib¥bnd¥db2cli.lst blocking all grant public
      ```

   b. ユーティリティーをバインドします。

      サーバーが DRDA ホスト・サーバーである場合は、接続先のホスト (OS/390、AS/400、あるいは VSE&VM) に応じて、 ddcsmvs.lst、ddcs400.lst、または ddcsvm.lst をバインドします。たとえば、次のようにします。

      ```
      db2 bind @C:¥sqllib¥bnd¥@ddcsmvs.lst blocking all grant public
      ```

      サーバーが DRDA ホスト・サーバーでない場合は、次のようにして db2ubind.lst ファイルをバインドします。

      ```
      db2 bind @C:¥sqllib¥bnd¥@db2ubind.lst blocking all grant public
      ```

   c. それから、次のようにして MTS サンプル・アプリケーションのサンプル表とサンプル・データを作成します。

      ```
      db2 create table account (accountno int, balance int)
      db2 insert into account values(1, 1)
      ```

4. DB2 クライアントで、データベース・マネージャー構成パラメーター *tp_mon_name* が、 MTS に設定されていることを確認します。
5. ″BANK″ アプリケーションを実行します。**「Account」**ボタンを選択し、**「Visual C++」**オプションを選択してから、要求を発信します。他のオプションは SQL Server に固有の SQL を使用するので、機能しません。

第11章 高可用性とフェールオーバー・サポートの紹介

成功する e- ビジネスは、安定したトランザクション処理システムの可用性に依存しています。言い換えれば、DB2 のようなデータベース管理システムによって導かれているということであり、それゆえに、データベース管理システムは 24 時間 365 日 (『24 x 7』) 使用可能でなければならないのです。

高可用性

高可用性 (HA) という言葉は、ほとんど常に稼働していて顧客が使用できるシステムを説明する語として用いられます。高可用性のあるシステムには、次のことが求められます。

- 動作のピーク時であっても、ほとんどパフォーマンスを低下させることなく (可用性さえも下げることなく)、効率的にトランザクションを処理しなければなりません。区画化されたデータベース環境において、DB2 は、区画内並列性と区画間並列性の両方を利用して効果的にトランザクションを処理できます。*区画内並列性* は、SMP 環境において、複雑な SQL ステートメントの様々なコンポーネントを同時に処理するために使用できます。一方、区分データベース環境での*区画間並列性* は、関係するすべてのノード (各ノードは表内の行のサブセットを処理する) における照会の同時処理を示します。並列性に関する詳細は、 35ページの『並列化のタイプ』を参照してください。
- システムは、ハードウェアやソフトウェアの障害が発生したとき、あるいは災害に遭ったときに、迅速にリカバリーできなければなりません。 DB2 は、高機能な継続的チェックポイント・システムと並列リカバリー機能を備えており、きわめて迅速な破損リカバリーが可能です。

  迅速なリカバリーが可能であるかどうかは、その状況で実績のあるバックアップ / リカバリーのストラテジーを持っているかどうかに依存します。リカバリーのストラテジーに関する詳細は、 *データ回復と高可用性の手引きと解説書* を参照してください。
- エンタープライズ・データベースを動かすソフトウェアは、継続的に稼働し、トランザクション処理に使用できなければなりません。データベース・マネージャーを稼働し続けさせるためには、障害が発生したときに別のデータベース・マネージャーがこれを引き継げるようにしておく必要があります。これをフェールオーバーといいます。フェールオーバー 機能を使用すると、ハードウェアの故障があった場合に、あるシステムから別のシステムに自動的にワークロードを移すことが可能になります。

フェールオーバー保護は、永続的にログ・ファイルをロールフォワードしている別のマシンにデータベースのコピーを保持することによって可能になります。*ログ・シップ* は、ログ・ファイル全体をアーカイブ・デバイスから、あるいは 1 次データベースに対

して稼働するユーザー出口プログラムを通して待機用のマシンにコピーするプロセスです。このアプローチでは、DB2 復元ユーティリティーか分割ミラー機能を使用して、 1 次データベースが待機マシンに復元されます。新しい中断入出力のサポートを使用して、新しいデータベースを迅速に初期化できます (181ページの『オンライン分割ミラーと中断入出力のサポートによる高可用性』を参照してください)。待機マシンの 2 次データベースでは、ログ・ファイルが継続的にロールフォワードされます。 1 次データベースで障害が発生した場合は、残りのログ・ファイルがすべて待機マシンにコピーされます。ログの最後までロールフォワードして操作が停止した後、すべてのクライアントは待機マシンの 2 次データベースに再接続されます。

フェールオーバーは、ユーザーがプラットフォーム固有のソフトウェアをシステムに追加することによっても利用できます。たとえば、次のとおりです。

- AIX 用高可用性クラスター・マルチプロセッシング、拡張スケーラビリティー

  HACMP/ES に関する詳細は、 *データ回復と高可用性の手引きと解説書* か、 『IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES』 というタイトルの付いたホワイト・ペーパーを参照してください。こちらのホワイト・ペーパーは、『DB2 UDB and DB2 Connect Online Support』 の Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できます。

- Microsoft Cluster Server (Windows NT や Windows 2000 の場合)

  MSCS に関する詳細は、 *データ回復と高可用性の手引きと解説書* を参照してください。

- Sun Cluster または VERITAS Cluster Server (Solaris 操作環境の場合)

  Sun Cluster 2.x については、 *データ回復と高可用性の手引きと解説書* を参照してください。 Sun Cluster 3.0 については、『DB2 and High Availability on Sun Cluster 3.0』 というタイトルのホワイト・ペーパーを参照してください。こちらのホワイト・ペーパーは、『DB2 UDB and DB2 Connect Online Support』 の Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できます。 VERITAS Cluster Server については、『DB2 and High Availability on VERITAS Cluster Server』 というタイトルのホワイト・ペーパーを参照してください。このホワイト・ペーパーも『DB2 UDB and DB2 Connect Online Support』 の Web サイトから入手できます。

- Multi-Computer/ServiceGuard (Hewlett-Packard の場合)

  HP MC/ServiceGuard に関する詳細は、 『IBM DB2 EE v.7.1 Implementation and Certification With Hewlett-Packard's MC/ServiceGuard High Availability Software』 というタイトルのホワイト・ペーパーを参照してください。このホワイト・ペーパーは『DB2 UDB and DB2 Connect Online Support』 の Web サイト (http://www.ibm.com/software/data/pubs/papers/) から入手できます。

フェールオーバーのストラテジーは、通常、システムのクラスターに基づいて立てられます。クラスター は、接続されたシステムのグループで、 1 つのシステムとして機能します。各プロセッサーは、クラスター内のノードと見なされます。クラスター化を行

うと、障害の発生したサーバーのワークロードを別のサーバーが引き受けることによって、障害の発生時にサーバーの相互支援が可能になります。

IP アドレスの引き受け (IP 引き受け) は、サーバーがダウンしたときに、サーバーの IP アドレスをあるマシンから別のマシンに転送する機能です。この場合、クライアント・アプリケーションからは、時を違えて現れる 2 つのマシンが同じサーバーのように認識されます。

フェールオーバー・ソフトウェアでは、可用性を確保するためにシステム間でハートビート・モニター やキープアライブ・パケット が使用される場合があります。ハートビート・モニターには、クラスター内のすべてのノードの間で一定の通信を保守するシステム・サービスが関係しています。ハートビートが検出されなくなると、バックアップ・システムへのフォールオーバーが開始されます。エンド・ユーザーは通常、システムで障害が発生したことに気付きません。

市場で最も一般的なフェールオーバー・ストラテジーとして知られているのは、*アイドル・スタンドバイ* と *相互引き受け* です。ただし、ベンダーによっては、これらの語によって連想される構成が、異なった語に関連付けられている場合もあります。

アイドル・スタンドバイ
この構成では、DB2 インスタンスを実行するために 1 つのシステムが使用され、もう 1 つのシステムが「アイドル状態」つまりスタンドバイ・モードになっています。スタンドバイ・システムは、最初のシステムでオペレーティング・システムやハードウェアの障害が発生したらインスタンスを引き継げる状態になっています。必要なときまでスタンドバイ・システムがアイドル状態になっているので、システム全体のパフォーマンスは影響を受けません。

相互引き受け
この構成では、それぞれのシステムが他方のシステムのバックアップ用に指定されています。一方のシステムで障害が発生した場合は、バックアップ・システムが追加の作業、つまり自身の作業に加えて障害を起こしたシステムが行っていた作業を行わなければならなくなるため、システム全体のパフォーマンスは影響を受けます。

フェールオーバー・ストラテジーは、インスタンス、区画、あるいは複数の論理ノードのフェールオーバーにも使用できます。

オンライン分割ミラーと中断入出力のサポートによる高可用性

中断入出力 は、連続的なシステム使用可能性を、オンライン分割ミラー・ハンドリングのフル・インプリメンテーション (つまり、データベースをシャットダウンせずにミラーを分割) を提供することでサポートします。*分割ミラー* は、データベースの「インスタントな」コピーであり、、データが含まれているディスクをミラーリングし、コピーが必要なときにミラーを分割することによって作成されます。ディスク・ミラーリング

は、すべてのデータを 2 つの別個のハード・ディスク (1 つはもう一方のミラー) に書き込むプロセスです。そして、ミラーの*分割* は、ミラーのバックアップ・コピーを作成するプロセスです。

大規模なデータベースのバックアップに DB2 バックアップ・ユーティリティーを使用したくない場合は、中断入出力と分割ミラー機能を使用してミラー・イメージをコピーできます。このアプローチには、次のような利点もあります。

- 実動マシンで、バックアップ操作のオーバーヘッドをなくすことができる。
- システムを迅速に複製できる。
- アイドル・スタンドバイのフェールオーバーを迅速にインプリメントできる。初期復元操作はなく、ロールフォワード操作が遅すぎることが明らかになった場合やエラーが発生した場合は、非常に高速な再初期設定が行われます。

db2inidb コマンドは分割ミラーを初期化するので、以下に使用できます。

- 複製データベースの作成

  たとえば、レポートの作成などのために、1 次データベースの読み取り専用の複製を使用できます。
- スタンドバイ・データベースとして
- バックアップ・イメージとして

このコマンドは、分割オフのミラーに対してのみ実行できます。分割オフのミラーは、使用される前にまず **db2inidb** を実行しなければなりません。(*データ回復と高可用性の手引きと解説書* を参照してください)。

区分データベース環境では、いっさいの区画の分割イメージを使用する前にすべての区画で **db2inidb** コマンドを実行する必要があります。ツールは、すべての区画で同時に実行することができます。

複製データベースの作成

データベースの複製は、1 次 (実動) データベースのオフラインの「バックアップ」です。ただし、複製したデータベースをバックアップしたり、オリジナルのシステムにこのイメージを復元したり、オリジナルのシステムで生成されたログ・ファイルからロールフォワードしたりすることはできません。

データベースを複製するには、次のステップを実行します。

1. 1 次データベースでの入出力を中断します。

   ```
   db2 set write suspend for database
   ```
2. 適当なオペレーティング・システム・レベルのコマンドを使用し、1 次データベースのミラーを分割します。
3. 1 次データベースでの入出力を再開します。

   ```
   db2 set write resume for database
   ```

4. ミラーリングされたデータベースに別のマシンから接続します。
5. データベース・インスタンスを開始します。

```
db2start
```

6. ミラーリングしたデータベースを 1 次データベースの複製として初期化します。

```
db2inidb database_alias as snapshot
```

注: このコマンドを実行すると、分割が行われる時点で進行中だったトランザクションがロールバックされます。

スタンドバイ・データベースとしての分割ミラーの使用

ミラーリングされた (スタンドバイ) データベースはログを通して継続的にロールフォワードするため、 1 次データベースで作成されている新しいログは、継続的に 1 次システムからフェッチされます。分割ミラーをスタンドバイ・データベースとして使用するには、以下のステップに従ってください。

1. 1 次データベースでの入出力を中断します。

```
db2 set write suspend for database
```

2. 適当なオペレーティング・システム・レベルのコマンドを使用し、1 次データベースのミラーを分割します。
3. 1 次データベースでの入出力を再開します。

```
db2 set write resume for database
```

4. ミラーリングされたデータベースを別のインスタンスに接続します。
5. ミラーリングされたデータベースをロールフォワード保留状態にします。

```
db2inidb database_alias as standby
```

DMS 表スペース (データベース管理スペース) しかない場合は、実動データベースのバックアップをとるためにかかるオーバーヘッドを軽減するために、完全なデータベースのバックアップを作成できます。

6. 1 次システムからログ・ファイルを取り出すようにユーザー出口プログラムをセットアップします。
7. データベースをログの最後までロールフォワードします。
8. 1 次データベースがダウンするまで、ログ・ファイルの取り出しと、ログの終わりに向かったデータベースのロールフォワードを続けます。

バックアップ・イメージとしての分割ミラーの使用

分割ミラーを「バックアップ・イメージ」として使用するには、次のステップを実行します。

1. 1 次データベースでの入出力を中断します。

```
db2 set write suspend for database
```

2. 適当なオペレーティング・システム・レベルのコマンドを使用し、1 次データベースのミラーを分割します。
3. 1 次データベースでの入出力を再開します。

```
db2 set write resume for database
```

4. 1 次システムで、バックアップからの復元を必要とする障害が発生します。
5. オペレーティング・システム・レベルのコマンドを使用して、分割オフのデータを 1 次システムにコピーします。ロールフォワード・リカバリーで 1 次ログが必要になりますので、分割オフのログはコピーしないでください。
6. 1 次データベース・インスタンスを開始します。

```
db2start
```

7. 1 次データベースを初期化します。

```
db2inidb database_alias as mirror
```

8. 1 次データベースをログの最後までロールフォワードします。

第4部 付録

付録A. 命名規則

お知りになりたい命名規則について説明したセクションに進んでください。

- 『全体の命名規則』
- 『オブジェクト名の命名規則』
- 192ページの『連合システムで大文字小文字を区別する値を保持する方法』

全体の命名規則

特に指定しない限り、すべての名前に以下の文字を使用できます。

- A から Z のアルファベット。ほとんどの名前で使用される場合、 A から Z の文字は、小文字から大文字に変換されます。
- 0 から 9 の数字
- @、#、$、および _ (下線)

名前の先頭を数字や下線文字にすることはできません。

SQL 予約語を使用して表、視点、列、索引、または許可 ID を命名しないでください。SQL 予約語のリストは、*SQL 解説書* をご覧ください。

この他にも、個々のオペレーティング・システムや DB2 の動作環境によって、他の特殊文字が使用できる場合もあります。ただし、それらの文字が使用できる場合もあるとはいえ、それが必ず使用できるという保証はありません。データベース内のオブジェクトに名前を付ける際には、こういった他の特殊文字を使用しないように勧められています。

オブジェクト名の命名規則

すべてのオブジェクトは、全体の命名規則に従います。それに加えて、いくつかのオブジェクトには、下に示されているような別の制限が伴います。

表 24. データベース名、データベース別名、およびインスタンス名の命名規則

オブジェクト	ガイドライン
• データベース • データベース別名 • インスタンス	• データベース名は、カタログされている場所で固有である必要があります。これは、DB2 を UNIX ベースでインプリメントしている場合ならディレクトリー・パス、 Windows でインプリメントしている場合ならドライブ名になります。 • データベース別名は、システム・データベース・ディレクトリー内で固有である必要があります。新規のデータベースが作成されると、別名はデータベース名 (デフォルト値) になります。結果として、データベース別名と同じ名前のデータベースがない場合でも、その名前を使用してデータベースを作成することはできません。 • データベース名、データベース別名、インスタンス名は、8 バイト以下でなければなりません。 • Windows NT と Windows 2000 のシステムでは、インスタンスにサービス名と同じ名前を付けることはできません。 **注:** 潜在的な問題を避けるには、データベースを通信環境で使用したい場合に、データベース名に特殊文字 @、#、および $ を使用しないでください。また、これらの文字はすべてのキーワードに共通ではないので、別の言語でそのデータベースを使用する計画がある場合は、これらの文字を使用しないでください。

表 25. データベース・オブジェクトの命名規則

オブジェクト	ガイドライン
• 別名 • バッファー・プール • 列 • イベント・モニター • 索引 • メソッド • ノードグループ • スキーマ • ストアード・プロシージャー • 表 • 表スペース • トリガー • UDF • UDT • 視点	以下の場合を除き、名前の長さは最長で 18 バイトです。 • 表名 (ビュー名、要約表名、別名、および相関名を含む) は、最長で 128 バイトです。 • 列名は、最長で 30 バイトです。 • スキーマ名は、最長で 30 バイトです。 • オブジェクト名には以下を含めることができます。 – 有効なアクセント付き文字 (ö など) – マルチバイト・スペース以外のマルチバイト文字 (マルチバイト環境)

スキーマ名に関する追加情報

- 18 バイトより長いスキーマ名のある表は複製できません。
- ユーザー定義タイプ (UDT) には 8 バイトより長いスキーマ名を使用できません。
- 次のスキーマ名は予約語のため、使用できません:
  SYSCAT、SYSFUN、SYSIBM、SYSSTAT
- 将来のマイグレーションで問題が生じる潜在的な原因を無くすため、 SYS で始まるスキーマ名は使用しないでください。データベース・マネージャーでは、SYS で始まるスキーマ名を使用して、トリガー、ユーザー定義タイプ、またはユーザー定義関数を作成することはできません。
- スキーマ名として SESSION を使用しないようにもお勧めします。宣言済み一時表は、 SESSION により修飾されなければなりません。したがって、アプリケーションで宣言された一時表が持続表と同じ名前になり、アプリケーション論理が非常に複雑になる可能性があります。宣言済み一時表を処理する場合を除いては、スキーマ SESSION の使用は避けてください。

表 26. ユーザー、ユーザー ID およびグループの命名規則

オブジェクト	ガイドライン
• グループ名 • ユーザー名 • ユーザー ID	• グループ名は最長で 8 バイトです。 • UNIX ベースのシステムでは、ユーザー ID が最長で 8 文字です。 • Windows では、ユーザー名が最長で 30 文字です。 Windows NT と Windows 2000 では、現在 20 文字が実質的な限界です。 • DCE 認証を使用する場合は、ユーザー名の制限が 8 文字になります。 • DCE 認証やクライアント認証を使用しない場合は、ユーザー名とパスワードが明示的に指定されていれば、 8 文字以上のユーザー名で Windows NT や Windows 2000 に接続する非 Windows 32 ビット・クライアントがサポートされます。 • 次のような名前や ID は使用できません。 – USERS、ADMINS、GUESTS、PUBLIC、 LOCAL または *SQL 解説書* にリストされているすべての SQL 予約語。 – IBM、SQL または SYS で始まる名前。 – アクセント付き文字を含む名前。 UNIX-based ベースのシステムでは、グループとユーザーの名前を同じにすることができます。 GRANT ステートメントの場合、グループまたはユーザーのいずれを参照するかを指定する必要があります。 REVOKE ステートメントの場合、許可カタログ表にさまざまな GRANTEETYPE 値の GRANTEE 行が複数あるかどうかに基づいて、ユーザーまたはグループを指定します。 Windows NT では、ローカル・グループ、グローバル・グループ、およびユーザーを同じ名前にすることはできません。 OS/2 では、グループとユーザーの名前を同じにすることはできません。 **注:** 1. オペレーティング・システムによっては、大文字小文字の区別のあるユーザー ID およびパスワードを使用できます。ご使用のオペレーティング・システムではその区別があるかどうか、資料を見て調べてください。 2. CONNECT または ATTACH が成功した場合に戻される許可 ID は、8 文字に切り捨てられます。許可 ID には、切り捨てを示す省略符号 (...) が付けられます。詳しくは、*SQL 解説書* の CONNECT ステートメントを参照してください。

パスワードに関する追加情報

パスワードの保守作業を行うために必要となることがあります。このような作業がサーバーで必要となり、かつ多数のユーザーがそのサーバー環境で効率よくまたは快適に働

けないため、この作業を実行するのは重大な障害となる可能性があります。 DB2 UDB には、サーバー上にいなくても、パスワードを更新し、検査するための手段が備わっています。たとえば、DB2 (OS/390 版) のバージョン 5 では、ユーザーのパスワードを変更するこの方式をサポートしています。エラー・メッセージ SQL1404N『パスワード失効』が出された場合には、下記のように CONNECT ステートメントを使用してパスワードを変更します。

```
CONNECT TO <database> USER <userid> USING <password>
   NEW <new_password> CONFIRM <new_password>
```

DB2 クライアント構成アシスタントの「パスワード変更 (Password change)」ダイアログもまた、パスワードを変更するのに使用される場合があります。パスワード変更の方法についての詳細は、 *SQL 解説書*、および CCA オンライン・ヘルプを参照してください。

表 27. 連合データベース・オブジェクトの命名規則

オブジェクト	ガイドライン
• 機能マッピング • 索引仕様 • ニックネーム • サーバー • タイプ・マッピング • ユーザー・マッピング • ラッパー	• ニックネーム、マッピング、索引仕様、サーバー名、およびラッパー名は、128 バイトを超過することはできません。 • サーバーとニックネームのオプション、およびオプション設定は、255 バイトに制限されています。 • 連合データベース・オブジェクトの名前には以下の文字を使用できます。 – 有効なアクセント付き文字 (ö など) – マルチバイト・スペース以外のマルチバイト文字 (マルチバイト環境)

オブジェクト名における区切り ID の使用

キーワードを使用できます。 SQL キーワードとして解釈される可能性もあるコンテキストでキーワードを使用する場合は、これを区切り ID として指定する必要があります。

区切り ID を使用すると、上記の命名規則に違反するオブジェクトの作成も可能ですが、それを引き続き使用すると、エラー状態になることがあります。たとえば、名前に + や - 記号が含まれている列を作成した後に、その列を索引で使用すると、表を再編成しようとするときに問題が起きます。区切り文字については、*SQL 解説書* の『SQL 区切り文字』を参照してください。

連合システムで大文字小文字を区別する値を保持する方法

分散要求では、ID とパスワードを指定しなければならないことがあります。 ID とパスワードは、データ・ソースでは大文字小文字が区別されます。データ・ソースに渡されたときに大文字小文字が正しいことを確認するには、以下の指針に従ってください。

- ID とパスワードは要求されている文字で指定し、二重引用符で囲みます。
- ユーザー ID を指定しているのであれば、データ・ソースの *fold_id* サーバー・オプションを ″n″ (『大文字小文字を変更せず』) に設定します。パスワードを指定しているのであれば、データ・ソースの *fold_pw* サーバー・オプションを ″n″ に設定します。

  ユーザー ID およびパスワード指定のための別の方法があります。データ・ソースで小文字のユーザー ID を必要とする場合は、任意の文字で指定してから *fold_id* サーバー・オプションを ″l″ (『この ID を小文字でデータ・ソースへ送る』) に設定できます。データ・ソースで大文字のユーザー ID を必要とする場合は、任意の文字で指定してから *fold_id* を ″u″ (『この ID を大文字でデータ・ソースへ送る』) に設定できます。同様に、データ・ソースで小文字あるいは大文字のパスワードを必要としたとしても、 *fold_pw* サーバー・オプションを ″l″ か ″u″ にセットすれば、この要件を満たすことができます。

  サーバー・オプションに関する詳細は、 *管理の手引き: インプリメンテーション* の『データ・ソースの定義に役立ち、認証処理を容易にするサーバー・オプションの使用』 を参照してください。
- オペレーティング・システムのコマンド・プロンプトで、大文字小文字を区別する ID またはパスワードを二重引用符で囲む場合、システムがその二重引用符を正しく解析することを確認しなければなりません。そのためには、以下のようにします。
  - UNIX ベースのオペレーティング・システムでは、ステートメントを単一引用符で囲みます。
  - Windows NT オペレーティング・システムでは、各引用符の前に円記号を付けます。

  たとえば、DB2 データ・ソースにある多くの区切り ID では、大文字小文字を区別します。 NORBASE というデータ・ソースに存在する DB2 for CS 視点 "my_schema"."wkly_sal" のために NICK1 というニックネームを作成するとします。

  UNIX ベースのシステムのコマンド・プロンプトでは、次のように入力します。

  ```
  db2 'create nickname nick1 for norbase."my_schema"."wkly_sal"'
  ```

  Windows NT コマンド・プロンプトでは、次のように入力します。

  ```
  db2 create nickname nick1 for norbase.¥"my_schema¥".¥"wkly_sal¥"
  ```

  DB2 のコマンド・プロンプト (対話モード) からステートメントを入力したり、アプリケーション・プログラムでステートメントを指定する場合は、単一引用符や円記号

| は必要ありません。たとえば、UNIX ベースのシステムまたは Windows NT システ
| ムのいずれかの DB2 コマンド・プロンプトで、次のように入力します。

```
create nickname nick1 for norbase."my_schema"."wkly_sal"
```

付録B. データベース・マイグレーションの計画

このセクションでは、マイグレーション作業の概要を説明します。 DB2 UDB バージョン 6 のデータベースは、バージョン 7 にマイグレーションする必要がないということに注意してください。 DB2 UDB バージョン 5.*x* のデータベースのマイグレーションに関する詳細は、ご使用のオペレーティング・システムに応じた*概説*および*インストール* に説明されています。

データベースのマイグレーション時には、次のようになります。

- 次のデータベース・エンティティーがマイグレーションされます。
  - データベース構成ファイル
  - データベース・システム・カタログ表
  - データベース・ディレクトリー
  - データベース・ログ・ファイル・ヘッダー
- システム・カタログ表が次のように変更されます。
  - 新規の列が追加される。
  - 新規の表が作成される。
  - カタログ視点セットがマイグレーションされ、新規のカタログ視点が SYSCAT スキーマに作成される。
  - 更新可能なカタログ視点セットが SYSSTAT スキーマに作成される。
  - 汎用スカラー関数セットは SYSFUN スキーマに保持されたまま、新しい汎用スカラー関数のセットがそこに作成される。 SYSFUN.DIFFERENCE スカラー関数だけは除去され、データベース・マイグレーション時に再作成されます。
- データベース・ヒストリー・ファイルとそのシャドーは、データベース・ディレクトリーに作成されます。このファイルには、データベースが復元される必要のある場合に使用できる、バックアップ情報の要約が含まれます。これは、データベースに対して特定の操作が実行されるときには、必ず更新されます。表スペースのバックアップおよび復元の操作のために、バックアップ情報の要約も保持されます。

マイグレーションに関する考慮事項

旧バージョンのデータベース・マネージャーで作成されたデータベースを正常にマイグレーションするには、次のことを考慮する必要があります。

- 196ページの『マイグレーションの制約事項』
- 196ページの『セキュリティーおよび権限』
- 196ページの『ストレージ要件』

- 197ページの『リリース間の非互換性』

マイグレーションの制約事項

データベースをバージョン 7 にマイグレーションしようとするときには、前提条件または制約事項がいくつかあるので、それについて知っておいてください。

- マイグレーションが行えるのは、V5.x または V6 からだけです。 DB2 V1.2 パラレル・エディションからのマイグレーションはサポートされていません。 DB2 (データベース・マネージャー) のこれよりも前のバージョンは、 V5.x または V6 にマイグレーションしてから V7 にマイグレーションします。
- データベースを V7 サーバーにマイグレーションするために、 V7 クライアントのマイグレーション・コマンドを出すことができます。ただし、古い DB2 クライアントのマイグレーション・コマンドでは、データベースを V7 サーバーにマイグレーションすることはできません。
- 異なるプラットフォーム間でのマイグレーションはサポートされません。
- ご使用のデータベース内のユーザー・オブジェクトが、オブジェクト修飾子として V7 の予約済みスキーマ名を持つことはできません。これらの予約済みスキーマ名には SYSCAT、SYSSTAT、および SYSFUN があります。
- データベースをマイグレーションする前に、BIGINT、REAL、DATALINK、または REFERENCE という名前のユーザー定義の特殊タイプの名前を変更しなければなりません。
- 次のいずれかの状態のデータベースはマイグレーションすることができません。
  - バックアップ保留状態
  - ロールフォワード保留状態
  - 1 つまたは複数の表スペースが正常状態ではない
  - トランザクション不整合
- 下位レベル (V5.x または V6) データベースのバックアップを復元することはできますが、下位レベル・ログのロールフォワードはサポートされません。

セキュリティーおよび権限

データベースをマイグレーションするには、SYSADM 権限が必要です。

ストレージ要件

マイグレーションする際には新旧両方のカタログにスペースが必要です。必要になるディスク・スペースの量は、データベースの数とサイズ、および複雑さによって異なります。これらのオブジェクトには、すべての表と視点が入ります。現在データベース・カタログが占めているディスク・スペースの少なくとも 2 倍のディスク・スペースを使用可能にしておく必要があります。ディスク・スペースが十分にないと、マイグレーションは失敗します。

SYSCAT 表スペースが SMS タイプの表スペースである場合、ログ・ファイルに関連するデータベース構成パラメーターを更新することも考慮する必要があります。これらのログ・ファイルのスペースが不足しないように (理由コード 3 の SQL1704N になる)、*logfilsiz*、*logprimary*、および *logsecond* の値を大きくしてください。スペースが不足した場合は、ログ・スペース・パラメーターを大きくして、 MIGRATE DATABASE コマンドを出し直してください。

リリース間の非互換性

データベースのマイグレーションを計画するときは、 2 つのバージョンの製品の非互換性の影響について考慮してください。

バージョン 7 の拡張機能を利用するためには、データベースをマイグレーションした後で、データベースおよびデータベース・マネージャー構成を調整する必要があります。マイグレーション前とマイグレーション後の構成パラメーター値を記録して比較すれば、この調整を簡単に行うことができます。 (GET DATABASE CONFIGURATION コマンドおよび GET DATABASE MANAGER CONFIGURATION コマンドについては、*コマンド解説書* を参照してください。)

データベースのマイグレーション

次に、データベースをマイグレーションする場合に行う必要のあるステップを示します。マイグレーションを開始するには、その前にデータベース・マネージャーを始動する必要があります。

マイグレーションの前に

注: マイグレーション前ステップは、以前のリリース (すなわち、新しいリリースにマイグレーションする前の、あるいは、新しいリリースをインストールする前の現行のリリース) で行う必要があります。

1. 196ページの『マイグレーションの制約事項』に関する未解決の問題がないことを確認します。
2. すべてのアプリケーションおよびエンド・ユーザーを、マイグレーションする各データベースから切断します (必要に応じて、 LIST APPLICATIONS コマンドまたは FORCE APPLICATIONS コマンドを使用してください)。
3. DB2CKMIG マイグレーション前ユーティリティーを使用して、データベースがマイグレーション可能かどうか判別します (このユーティリティーの使用についての詳細は、ご使用のプラットフォームに対応した*概説およびインストール* を参照してください)。 Windows NT または OS/2 では、インストール時にこのツールを実行するようにプロンプトが出されますが、 UNIX ベースのシステムでは、このツールはインスタンスのマイグレーション時に自動的に呼び出されるということに注意してください。
4. データベースをバックアップします。

マイグレーションはリカバリー不可能なプロセスです。バージョン 6 の予約済みスキーマ名を変更する前にデータベースのバックアップを取ると、 DB2 UDB バージョン 7 を使用してデータベースを復元することができなくなります。このデータベースを復元するには、前のバージョンのデータベース・マネージャーを使用しなければなりません。

警告! データベースのバックアップを取っていない場合は、マイグレーションが失敗すると、 DB2 UDB バージョン 7 または以前のバージョンのデータベース・マネージャーを使用してデータベースを復元することができなくなります。

また、バックアップが取られた時点と、バージョン 7 へのアップグレードが完了する時点の間に行われたデータベース・トランザクションは、リカバリー不能であることも知っている必要があります。すなわち、バージョン 7 のインストールおよびマイグレーションの完了に続く時点で、データベースを (バージョン 7 レベルに) 復元する必要が起きた場合、バージョン 7 のインストールよりも前に書き込まれたログはロールフォワード・リカバリーでは使用できません。

マイグレーション

5. 次のいずれか 1 つを使用して、データベースをマイグレーションします。
   - MIGRATE DATABASE コマンド
   - RESTORE DATABASE コマンド (データベースの全バックアップを復元する場合)
   - sqlemgdb - データベース・マイグレーション API

   OS/2 の場合: DB2CIDMG マイグレーション・ユーティリティーは DB2 (OS/2 版) だけで使用できます。これは、構成 / インストール / 配布 (Configuration/Installation/Distribution (CID)) 体系環境で実行されます。これは、LAN ベースのワークステーションでリモートからオペレーター不在のインストールおよび構成を行えるようにします。 CID マイグレーションを使用するには、ご使用の LAN に NetView DM/2 が必要です。

   UNIX ベースのシステムの場合: 特定のインスタンス内のすべてのデータベースをマイグレーションしない場合の作業については、ご使用のプラットフォームに対応した*概説およびインストール* に説明があります。

マイグレーションの後で

6. オプションで、DB2UIDDL ユーティリティーを使用して、独自のスケジュールによる固有索引の段階的なマイグレーションを容易に管理することもできます。 (バージョン 5 で作成された DB2 バージョン 5 のデータベースは、このツールを利用して据え置き固有性検査を行う必要はありません。これは、バージョン 5 で作成された固有索引にはすでにこれらのセマンティクスがあるためです。ただし、それ以前のバージョンからバージョン 5 にマイグレーションされたデータベースの場合は、 DB2UIDDL ユーティリティーを使用して固有索引を変更しない限り、これらのセマンティクスが自動的に付与されることはありません。) このユーティリティ

ーは、ユーザー表に対する固有索引用に CREATE UNIQUE INDEX ステートメントを生成し、そのステートメントをファイルに書き込みます。このファイルを DB2 CLP コマンド・ファイルとして実行すると、固有索引がバージョン 7 セマンティクスに変換されます。このユーティリティーの詳細については、 *概説およびインストール* を参照してください。

7. オプションで、SQL 照会のパフォーマンスに特に重大な影響を与える表に対して RUNSTATS コマンドを出すこともできます。古い統計はマイグレーション・データベースに保存されます。この統計は、RUNSTATS コマンドを呼び出さない限り更新されません。
8. オプションで、 DB2RBIND ユーティリティーを使用して、すべてのパッケージの再妥当性検査を行ったり、またはパッケージが最初に使用されるときにパッケージ再妥当性検査が暗黙で生じるようにすることもできます。
9. バージョン 7 で Explain 表を使用する計画の場合は、オプションで Explain 表をマイグレーションすることができます。詳細については、 *管理の手引き: パフォーマンス* で『SQL Explain 機能』を参照してください。
10. バージョン 7 の拡張機能を十分活用できるように、データベースとデータベース・マネージャー構成パラメーターを調整してください。

付録C. リリース間の非互換性

このセクションでは、DB2 ユニバーサル・データベースと DB2 の以前のリリースとの間にある非互換性について説明します。

非互換性 とは、 DB2 ユニバーサル・データベースのうち、以前のリリースの DB2 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 ユニバーサル・データベース バージョン 6 およびバージョン 7 の非互換性について説明します。非互換性は次のようにソートされます。

- システム・カタログの視点
- 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
- 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・システムを示す記号が以下のように示されます。

WIN DB2 がサポートする Microsoft Windows プラットフォーム

UNIX DB2 がサポートする UNIX ベースのプラットフォーム

OS/2 OS/2

注: DB2 ユニバーサル・データベース バージョン 6 以降、バージョン 1.x およびバージョン 2.x クライアントは、 (DB2 パラレル・エディション バージョン 1.2 サーバーに付属のクライアントを含めて) サポートされていません。

DB2 ユニバーサル・データベースの計画済みの非互換性

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 ユニバーサル・データベースのユーザーは、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マイグレーションすることができます。

将来の DB2 ユニバーサル・データベースのバージョンでの読み取り専用視点

WIN	UNIX	OS/2

変更点

システム・カタログ視点は、読み取り専用視点になります。 SYSSTAT 視点は、更新可能のままです。

症状

SYSCAT 視点の列で実行していた UPDATE ステートメントが失敗します。

説明

SYSCAT 視点での定義にしたがって列を更新することにより、カタログ内の値を変更するようにツールまたはアプリケーションがエンコードされています。

解決方法

SYSSTAT 視点での定義にしたがって列を更新して、カタログを変更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 ユニバーサル・データベースのバージョンでの PK_COLNAMES および FK_COLNAMES

WIN	UNIX	OS/2

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 視点を使用するように、ツールまたはアプリケーションを変更します。

将来の DB2 ユニバーサル・データベースのバージョンで COLNAMES が使用できない

WIN	UNIX	OS/2

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 視点を使用するように、ツールまたはアプリケーションを変更します。

DB2 ユニバーサル・データベース バージョン 7 の非互換性

このセクションでは、DB2 ユニバーサル・データベース バージョン 7 の非互換性について説明します。

アプリケーション・プログラミング

クエリー・パトローラー・ユニバーサル・クライアント

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、クエリー・パトローラー・サーバー バージョン 7 でのみ動作します。 CAE は DB2 へのアプリケーション・インターフェースで、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。 *アプリケーション開発の手引き* で説明されているように、 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・プロシージャーが隔離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、隔離されたストアード・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 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 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 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。

スキーマ SESSION によって修飾される正規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 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、正規基礎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、アプリケーション固有の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礎表が存在しても、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、常に永続基礎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 からのマイグレーション。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) との間では共用できません。
- バージョン 6 からバージョン 7 へのマイグレーション。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、視点、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、視点、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 にマイグレーション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

Solaris でのデータ・リンク・ファイル・マネージャーとファイル・システム・フィルター

	UNIX	

変更点: データ・リンク・ファイル・マネージャーとファイル・システム・フィルターは、Solaris OS 2.5.1 ではサポートされていません。

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。

データ・リンク・ファイル・システムと Norton ユーティリティー (Norton Utilities**)

WIN		

変更点: Windows NT データ・リンク・ファイル・システムは、Norton ユーティリティーと互換性がありません。

症状: ファイルが DLFS により制御されているドライブから削除されると、次のカーネル例外が発生します: エラー 0x1E (カーネル・モード例外は扱えません)。例外は 0xC00000005 (アクセス違反) になります。

説明: このアクセス違反は、 Norton ユーティリティー・ドライバーが DLFS フィルター・ドライバーのロード後にロードされるために起きます。

解決方法: 回避策としては、 Norton ユーティリティー・ドライバーがロードされた後に DLFSD ドライバーをロードする方法があります。「スタート」から、「設定—> コントロール パネル—> デバイス—> DLFSD」を選択して、 DLFSD ドライバーの開始を手動に変更します。

システムの始動時に DLFSD ドライバーと DLFM サービスをロードするバッチ・ファイルを作成できます。バッチ・ファイルの内容は次のようになります。

```
net start dlfsd
net start "dlfm service"
```

このバッチ・ファイルに start_dlfs.bat という名前を付け、**WINNT¥Profiles¥Administrator¥Start Menu¥Programs¥Startup** ディレクトリーにコピーします。管理者のみが DLFS フィルター・ドライバーおよび DLFS サービスをロードする権限を持っています。

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。

DB2 ユニバーサル・データベース バージョン 6 の非互換性

このセクションでは、 DB2 ユニバーサル・データベース バージョン 6 の非互換性について説明します。

システム・カタログの視点

DB2 ユニバーサル・データベース バージョン 6 でのシステム・カタログ視点

WIN	UNIX	OS/2

変更点: システム・カタログ視点では、新しいコードが導入されています。タイプ付き表を表す ″U″ と、タイプ付き視点を表す ″W″ です。

症状: システム・カタログで表および視点を検索する照会で、表にタイプ・コード ″T″ を、視点に ″V″ をそれぞれ使用する場合、タイプ付き表および視点が見つかりません。

説明: システム・カタログ視点 TABLES、PACKAGEDEP、TRIGDEP、および VIEWDEP を含む一部のシステム・カタログには、 1 文字タイプ・コードを含む列 TYPE または BTYPE があります。バージョン 5.2 では、すべての表にタイプ・コード ″T″ が、そしてすべての視点に ″V″ が使用されていました。バージョン 6 では、非タイプ付き表のタイプ・コードは ″T″ のままで、タイプ付き表には新しいタイプ・コード ″U″ が割り当てられます。同様に、非タイプ付き視点のタイプ・コードは ″V″ のままで、タイプ付き視点には新しいタイプ・コード ″W″ が割り当てられます。また、階層表と呼ばれる新しい種類の表は、ユーザーが直接作成するものではなく、システムが表階層を実装するために使用するものですが、システム・カタログ表にタイプ・コード ″H″ で表示されます。

解決方法: タイプ付き表または視点のコードを認識できるように、ツールまたはアプリケーションを変更します。ツールまたはアプリケーションに表の論理ビューが必要な場合には、タイプ・コード ″T″、″U″ 、″V″ 、および ″W″ が使用されます。ツールまたはアプリケーションに階層表を含む表の物理ビューが必要な場合には、タイプ・コード ″T″ および ″H″ が使用されます。

DB2 ユニバーサル・データベース バージョン 6 での基本および外部キー列名

WIN	UNIX	OS/2

変更点: 2 つの SYSCAT.REFERENCES 列、PK_COLNAMES と FK_COLNAMES のデータ・タイプは、 VARCHAR(320) から VARCHAR(640) に変更されています。

症状: 基本キーまたは外部キーの列名が、切り捨てられたか、正しくないか、または欠落しています。

説明: 基本キーまたは外部キーで 18 バイトより長い列名が使用される場合、これら 2 つの列に列名のリストが保管される形式はもはや同じではありません。長さ (*n*) が

18 を超える列に続く、ブランクで区切られた 20 バイトの列名は、 *n*-18 バイトだけ右にシフトされます。同様に、列名のリストが 640 バイトを超える場合には、列には空ストリングが含まれます。

解決方法: SYSCAT.KEYCOLUSE 視点には、基本、外部、および固有キーを構成する列のリストが含まれており、 SYSCAT.REFERENCES にある列の代わりにこれを使用しなければなりません。代わりに、ユーザーは列名の名前を 18 バイトに制限するか、または列のリストの合計の長さを 640 バイトに制限できます。

DB2 ユニバーサル・データベース バージョン 6 での SYSCAT.VIEWS の列 TEXT

WIN	UNIX	OS/2

変更点: SYSCAT.VIEWS の列 TEXT にある視点テキストは、複数の行に分割されなくなりました。データ・タイプは、VARCHAR(3600) から CLOB(64K) に変更されています。

症状: ツールまたはアプリケーションは、完全な視点テキストを提供しません。

説明: 一度に TEXT 列から戻される長さが 3600 (または 3900) を超えないことを想定して作成されたツールまたはアプリケーションは、このフィールドのサイズの増加に対応できません。複数の行を検索し、SEQNO フィールドを使用して視点テキストを再構築するメカニズムは、必要ではなくなりました。 SEQNO 値は常に 1 になります。

解決方法: 3600 バイトより大きい TEXT 列からの値を処理できるように、ツールまたはアプリケーションを変更します。または、視点 TEXT を 3600 バイト以内に収めるように書き直すこともできます。

DB2 ユニバーサル・データベース バージョン 6 での SYSCAT.STATEMENTS の列 TEXT

WIN	UNIX	OS/2

変更点: SYSCAT.STATEMENTS の列 TEXT にあるステートメント・テキストは、複数の行に分割されなくなりました。データ・タイプは、VARCHAR(3600) から CLOB(64K) に変更されています。

症状: ツールまたはアプリケーションは、完全なステートメント・テキストを提供しません。

説明: 一度に TEXT 列から戻される長さが 3600 (または 3900) を超えないことを想定して作成されたツールまたはアプリケーションは、このフィールドのサイズの増加に

対応できません。複数の行を検索し、 SEQNO フィールドを使用してステートメント・テキストを再構築するメカニズムは、必要ではなくなりました。 SEQNO 値は常に 1 になります。

解決方法: 3600 バイトより大きい TEXT 列からの値を処理できるように、ツールまたはアプリケーションを変更します。または、ステートメント TEXT を 3600 バイト以内に収めるように書き直すこともできます。

DB2 ユニバーサル・データベース バージョン 6 での SYSCAT.INDEXES の列 COLNAMES

WIN	UNIX	OS/2

変更点: SYSCAT.INDEXES の列 COLNAMES データ・タイプは、 VARCHAR(320) から VARCHAR(640) に変更されています。

症状: 列名が索引から欠落しています。

説明: データ・タイプ VARCHAR(320) の列からデータを検索するように作成されたツールまたはアプリケーションは、このフィールドのサイズの増加には対応できません。

解決方法: SYSCAT.INDEXCOLUSE 視点には、索引を構成する列のリストが含まれており、 COLNAMES 列の代わりにこれを使用しなければなりません。あるいは、索引から列を除去するか、列名のサイズを小さくして、列名のリストが (先頭の + または – も含めて) 320 バイト以内に収まるようにすることもできます。

DB2 ユニバーサル・データベース バージョン 6 での SYSCAT.CHECKS の列 TEXT

WIN	UNIX	OS/2

変更点: CHECKS 列 TEXT データ・タイプは、CLOB(32K) から CLOB(64K) に変更されています。

症状: 検査制約文節が不完全です。

説明: データ・タイプ CLOB(32K) の列からデータを検索するように作成されたツールまたはアプリケーションは、このフィールドのサイズの増加には対応できません。

解決方法: 32 KB より長い TEXT 列からの値を処理できるように、ツールまたはアプリケーションを変更します。あるいは、検査制約文節を書き直して、 32 KB に収まるように文字数を少なくすることもできます。

DB2 ユニバーサル・データベース バージョン 6 での BIGINT の列データ・タイプ

WIN	UNIX	OS/2

変更点: システム・カタログ視点列の中には、データ・タイプが INTEGER から BIGINT に変えられているものがあります。

症状: 値、特に統計情報の値が、予想より小さすぎ (または大きすぎ) ます。

説明: データ・タイプ INTEGER の列からデータを検索するように作成されたツールまたはアプリケーションは、このフィールドのサイズの増加には対応できません。

解決方法: INTEGER フィールドで保管できる最大値または最小値を超える値を処理できるように、ツールまたはアプリケーションを変更します。あるいは、INTEGER フィールドで表示できる値を超える原因となっている、基礎となる構造または SQL コードを変更することもできます。

DB2 ユニバーサル・データベース バージョン 6 での列のミスマッチ

WIN	UNIX	OS/2

変更点: SYSCAT 視点定義で視点の終わりに新しい列が挿入されません。

症状: 複数の列の不一致または列のデータ・タイプの不一致で再プリプロセスに失敗します。

説明: システム・カタログ視点に新しい列が導入されており、随時照会環境で便利な位置にあります。特に、短い列は非常に長い列の前にあり、REMARKS 列は常に最後にあります。

解決方法: ″SELECT *″ とコーディングする代わりに、選択リストで列を明示的に指定します。

DB2 ユニバーサル・データベース バージョン 6 での SYSCAT.COLUMNS および SYSCAT.ATTRIBUTES

WIN	UNIX	OS/2

変更点: SYSCAT.COLUMNS および SYSCAT.ATTRIBUTES には現在、継承された列および属性のエントリーが入っています。

症状: タイプ付き表または視点の列を検索する SYSCAT.COLUMNS の照会、および構造型の属性を検索する SYSCAT.ATTRIBUTES の照会は、照会の対象が副表、副視点、またはサブタイプの場合には、バージョン 5.2 よりバージョン 6 の方が多くの行を戻す可能性があります。

説明: バージョン 5.2 では、指定された表、視点、または構造タイプについて、COLUMNS および ATTRIBUTES カタログには、その表、視点、またはタイプにより導入された、列および属性のエントリーしか含まれていませんでした。スーパー表またはスーパータイプから継承した列および属性は、カタログに表示されませんでした。しかし、バージョン 6 では、COLUMNS および ATTRIBUTES カタログには、継承された列および属性のエントリーが入ります。

解決方法: COLUMNS および ATTRIBUTES カタログで新しいエントリーを認識するように、ツールまたはアプリケーションを変更します。

DB2 ユニバーサル・データベース バージョン 6 で OBJCAT 視点がサポートされなくなった

WIN	UNIX	OS/2

変更点: バージョン 5.2 の OBJCAT スキーマにある再帰的カタログ視点は、 DB2 ユニバーサル・データベース製品の一部として提供されなくなりました。

症状: OBJCAT カタログ視点に対して作成された照会が、正しく実行しません。

解決方法: 以前 OBJCAT 視点にあった情報の大半は、正規の SYSCAT カタログ視点に組み込まれています。たいていの場合、システム・カタログ視点から情報が得られます。バージョン 5.2 からマイグレーションする場合で、OBJCAT カタログ視点が存在する場合には、これらを除去しなければなりません。これを実行するには、ディレクトリー `sqllib` の `misc` サブディレクトリーの下の `objcatdp.db2` という CLP スクリプトを実行します。

バージョン 5.2 でサポートされているカタログと同等の、独自の OBJCAT 視点のセットを作成することもできます。

バージョン 5.2 では、*SQL 解説書* の『付録 E』で、 OBJCAT カタログ視点は一時的なものであり、将来のリリースではサポートされなくなることが警告されていました。

DB2 ユニバーサル・データベース バージョン 6 で従属関係が変更された

WIN	UNIX	OS/2

変更点: システム・カタログ視点では、階層従属関係は以前はコード ″H″ で表示されていましたが、現在はコード ″O″ で表示されています。

症状: カタログ視点で、コード ″H″ によって階層従属性を検索する照会が、正しく動作しなくなりました。

説明: システム・カタログ視点 PACKAGEDEP、TRIGDEP、および VIEWDEP を含むいくつかのシステム・カタログには、BTYPE という列があります。バージョン 5.2 では、OBJCAT 視点は階層従属関係をコード ″H″ で表示していました。バージョン 6 では、これらの従属関係はコード ″O″ で表示されています。

解決方法: コード ″O″ を検索するように、これらの照会を変更します。

DB2 ユニバーサル・データベース バージョン 6 での SYSIBM 基本カタログ表

WIN	UNIX	OS/2

変更点: 以下に示すのは、現在でも SYSCAT 視点の代わりに使用できる SYSIBM 基本カタログ表の変更点です。

- 削除されたフィールド (SYSCAT 視点にはある)
  - SYSSTMT.SEQNO
  - SYSVIEWS.SEQNO
- 名前変更されたカタログ表: SYSTRIGDEP は SYSDEPENDENCIES に変更。同様に、列 BCREATOR および DCREATOR は、それぞれ BSCHEMA および DSCHEMA に名前変更されました。視点 SYSCAT.TRIGDEP は変更されていません。
- 削除されたフィールド (SYSCAT 視点にもない)
  - SYSATTRIBUTES.DEFAULT_VALUE
  - SYSATTRIBUTES.NULLS
  - SYSCOLUMNS.SERVERTYPE
  - SYSDATATYPES.REFREP_TYPENAME
  - SYSDATATYPES.REFREP_TYPESCHEMA
  - SYSDATATYPES.REFREP_LENGTH
  - SYSDATATYPES.REFREP_SCALE
  - SYSDATATYPES.REFREP_CODEPAGE
  - SYSINDEXES.TEXT

    (将来の使用に備える目的だけで視点に含まれていました。)
  - SYSPLANDEP.PUBLICPRIV
  - SYSSECTION.SEQNO
  - SYSTABAUTH.UPDATE_BY_COLS
  - SYSTABAUTH.REF_BY_COLS

- SYSTABLES.MINPDLENGTH
- SYSTABLESPACES.READONLY
- SYSTABLESPACES.REMOVABLEMEDIA

• データ・タイプの変更
- SYSSECTION.SECTION が VARCHAR(3600) から CLOB(10M) に変更
- SYSPLANDEP.COLUSAGE が VARCHAR(3000) FOR BIT DATA から BLOB(5K) に変更

アプリケーション・プログラミング

DB2 ユニバーサル・データベース バージョン 6 での VARCHAR データ・タイプ

WIN	UNIX	OS/2

変更点: VARCHAR (VARGRAPHIC) データ・タイプの最大許容サイズは、バージョン 6 では、 4000 文字 (2 バイト文字では 2000) から、 32672 文字 (2 バイト文字では 16336) になりました。

症状: VARCHAR (VARGRAPHIC) データ・タイプに 4000 バイトの固定長バッファーを使用するアプリケーションでは、 4000 バイトより大きい VARCHAR フィールドを小さいバッファーに取り出す場合に、このバッファーを上書きしたり切り捨てたりする可能性があります。 CLI 関数 `SQLGetTypeInfo()` は、 VARCHAR のサイズとして 32672 を戻すようになりました。表 DDL 内のこの値を使用する CLI アプリケーションは、十分なページ・サイズの表スペースを使用できないことが原因でエラーを受け取る可能性があります。表スペースのページ・サイズについて詳しくは、91ページの『ユーザー表データ』を参照してください。

解決方法: アプリケーションを作成する際には、先に (DESCRIBE ステートメントを使用して) 結果セットの列を記述して、次に DESCRIBE ステートメントから戻される長さに基づくバッファーのサイズを使用することをお勧めします。

DB2 ユニバーサル・データベース バージョン 6 での Java プログラミングの位置指定 UPDATE および DELETE

WIN	UNIX	OS/2

変更点: バージョン 6 で Java をプログラミングする場合、位置指定されている UPDATE および DELETE ステートメントは、カーソル・パッケージをバインドした担当者の許可 ID をデフォルトとして使用します。これは、パッケージを実行する人の許可 ID が使用されたバージョン 5.2 とは異なります。

症状: 位置指定された UPDATE および DELETE ステートメントを含むパッケージが実行しません。パッケージを結合した人の許可 ID に十分な許可がないことが原因です。

解決方法: パッケージを結合する人の許可 ID には、このパッケージ内の位置指定された UPDATE ステートメントおよび DELETE ステートメントを実行するのに十分な許可が授与されなければなりません。正しい特権を授与し、それからパッケージを再バインドしてください。

DB2 ユニバーサル・データベース バージョン 6 の FOR UPDATE 文節での構文変更

WIN	UNIX	OS/2

変更点: バージョン 5.2 では、SQLJ プログラムで、 SELECT ステートメントに FOR UPDATE 文節を使用して、後続の位置指定 UPDATE ステートメントで更新できる列を識別できるようになっていました。バージョン 6 では構文が変更されています。

症状: SELECT ステートメントに FOR UPDATE 文節が含まれている場合に、エラー・メッセージ SQJ0204E を受け取ります。

解決方法: SELECT ステートメントから FOR UPDATE 文節を除去します。イテレーター宣言文節で、更新可能イテレーターを指定します。たとえば、次のようにします。

```
#sql public iterator DelByName implements sqlj.runtime.ForUpdate(String EmpNo)
   with updateColumns = (salary);
```

どの列が更新可能かを明示的に識別したい場合には、 WITH 文節と共に updateColumns キーワードを使ってこれらを指定します。

位置指定されたイテレーター宣言について詳しくは、 *アプリケーション開発の手引き* を参照してください。

DB2 ユニバーサル・データベース バージョン 6 での文字名サイズ

WIN	UNIX	OS/2

変更点: DB2 ユニバーサル・データベース バージョン 6 は、128 バイトの表、視点、別名、および 30 バイトの列名をサポートします。以前のサポートは、それぞれのエンティティー名につき 18 バイトでした。

USER および CURRENT SCHEMA 特殊レジスターは CHAR(8) でしたが、現在は VARCHAR(128) です。 CURRENT EXPLAIN MODE 特殊レジスターは CHAR(8) でしたが、現在は VARCHAR(254) です。 TYPE_SCHEMA および TABLE_SCHEMA 組み込み関数の出力は、 CHAR(8) でしたが、現在は VARCHAR(128) です。

症状: バージョン 6 より前に開発したアプリケーションが、長さの制限を拡張していないバージョン 6 データベースに対して実行された場合、アプリケーションの動作はまったく変わりません。しかし、長い名前を*使用する* バージョン 6 のデータベースに対してこれらのアプリケーションを実行すると、これらのアプリケーションが作成された方法によっては、何らかの副次作用が発生する可能性があります。

以下は、その例です。

- 長さを 18 バイトと定義されたホスト変数に表または列名を (通常は、カタログ視点から) FETCH する既存のアプリケーションを考慮します。バージョン 6 より前は 18 バイトが表名または列名のサイズの限度であったため、このアプリケーションは、 SQLCA の `sqlwarn1` ビットをあえて検査することはありません。切り捨てが行われないことを (誤って) 想定します。
- 表または列名を (通常はカタログ視点から) SQLDA にフェッチするアプリケーションで、 `sqldata` フィールドのサイズが SELECT の DESCRIBE からの `sqllen` フィールドに基づいて割り当てられているアプリケーションについて考慮します。これにより、表または列名のサイズが増えても、正しい (切り捨てられていない) 結果がアプリケーションに戻されることになります。列名が 18 バイトに制限されるという前提で他のアプリケーション論理が実行されると、戻されるそれより長い名前が予期されない方法で処理される (たとえば、長い列名の表示が 18 バイトで切り捨てられる) 可能性があります。
- SQLCA トークン・フィールド (`sqlerrmc`) の制限は 70 バイトであるため、表への行の挿入を試みている既存のアプリケーションが影響を受ける可能性があります。エラー SQL0204N に応答して、そのようなアプリケーションは SQLCA `sqlerrmc` フィールドから表の名前を判別し、それから、オブジェクト名に基づいていくつかの操作を実行します。以前のバージョンの DB2 では、表またはスキーマ ID の制限によって、表名全体が SQLCA に含まれるようになっていました。これはバージョン 6 には当てはまりません。
- バックレベルの API を使用しているアプリケーションでは、表名の最初の 18 バイトしか表示できません。
- (SQLTables() や SQLColumns() などの) スキーマ関数を使用する既存の CLI および ODBC アプリケーションは、 18 バイトより大きい名前をサポートするサーバーに接続するときに影響を受けます。切り捨てについての警告が出されるものの、アプリケーションはこの警告を検査せずに、切り捨てられた名前を使用して処理を継続する可能性があります。

解決方法: このタイプの問題を解決する最善の方法は、アプリケーションをコーディングしなおして、より長い表および列名を処理できるようにすることです。または、これらのアプリケーションが 18 バイトを超える名前を使用するバージョン 6 データベースに対して実行されないようにします。

DB2 ユニバーサル・データベース バージョン 6 での PC/IXF の形式変更

WIN	UNIX	OS/2

変更点: DB2 ユニバーサル・データベース バージョン 6 は、128 バイトの表、視点、別名、および 30 バイトの列名をサポートします。以前のサポートは、それぞれのエンティティー名につき 18 バイトでした。

症状: DB2 ユニバーサル・データベース バージョン 5 クライアントは、 DB2 ユニバーサル・データベース バージョン 6 クライアントによってエクスポートされた PC/IXF をインポートできません (エラー SQL3059N)。また、(DB2 ユニバーサル・データベース バージョン 6 クライアントからエクスポートされた) PC/IXF ファイルは、DB2 ユニバーサル・データベース バージョン 5 データベースにロードできません (エラー SQL3059N)。

解決方法: PC/IXF データをインポートまたはロードするときには、互換性のある DB2 ユニバーサル・データベースのバージョンを使用します。

DB2 ユニバーサル・データベース バージョン 6 での非ダブル SQLVAR の SQLNAME

WIN	UNIX	OS/2

変更点: DB2 ユニバーサル・データベース バージョン 6 は、30 バイトの列名をサポートします。以前のサポートは 18 バイトでした。バージョン 5 で説明されている動作では、非ダブルの SQLVAR の SQLNAME フィールドの 30 番目のバイトに ″0xFF″ が入るようになっていました。しかし、システムの生成する名前、および ″AS″ 文節でユーザーの指定した列名については、 30 番目のバイトに ″0x00″ が入ります。

バージョン 6 では、システムが生成した名前の場合にのみ、 30 番目のバイトに ″0xFF″ が戻されます。

症状: ユーザー指定の名前かシステム生成の名前かを判別するために SQLNAME フィールドの 30 番目のバイトを調べるアプリケーションでは、ユーザー指定の列名の長さが 30 文字の場合、予期しない論理チェックを受け取る可能性があります。これは、めったに起きません。

解決方法: SQLNAME フィールドの長さが 30 バイトより短い場合には、これらのアプリケーションを変更して、 SQLNAME フィールドの 30 番目のバイトで ″0xFF″ の検査だけを実行するようにできます。この場合、名前はユーザー生成の名前になります。

DB2 ユニバーサル・データベース バージョン 6 で使用されなくなった DB2 CLI/ODBC 構成キーワード

WIN		

変更点: DB2 UDB の新しいバージョンにマイグレーションする際に、 db2cli.ini ファイルで一連のオプショナル・キーワードを指定して、 DB2 CLI/ODBC ドライバーの動作を変更することができます。

バージョン 6 では、TRANSLATEDLL および TRANSLATEOPTION キーワードは廃止されました。

症状: これらのキーワードは、存在していても無視されます。これらの設定値が除去されたために、動作が変わる場合があります。

解決方法: 有効なパラメーターの最新のリストを調べて、自分の環境にとって適切なキーワードと設定値を決定してください。これらのキーワードについては、*コール・レベル・インターフェースの手引きおよび解説書* を参照してください。

DB2 ユニバーサル・データベース バージョン 6 でのイベント・モニターの出力ストリーム形式

WIN	UNIX	OS/2

変更点: イベント・モニターの出力ストリームには、バージョン管理がありません。結果として、18 バイトより大きい表名のサポートを追加すると、出力ストリーム形式へのマイグレーションが必要になります。

症状: イベント・モニターの出力ストリームを解析するアプリケーションは、正しく機能しなくなります。

解決方法: 以下の 2 つから選択できます。

- 新しいデータ・ストリームを使用するようにアプリケーションを更新します。
- レジストリー変数を次のように設定します。

  ```
  DB2OLDEVMON=evmonname1,evmonname2,...
  ```

  ここで、*evmonname* は古いデータ形式で作成したいイベント・モニターの名前です。古い形式では、イベント・モニターの新しいフィールドにはアクセスできないことに注意してください。

SQL

DB2 ユニバーサル・データベース バージョン 6 での DATALINK 列

	UNIX	

変更点: DB2 ユニバーサル・データベース バージョン 6 に挿入される DATALINK 値には、列値記述子に 4 バイトのスペースが余分に必要になります。

症状: バージョン 5.2 で作成された DATALINK 列を更新するとき、新しい列値を保管するためにデータ・ページに 4 バイト追加する必要があります。結果として、この更新を完了するのに必要なスペースがデータ・ページにないために、これが新しいページに移動される可能性があります。このために、更新によってスペースが使い尽くされてしまいます。

解決方法: 更新できるように、システムにさらにスペースを追加する必要があります。

DB2 ユニバーサル・データベース バージョン 6 での SYSFUN ストリング関数シグニチャー

WIN	UNIX	OS/2

変更点: SYSFUN スキーマにある多くのストリング関数には、 SYSIBM スキーマ (組み込み関数) で定義された改良済みのバージョンがあります。関数名は、LCASE、LTRIM、RTRIM、および UCASE です。

症状: ステートメントを準備したり視点を作成する際には、これらの関数のいずれかから戻されるデータ・タイプが、バージョン 6 では異なっている可能性があります。これが発生する原因は、(SYSIBM スキーマにある) 組み込み関数が通常、 SYSFUN スキーマにある関数が解決されるよりも前に解決されるためです。

解決方法: 処置は必要ありません。通常、組み込み関数は SYSFUN スキーマにある関数よりも優先的に使用されます。以前のバージョンの動作を復元するには、(SYSFUN が SYSIBM に先行するように) SQL パスを切り替えます。ただし、これによってパフォーマンスは低下します。または、関数名をスキーマ名 SYSFUN で修飾することによって、以前のバージョンの関数を呼び出すこともできます。

これらの関数を参照するマイグレーション済みのパッケージ、視点、要約表、トリガー、および制約は、パッケージを明示的にバインドしたり、視点、要約表、トリガー、または制約を作成し直すなどの明示的なアクションを実行しないかぎり、 SYSFUN スキーマからのバージョンを使用し続けます。

SET CONSTRAINTS から SET INTEGRITY への変更

WIN	UNIX	OS/2

変更点: SET CONSTRAINTS ステートメントは SET INTEGRITY ステートメントに置き換えられました。互換性については、どちらのステートメントも DB2 バージョン 6 およびバージョン 7 で受け入れられます。

DB2 ユニバーサル・データベース バージョン 6 での新しい保全状態での SYSTABLE 列の変更

WIN	UNIX	OS/2

変更点: SET INTEGRITY ... OFF ステートメントが実行されるときの、SYSCAT.TABLES の CONST_CHECKED 列にある ″U″ 状態の変化の仕方が変わりました。

症状: バージョン 6 よりも前は、 SET INTEGRITY ... OFF ステートメントが実行されるときに、 CONST_CHECKED 列の ″U″ 状態が ″N″ 状態に変化しました。現在では、″U″ 状態は ″W″ 状態に変化します。

解決方法: 処置は必要ありません。 CONST_CHECKED 列の新しい ″W″ 状態を使用して、制約タイプが以前にユーザーによって検査されたこと、および表にある一部のデータは保全性の検査をする必要があることを示します。

″N″ 状態によって、データベース・マネージャーが検査していない古いデータが存在するかどうかをはっきり確認することはできません。これに続けて SET INTEGRITY ... IMMEDIATE CHECKED INCREMENTAL ステートメントが発行されると、データベース・マネージャーは必ずエラーを戻します。これは、新しい変更のみが検査されたためデータ保全性が保証できないからです。一方、(INCREMENTAL オプションが指定されている場合に) ″W″ 状態を ″U″ 状態に戻して、引き続きユーザーが表のデータの保全性を保つ責任があることを示すこともできます。 INCREMENTAL オプションが指定されていない場合には、データベース・マネージャーは完全処理を選択し、 ″W″ 状態を ″Y″ 状態に変更して、データ保全を担当することを再び想定します。

データベースのセキュリティーと調整

DB2 ユニバーサル・データベース バージョン 6 でクライアントを使用してデータベースを作成する

WIN	UNIX	OS/2

変更点: データベースを作成するためにクライアントにより使用される方式。

症状: バックレベルのクライアントを使用してデータベースを作成するとエラーが発生します。

解決方法: クライアントを使用してデータベースを作成する場合には、クライアントとサーバーが同じレベルの DB2 コードを実行するようにします。

DB2 ユニバーサル・データベース バージョン 6 の階層で必要な SELECT 特権

WIN 32 ビット	UNIX	OS/2

変更点: (表に) ONLY キーワードを指定すると、ユーザーには、指定されているタイプ付き表のすべての副表に対する SELECT 特権が必要になりました。同様に、(視点に) ONLY キーワードを指定すると、ユーザーには、指定されているタイプ付き表のすべての副視点に対する SELECT 特権が必要になりました。以前のバージョンの DB2 では、指定された表または視点のみの SELECT 特権が必要でした。

症状: 症状として起こる可能性があるのは、以下の 2 つです。

- 許可エラー (SQLCODE -551、SQLSTATE 42501) は、 FROM 文節に ONLY キーワードを指定する SQL ステートメントを含むパッケージを再バインドする際に、指定されたタイプ付き表 (または視点) の副表において、パッケージがバインドされた許可 ID に SELECT 特権が欠落している場合に起きます。
- 視点またはトリガーの定義に、 FROM 文節の ONLY キーワードが含まれている場合には、この視点またはトリガーは正常に実行を継続します。ただし、指定された表 (または視点) のすべての副表に対する SELECT 特権が作成者にないかぎり、視点またはトリガーの定義を使って新しい視点またはトリガーを作成することができなくなりました。

解決方法: パッケージの再バインドや新しい視点およびトリガーの作成に必要な許可 ID に、 ONLY キーワードに続いて指定された表 (および視点) のすべての副表 (および副視点) に対する SELECT 特権を与える必要があります。

DB2 ユニバーサル・データベース バージョン 6 で使用されなくなったプロファイル登録変数および環境変数

WIN	UNIX	OS/2

変更点: 以下のプロファイル・レジストリーまたは環境変数は使用されなくなりました。

- DB2_VECTOR

解決方法: これらの変数は必要でなくなりました。

ユーティリティーとツール

DB2 ユニバーサル・データベース バージョン 6 での現行の Explain モード

WIN	UNIX	OS/2

変更点: ″CURRENT EXPLAIN MODE″ 特殊レジスターのタイプは、 CHAR(8) から VARCHAR(254) に変更されました。

症状: アプリケーションがタイプを CHAR(8) のままと想定している場合には、値が 254 バイトから 8 バイトに切り捨てられる可能性があります。

解決方法: 特殊レジスターを読み取るすべてのホスト変数のタイプを、 CHAR(8) から VARCHAR(254) に再定義します。

この変更は、″CURRENT EXPLAIN MODE″ 特殊レジスターの 2 つの新しい値を使用できるようにするために必要です。新しい値は ″EVALUATE INDEXES″ および ″RECOMMEND INDEXES″ です。

DB2 ユニバーサル・データベース バージョン 6 での USING および SORT BUFFER パラメーター

WIN	UNIX	OS/2

変更点: バージョン 6 では、LOAD コマンドの USING および SORT BUFFER パラメーターはサポートされなくなりました。これらのパラメーターは無視されます。

症状: パラメーター USING および SORT BUFFER はもはやサポートされず、ロード・ユーティリティーはこれらを無視するという警告メッセージが戻されます。

解決方法: 警告メッセージを無視します。詳細については、*データ移動ユーティリティー手引きおよび解説書* を参照してください。

接続性と共存

DB2 ユニバーサル・データベース バージョン 6 での RUMBA から PCOMM への置換

WIN		

変更点: バージョン 6 において、Windows NT、Windows 98、および Windows 95 では RUMBA が PCOMM に置換されました (Windows 3.1 では RUMBA のままです)。

症状: なし。

解決方法: なし。

構成パラメーター

使用されなくなったデータベース構成パラメーター

WIN	UNIX	OS/2

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。

- DL_NUM_BACKUP (NUM_DB_BACKUP データベース構成パラメーターにより置換)

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。

付録D. 各国語サポート (NLS)

ここでは、DB2 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国や地域、言語、およびコード・ページ (コード・セット) に関する情報や、ユーザーのデータベースおよびアプリケーションで DB2 NLS 機能を構成し、使用する方法などが含まれています。

各国語版

バージョン 7 の各国語版は、英語、フランス語、ドイツ語、イタリア語、スペイン語、ブラジル・ポルトガル語、日本語、韓国語、中国語 (簡体字)、中国語 (繁体字)、デンマーク語、フィンランド語、ノルウェー語、スウェーデン語、チェコ語、オランダ語、ハンガリー語、ポーランド語、トルコ語、ロシア語、ブルガリア語、スロベニア語で入手できます。

国 / 地域別コードとコード・ページのサポート

226ページの表28 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国 / 地域別コードとコード・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、この表の各列を説明します。

- **Code Page (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。
- **Group (グループ)** は、コード・ページが単一バイト (「S」)、または 2 バイト (「D」) のどちらかを示します。「-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 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。
- **Code Set (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 コード・ページにマップされます。
- **Tr. (テリトリー)** は、テリトリー ID を表します。
- **Country/Region Code (国 / 地域別コード)** 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。
- **Locale (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。
- **OS (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。
- **Country/Region Name (国 / 地域名)** は、地域や国の名前を示します。

表 28. サポートされる言語およびコード・セット

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
437	S-1	IBM-437	AL	355	-	OS2	アルバニア
850	S-1	IBM-850	AL	355	-	OS2	アルバニア
819	S-1	ISO8859-1	AL	355	sq_AL	AIX	アルバニア
850	S-1	IBM-850	AL	355	Sq_AL	AIX	アルバニア
819	S-1	iso88591	AL	355	-	HP	アルバニア
1051	S-1	roman8	AL	355	-	HP	アルバニア
819	S-1	ISO8859-1	AL	355	-	Sun	アルバニア
1252	S-1	1252	AL	355	-	WIN	アルバニア
1275	S-1	1275	AL	355	-	Mac	アルバニア
37	S-1	IBM-37	AL	355	-	HOST	アルバニア
1140	S-1	IBM-1140	AL	355	-	HOST	アルバニア
864	S-6	IBM-864	AA	785	-	OS2	アラブ諸国
1046	S-6	IBM-1046	AA	785	Ar_AA	AIX	アラブ諸国
1089	S-6	ISO8859-6	AA	785	ar_AA	AIX	アラブ諸国
1089	S-6	iso88596	AA	785	ar_SA.iso88596	HP	アラブ諸国
1256	S-6	1256	AA	785	-	WIN	アラブ諸国
420	S-6	IBM-420	AA	785	-	HOST	アラブ諸国
437	S-1	IBM-437	AU	61	-	OS2	オーストラリア
850	S-1	IBM-850	AU	61	-	OS2	オーストラリア
819	S-1	ISO8859-1	AU	61	en_AU	AIX	オーストラリア
850	S-1	IBM-850	AU	61	En_AU	AIX	オーストラリア
819	S-1	iso88591	AU	61	-	HP	オーストラリア
1051	S-1	roman8	AU	61	-	HP	オーストラリア
819	S-1	ISO8859-1	AU	61	en_AU	Sun	オーストラリア
819	S-1	ISO8859-1	AU	61	en_AU	SCO	オーストラリア
1252	S-1	1252	AU	61	-	WIN	オーストラリア
1275	S-1	1275	AU	61	-	Mac	オーストラリア
37	S-1	IBM-37	AU	61	-	HOST	オーストラリア
1140	S-1	IBM-1140	AU	61	-	HOST	オーストラリア
437	S-1	IBM-437	AT	43	-	OS2	オーストリア
850	S-1	IBM-850	AT	43	-	OS2	オーストリア
819	S-1	ISO8859-1	AT	43	ge_AT	AIX	オーストリア
850	S-1	IBM-850	AT	43	Ge_AT	AIX	オーストリア
819	S-1	iso88591	AT	43	-	HP	オーストリア
1051	S-1	roman8	AT	43	-	HP	オーストリア
819	S-1	ISO8859-1	AT	43	de_AT	SCO	オーストリア
819	S-1	ISO-8859-1	AT	43	de_AT	Linux	オーストリア
819	S-1	ISO8859-1	AT	43	de_AT	Sun	オーストリア
1252	S-1	1252	AT	43	-	WIN	オーストリア
1275	S-1	1275	AT	43	-	Mac	オーストリア
37	S-1	IBM-37	AT	43	-	HOST	オーストリア
1140	S-1	IBM-1140	AT	43	-	HOST	オーストリア

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
915	S-5	ISO8859-5	BY	375	-	OS2	ベラルーシ
915	S-5	ISO8859-5	BY	375	be_BY	AIX	ベラルーシ
1131	S-5	IBM-1131	BY	375	-	OS2	ベラルーシ
1251	S-5	1251	BY	375	-	WIN	ベラルーシ
1283	S-5	1283	BY	375	-	Mac	ベラルーシ
1025	S-5	IBM-1025	BY	375	-	HOST	ベラルーシ
274	S-1	IBM-274	BE	32	-	HOST	ベルギー
437	S-1	IBM-437	BE	32	-	OS2	ベルギー
850	S-1	IBM-850	BE	32	-	OS2	ベルギー
819	S-1	ISO8859-1	BE	32	nl_BE	AIX	ベルギー
850	S-1	IBM-850	BE	32	Nl_BE	AIX	ベルギー
819	S-1	iso88591	BE	32	-	HP	ベルギー
819	S-1	ISO8859-1	BE	32	fr_BE	SCO	ベルギー
819	S-1	ISO8859-1	BE	32	nl_BE	SCO	ベルギー
819	S-1	ISO-8859-1	BE	32	nl_BE	Linux	ベルギー
819	S-1	ISO8859-1	BE	32	nl_BE	Sun	ベルギー
1252	S-1	1252	BE	32	-	WIN	ベルギー
1275	S-1	1275	BE	32	-	Mac	ベルギー
500	S-1	IBM-500	BE	32	-	HOST	ベルギー
1148	S-1	IBM-1148	BE	32	-	HOST	ベルギー
855	S-5	IBM-855	BG	359	-	OS2	ブルガリア
915	S-5	ISO8859-5	BG	359	-	OS2	ブルガリア
915	S-5	ISO8859-5	BG	359	bg_BG	AIX	ブルガリア
915	S-5	iso88595	BG	359	bg_BG.iso88595	HP	ブルガリア
1251	S-5	1251	BG	359	-	WIN	ブルガリア
1283	S-5	1283	BG	359	-	Mac	ブルガリア
1025	S-5	IBM-1025	BG	359	-	HOST	ブルガリア
850	S-1	IBM-850	BR	55	-	OS2	ブラジル
850	S-1	IBM-850	BR	55	-	AIX	ブラジル
819	S-1	ISO8859-1	BR	55	pt_BR	AIX	ブラジル
819	S-1	ISO8859-1	BR	55	-	HP	ブラジル
819	S-1	ISO8859-1	BR	55	pt_BR	SCO	ブラジル
819	S-1	ISO8859-1	BR	55	pt_BR	Sun	ブラジル
819	S-1	ISO-8859-1	BR	55	pt_BR	Linux	ブラジル
1252	S-1	1252	BR	55	-	WIN	ブラジル
37	S-1	IBM-37	BR	55	-	HOST	ブラジル
1140	S-1	IBM-1140	BR	55	-	HOST	ブラジル

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
850	S-1	IBM-850	CA	1	-	OS2	カナダ
850	S-1	IBM-850	CA	1	En_CA	AIX	カナダ
819	S-1	ISO8859-1	CA	1	en_CA	AIX	カナダ
819	S-1	iso88591	CA	1	fr_CA.iso88591	HP	カナダ
1051	S-1	roman8	CA	1	fr_CA.roman8	HP	カナダ
819	S-1	ISO8859-1	CA	1	en_CA	SCO	カナダ
819	S-1	ISO8859-1	CA	1	fr_CA	SCO	カナダ
819	S-1	ISO8859-1	CA	1	en_CA	Sun	カナダ
819	S-1	ISO8859-1	CA	1	en_CA	Sun	カナダ
819	S-1	ISO-8859-1	CA	1	en_CA	Linux	カナダ
1252	S-1	1252	CA	1	-	WIN	カナダ
1275	S-1	1275	CA	1	-	Mac	カナダ
37	S-1	IBM-37	CA	1	-	HOST	カナダ
1140	S-1	IBM-1140	CA	1	-	HOST	カナダ
863	S-1	IBM-863	CA	2	-	OS2	カナダ(フランス語)
1381	D-4	IBM-1381	CN	86	-	OS2	中華人民共和国
1386	D-4	GBK	CN	86	-	OS2	中華人民共和国
1383	D-4	IBM-eucCN	CN	86	zh_CN	AIX	中華人民共和国
1386	D-4	GBK	CN	86	Zh_CN.GBK	AIX	中華人民共和国
1383	D-4	hp15CN	CN	86	zh_CN.hp15CN	HP	中華人民共和国
1383	D-4	eucCN	CN	86	zh_CN	SCO	中華人民共和国
1383	D-4	eucCN	CN	86	zh_CN.eucCN	SCO	中華人民共和国
1383	D-4	gb2312	CN	86	zh	Sun	中華人民共和国
1381	D-4	IBM-1381	CN	86	-	WIN	中華人民共和国
1386	D-4	GBK	CN	86	-	WIN	中華人民共和国
935	D-4	IBM-935	CN	86	-	HOST	中華人民共和国
1388	D-4	IBM-1388	CN	86	-	HOST	中華人民共和国
5488**	D-4		CN	86	-		中華人民共和国 (PRC (拡張中国語 (簡体字)))
** コード・ページ 5488 は、LOAD または IMPORT ユーティリティーによるコード・ページ 5488 から DB2 ユニコード・データベースへのデータの移動、あるいは EXPORT ユーティリティーによる DB2 ユニコード・データベースからコード・ページ 5488 へのデータの移動にのみ使用できます。詳細については、Version 7.2 FixPak 4 Release Notes の データ移動ユーティリティー手引きおよび解説書のセクションを参照してください。							
852	S-2	IBM-852	HR	385	-	OS2	クロアチア
912	S-2	ISO8859-2	HR	385	hr_HR	AIX	クロアチア
912	S-2	iso88592	HR	385	hr_HR.iso88592	HP	クロアチア
912	S-2	ISO8859-2	HR	385	hr_HR.ISO8859-2	SCO	クロアチア
912	S-2	ISO-8859-2	HR	385	hr_HR	Linux	クロアチア
1250	S-2	1250	HR	385	-	WIN	クロアチア
1282	S-2	1282	HR	385	-	Mac	クロアチア
870	S-2	IBM-870	HR	385	-	HOST	クロアチア

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
852	S-2	IBM-852	CZ	421	-	OS2	チェコ共和国
912	S-2	ISO8859-2	CZ	421	cs_CZ	AIX	チェコ共和国
912	S-2	iso88592	CZ	421	cs_CZ.iso88592	HP	チェコ共和国
912	S-2	ISO8859-2	CZ	421	cs_CZ.ISO8859-2	SCO	チェコ共和国
912	S-2	ISO-8859-2	CZ	421	cs_CZ	Linux	チェコ共和国
1250	S-2	1250	CZ	421	-	WIN	チェコ共和国
1282	S-2	1282	CZ	421	-	Mac	チェコ共和国
870	S-2	IBM-870	CZ	421	-	HOST	チェコ共和国
850	S-1	IBM-850	DK	45	-	OS2	デンマーク
819	S-1	ISO8859-1	DK	45	da_DK	AIX	デンマーク
850	S-1	IBM-850	DK	45	Da_DK	AIX	デンマーク
819	S-1	iso88591	DK	45	da_DK.iso88591	HP	デンマーク
1051	S-1	roman8	DK	45	da_DK.roman8	HP	デンマーク
819	S-1	ISO8859-1	DK	45	da	SCO	デンマーク
819	S-1	ISO8859-1	DK	45	da_DA	SCO	デンマーク
819	S-1	ISO8859-1	DK	45	da_DK	SCO	デンマーク
819	S-1	ISO8859-1	DK	45	da	Sun	デンマーク
819	S-1	ISO8859-1	DK	45	da	Sun	デンマーク
819	S-1	ISO-8859-1	DK	45	da_DK	Linux	デンマーク
1252	S-1	1252	DK	45	-	WIN	デンマーク
1275	S-1	1275	DK	45	-	Mac	デンマーク
277	S-1	IBM-277	DK	45	-	HOST	デンマーク
1142	S-1	IBM-1142	DK	45	-	HOST	デンマーク
922	S-10	IBM-922	EE	372	-	OS2	エストニア
922	S-10	IBM-922	EE	372	Et_EE	AIX	エストニア
1257	S-10	1257	EE	372	-	WIN	エストニア
1122	S-10	IBM-1122	EE	372	-	HOST	エストニア
437	S-1	IBM-437	FI	358	-	OS2	フィンランド
850	S-1	IBM-850	FI	358	-	OS2	フィンランド
819	S-1	ISO8859-1	FI	358	fi_FI	AIX	フィンランド
850	S-1	IBM-850	FI	358	Fi_FI	AIX	フィンランド
819	S-1	iso88591	FI	358	fi_FI.iso88591	HP	フィンランド
819	S-1	ISO8859-1	FI	358	fi	SCO	フィンランド
819	S-1	ISO8859-1	FI	358	fi_FI	SCO	フィンランド
819	S-1	ISO8859-1	FI	358	sv_FI	SCO	フィンランド
819	S-1	ISO8859-1	FI	358	-	Sun	フィンランド
819	S-1	ISO-8859-1	FI	358	fi_FI	Linux	フィンランド
1051	S-1	roman8	FI	358	-	HP	フィンランド
1252	S-1	1252	FI	358	-	WIN	フィンランド
1275	S-1	1275	FI	358	-	Mac	フィンランド
278	S-1	IBM-278	FI	358	-	HOST	フィンランド
1143	S-1	IBM-1143	FI	358	-	HOST	フィンランド

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
855	S-5	IBM-855	MK	389	-	OS2	FYR マケドニア
915	S-5	ISO8859-5	MK	389	-	OS2	FYR マケドニア
915	S-5	ISO8859-5	MK	389	mk_MK	AIX	FYR マケドニア
915	S-5	iso88595	MK	389	-	HP	FYR マケドニア
1251	S-5	1251	MK	389	-	WIN	FYR マケドニア
1283	S-5	1283	MK	389	-	Mac	FYR マケドニア
1025	S-5	IBM-1025	MK	389	-	HOST	FYR マケドニア
437	S-1	IBM-437	FR	33	-	OS2	フランス
850	S-1	IBM-850	FR	33	-	OS2	フランス
819	S-1	ISO8859-1	FR	33	fr_FR	AIX	フランス
850	S-1	IBM-850	FR	33	Fr_FR	AIX	フランス
819	S-1	iso88591	FR	33	fr_FR.iso88591	HP	フランス
1051	S-1	roman8	FR	33	fr_FR.roman8	HP	フランス
819	S-1	ISO8859-1	FR	33	fr	Sun	フランス
819	S-1	ISO8859-1	FR	33	fr	SCO	フランス
819	S-1	ISO8859-1	FR	33	fr_FR	SCO	フランス
819	S-1	ISO-8859-1	FR	33	fr_FR	Linux	フランス
1252	S-1	1252	FR	33	-	WIN	フランス
1275	S-1	1275	FR	33	-	Mac	フランス
297	S-1	IBM-297	FR	33	-	HOST	フランス
1147	S-1	IBM-1147	FR	33	-	HOST	フランス
437	S-1	IBM-437	DE	49	-	OS2	ドイツ
850	S-1	IBM-850	DE	49	-	OS2	ドイツ
819	S-1	ISO8859-1	DE	49	de_DE	AIX	ドイツ
850	S-1	IBM-850	DE	49	De_DE	AIX	ドイツ
819	S-1	iso88591	DE	49	de_DE.iso88591	HP	ドイツ
1051	S-1	roman8	DE	49	de_DE.roman8	HP	ドイツ
819	S-1	ISO8859-1	DE	49	de	SCO	ドイツ
819	S-1	ISO8859-1	DE	49	de_DE	SCO	ドイツ
819	S-1	ISO8859-1	DE	49	de	Sun	ドイツ
819	S-1	ISO-8859-1	DE	49	de_DE	Linux	ドイツ
1252	S-1	1252	DE	49	-	WIN	ドイツ
1275	S-1	1275	DE	49	-	Mac	ドイツ
273	S-1	IBM-273	DE	49	-	HOST	ドイツ
1141	S-1	IBM-1141	DE	49	-	HOST	ドイツ
819	S-1	ISO8859-1	DE	49	De_DE.88591	SINIX	ドイツ
819	S-1	ISO8859-1	DE	49	De_DE.6937	SINIX	ドイツ

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
813	S-7	ISO8859-7	GR	30	-	OS2	ギリシャ
869	S-7	IBM-869	GR	30	-	OS2	ギリシャ
813	S-7	ISO8859-7	GR	30	el_GR	AIX	ギリシャ
813	S-7	iso88597	GR	30	el_GR.iso88597	HP	ギリシャ
813	S-7	ISO8859-7	GR	30	el_GR.ISO8859-7	SCO	ギリシャ
813	S-7	ISO-8859-7	GR	30	gr_GR	Linux	ギリシャ
737	S-7	737	GR	30	-	WIN	ギリシャ
1253	S-7	1253	GR	30	-	WIN	ギリシャ
1280	S-7	1280	GR	30	-	Mac	ギリシャ
423	S-7	IBM-423	GR	30	-	HOST	ギリシャ
875	S-7	IBM-875	GR	30	-	HOST	ギリシャ
852	S-2	IBM-852	HU	36	-	OS2	ハンガリー
912	S-2	ISO8859-2	HU	36	hu_HU	AIX	ハンガリー
912	S-2	iso88592	HU	36	hu_HU.iso88592	HP	ハンガリー
912	S-2	ISO8859-2	HU	36	hu_HU.ISO8859-2	SCO	ハンガリー
912	S-2	ISO-8859-2	HU	36	hu_HU	Linux	ハンガリー
1250	S-2	1250	HU	36	-	WIN	ハンガリー
1282	S-2	1282	HU	36	-	Mac	ハンガリー
870	S-2	IBM-870	HU	36	-	HOST	ハンガリー
850	S-1	IBM-850	IS	354	-	OS2	アイスランド
819	S-1	ISO8859-1	IS	354	is_IS	AIX	アイスランド
850	S-1	IBM-850	IS	354	Is_IS	AIX	アイスランド
819	S-1	iso88591	IS	354	is_IS.iso88591	HP	アイスランド
1051	S-1	roman8	IS	354	is_IS.roman8	HP	アイスランド
819	S-1	ISO8859-1	IS	354	is	SCO	アイスランド
819	S-1	ISO8859-1	IS	354	is_IS	SCO	アイスランド
819	S-1	ISO8859-1	IS	354	-	Sun	アイスランド
819	S-1	ISO-8859-1	IS	354	is_IS	Linux	アイスランド
1252	S-1	1252	IS	354	-	WIN	アイスランド
1275	S-1	1275	IS	354	-	Mac	アイスランド
871	S-1	IBM-871	IS	354	-	HOST	アイスランド
1149	S-1	IBM-1149	IS	354	-	HOST	アイスランド

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
437	S-1	IBM-437	IE	353	-	OS2	アイルランド
850	S-1	IBM-850	IE	353	-	OS2	アイルランド
819	S-1	ISO8859-1	IE	353	en_IE	AIX	アイルランド
850	S-1	IBM-850	IE	353	En_IE	AIX	アイルランド
819	S-1	iso88591	IE	353	-	HP	アイルランド
1051	S-1	roman8	IE	353	-	HP	アイルランド
819	S-1	ISO8859-1	IE	353	en_IE	Sun	アイルランド
819	S-1	ISO8859-1	IE	353	en_IE.ISO8859-1	SCO	アイルランド
819	S-1	ISO-8859-1	IE	353	en_IE	Linux	アイルランド
1252	S-1	1252	IE	353	-	WIN	アイルランド
1275	S-1	1275	IE	353	-	Mac	アイルランド
285	S-1	IBM-285	IE	353	-	HOST	アイルランド
1146	S-1	IBM-1146	IE	353	-	HOST	アイルランド
806	S-12	IBM-806	IN	91	hi_IN	-	インド
1137	S-12	IBM-1137	IN	91	-	HOST	インド
862	S-8	IBM-862	IL	972	-	OS2	イスラエル
916	S-8	ISO8859-8	IL	972	iw_IL	AIX	イスラエル
856	S-8	IBM-856	IL	972	Iw_IL	AIX	イスラエル
916	S-8	ISO-8859-8	IL	972	iw_IL	Linux	イスラエル
1255	S-8	1255	IL	972	-	WIN	イスラエル
424	S-8	IBM-424	IL	972	-	HOST	イスラエル
437	S-1	IBM-437	IT	39	-	OS2	イタリア
850	S-1	IBM-850	IT	39	-	OS2	イタリア
819	S-1	ISO8859-1	IT	39	it_IT	AIX	イタリア
850	S-1	IBM-850	IT	39	It_IT	AIX	イタリア
819	S-1	iso88591	IT	39	it_IT.iso88591	HP	イタリア
1051	S-1	roman8	IT	39	it_IT.roman8	HP	イタリア
819	S-1	ISO8859-1	IT	39	it	SCO	イタリア
819	S-1	ISO8859-1	IT	39	it_IT	SCO	イタリア
819	S-1	ISO8859-1	IT	39	it	Sun	イタリア
819	S-1	ISO-8859-1	IT	39	it_IT	Linux	イタリア
1252	S-1	1252	IT	39	-	WIN	イタリア
1275	S-1	1275	IT	39	-	Mac	イタリア
280	S-1	IBM-280	IT	39	-	HOST	イタリア
1144	S-1	IBM-1144	IT	39	-	HOST	イタリア

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
932	D-1	IBM-932	JP	81	-	OS2	日本
942	D-1	IBM-942	JP	81	-	OS2	日本
943	D-1	IBM-943	JP	81	-	OS2	日本
954	D-1	IBM-eucJP	JP	81	ja_JP	AIX	日本
943*	D-1	IBM-943	JP	81	Ja_JP	AIX	日本
954	D-1	eucJP	JP	81	ja_JP.eucJP	HP	日本
5039	D-1	SJIS	JP	81	ja_JP.SJIS	HP	日本
954	D-1	eucJP	JP	81	ja	SCO	日本
954	D-1	eucJP	JP	81	ja_JP	SCO	日本
954	D-1	eucJP	JP	81	ja_JP.EUC	SCO	日本
954	D-1	eucJP	JP	81	ja_JP.eucJP	SCO	日本
954	D-1	eucJP	JP	81	ja	Sun	日本
943	D-1	IBM-943	JP	81	ja_JP.PCK	Sun	日本
954	D-1	EUC-JP	JP	81	ja_JP	Linux	日本
943	D-1	IBM-943	JP	81	-	WIN	日本
930	D-1	IBM-930	JP	81	-	HOST	日本
939	D-1	IBM-939	JP	81	-	HOST	日本
5026	D-1	IBM-5026	JP	81	-	HOST	日本
5035	D-1	IBM-5035	JP	81	-	HOST	日本
1390	D-1		JP	81	-	HOST	日本
1399	D-1		JP	81	-	HOST	日本
1394**	D-1		JP	81	-		日本

* 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 になります。

** コード・ページ 1394 は、LOAD または IMPORT ユーティリティーでコード・ページ 1394 から DB2 ユニコード・データベースにデータを移動する場合や、EXPORT ユーティリティーで DB2 ユニコード・データベースからコード・ページ 1394 にデータを移動する場合にのみ使用できます。詳細については、Version 7.2 FixPak 4 Release Notes のデータ移動ユーティリティー 手引きおよび解説書のセクションを参照してください。

949	D-3	IBM-949	KR	82	-	OS2	韓国
970	D-3	IBM-eucKR	KR	82	ko_KR	AIX	韓国
970	D-3	eucKR	KR	82	ko_KR.eucKR	HP	韓国
970	D-3	eucKR	KR	82	ko_KR.eucKR	SGI	韓国
970	D-3	5601	KR	82	ko	Sun	韓国
1363	D-3	1363	KR	82	-	WIN	韓国
933	D-3	IBM-933	KR	82	-	HOST	韓国
1364	D-3	IBM-1364	KR	82	-	HOST	韓国

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
437	S-1	IBM-437	Lat	3	-	OS2	ラテンアメリカ
850	S-1	IBM-850	Lat	3	-	OS2	ラテンアメリカ
819	S-1	ISO8859-1	Lat	3	-	AIX	ラテンアメリカ
850	S-1	IBM-850	Lat	3	-	AIX	ラテンアメリカ
819	S-1	iso88591	Lat	3	-	HP	ラテンアメリカ
819	S-1	ISO8859-1	Lat	3	-	Sun	ラテンアメリカ
819	S-1	ISO-8859-1	Lat	3	-	Linux	ラテンアメリカ
1051	S-1	roman8	Lat	3	-	HP	ラテンアメリカ
1252	S-1	1252	Lat	3	-	WIN	ラテンアメリカ
1275	S-1	1275	Lat	3	-	Mac	ラテンアメリカ
284	S-1	IBM-284	Lat	3	-	HOST	ラテンアメリカ
1145	S-1	IBM-1145	Lat	3	-	HOST	ラテンアメリカ
921	S-10	IBM-921	LV	371	-	OS2	ラトビア
921	S-10	IBM-921	LV	371	Lv_LV	AIX	ラトビア
1257	S-10	1257	LV	371	-	WIN	ラトビア
1112	S-10	IBM-1112	LV	371	-	HOST	ラトビア
921	S-10	IBM-921	LT	370	-	OS2	リトアニア
921	S-10	IBM-921	LT	370	Lt_LT	AIX	リトアニア
1257	S-10	1257	LT	370	-	WIN	リトアニア
1112	S-10	IBM-1112	LT	370	-	HOST	リトアニア
437	S-1	IBM-437	NL	31	-	OS2	オランダ
850	S-1	IBM-850	NL	31	-	OS2	オランダ
819	S-1	ISO8859-1	NL	31	nl_NL	AIX	オランダ
850	S-1	IBM-850	NL	31	Nl_NL	AIX	オランダ
819	S-1	iso88591	NL	31	nl_NL.iso88591	HP	オランダ
1051	S-1	roman8	NL	31	nl_NL.roman8	HP	オランダ
819	S-1	ISO8859-1	NL	31	nl	SCO	オランダ
819	S-1	ISO8859-1	NL	31	nl_NL	SCO	オランダ
819	S-1	ISO8859-1	NL	31	nl	Sun	オランダ
819	S-1	ISO-8859-1	NL	31	nl_NL	Linux	オランダ
1252	S-1	1252	NL	31	-	WIN	オランダ
1275	S-1	1275	NL	31	-	Mac	オランダ
37	S-1	IBM-37	NL	31	-	HOST	オランダ
1140	S-1	IBM-1140	NL	31	-	HOST	オランダ
850	S-1	IBM-850	NZ	64	-	OS2	ニュージーランド
850	S-1	IBM-850	NZ	64	En_NZ	AIX	ニュージーランド
819	S-1	ISO8859-1	NZ	64	en_NZ	AIX	ニュージーランド
819	S-1	ISO8859-1	NZ	64	-	HP	ニュージーランド
819	S-1	ISO8859-1	NZ	64	en_NZ	SCO	ニュージーランド
819	S-1	ISO8859-1	NZ	64	en_NZ	Sun	ニュージーランド
1252	S-1	1252	NZ	64	-	WIN	ニュージーランド
37	S-1	IBM-37	NZ	64	-	HOST	ニュージーランド
1140	S-1	IBM-1140	NZ	64	-	HOST	ニュージーランド

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
850	S-1	IBM-850	NO	47	-	OS2	ノルウェー
819	S-1	ISO8859-1	NO	47	no_NO	AIX	ノルウェー
850	S-1	IBM-850	NO	47	No_NO	AIX	ノルウェー
819	S-1	iso88591	NO	47	no_NO.iso88591	HP	ノルウェー
1051	S-1	roman8	NO	47	no_NO.roman8	HP	ノルウェー
819	S-1	ISO8859-1	NO	47	no	SCO	ノルウェー
819	S-1	ISO8859-1	NO	47	no_NO	SCO	ノルウェー
819	S-1	ISO8859-1	NO	47	no	Sun	ノルウェー
819	S-1	ISO-8859-1	NO	47	no_NO	Linux	ノルウェー
1252	S-1	1252	NO	47	-	WIN	ノルウェー
1275	S-1	1275	NO	47	-	Mac	ノルウェー
277	S-1	IBM-277	NO	47	-	HOST	ノルウェー
1142	S-1	IBM-1142	NO	47	-	HOST	ノルウェー
852	S-2	IBM-852	PL	48	-	OS2	ポーランド
912	S-2	ISO8859-2	PL	48	pl_PL	AIX	ポーランド
912	S-2	iso88592	PL	48	pl_PL.iso88592	HP	ポーランド
912	S-2	ISO8859-2	PL	48	pl_PL.ISO8859-2	SCO	ポーランド
912	S-2	ISO-8859-2	PL	48	pl_PL	Linux	ポーランド
1250	S-2	1250	PL	48	-	WIN	ポーランド
1282	S-2	1282	PL	48	-	Mac	ポーランド
870	S-2	IBM-870	PL	48	-	HOST	ポーランド
860	S-1	IBM-860	PT	351	-	OS2	ポルトガル
850	S-1	IBM-850	PT	351	-	OS2	ポルトガル
819	S-1	ISO8859-1	PT	351	pt_PT	AIX	ポルトガル
850	S-1	IBM-850	PT	351	Pt_PT	AIX	ポルトガル
819	S-1	iso88591	PT	351	pt_PT.iso88591	HP	ポルトガル
1051	S-1	roman8	PT	351	pt_PT.roman8	HP	ポルトガル
819	S-1	ISO8859-1	PT	351	pt	SCO	ポルトガル
819	S-1	ISO8859-1	PT	351	pt_PT	SCO	ポルトガル
819	S-1	ISO8859-1	PT	351	pt	Sun	ポルトガル
819	S-1	ISO-8859-1	PT	351	pt_PT	Linux	ポルトガル
1252	S-1	1252	PT	351	-	WIN	ポルトガル
1275	S-1	1275	PT	351	-	Mac	ポルトガル
37	S-1	IBM-37	PT	351	-	HOST	ポルトガル
1140	S-1	IBM-1140	PT	351	-	HOST	ポルトガル
852	S-2	IBM-852	RO	40	-	OS2	ルーマニア
912	S-2	ISO8859-2	RO	40	ro_RO	AIX	ルーマニア
912	S-2	iso88592	RO	40	ro_RO.iso88592	HP	ルーマニア
912	S-2	ISO8859-2	RO	40	ro_RO.ISO8859-2	SCO	ルーマニア
912	S-2	ISO-8859-2	RO	40	ro_RO	Linux	ルーマニア
1250	S-2	1250	RO	40	-	WIN	ルーマニア
1282	S-2	1282	RO	40	-	Mac	ルーマニア
870	S-2	IBM-870	RO	40	-	HOST	ルーマニア

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
866	S-5	IBM-866	RU	7	-	OS2	ロシア
915	S-5	ISO8859-5	RU	7	-	OS2	ロシア
915	S-5	ISO8859-5	RU	7	ru_RU	AIX	ロシア
915	S-5	iso88595	RU	7	ru_RU.iso88595	HP	ロシア
915	S-5	ISO8859-5	RU	7	ru_RU.ISO8859-5	SCO	ロシア
915	S-5	ISO-8859-5	RU	7	ru_RU	Linux	ロシア
1251	S-5	1251	RU	7	-	WIN	ロシア
1283	S-5	1283	RU	7	-	Mac	ロシア
1025	S-5	IBM-1025	RU	7	-	HOST	ロシア
855	S-5	IBM-855	SP	381	-	OS2	セルビア / モンテネグロ
915	S-5	ISO8859-5	SP	381	-	OS2	セルビア / モンテネグロ
915	S-5	ISO8859-5	SP	381	sr_SP	AIX	セルビア / モンテネグロ
915	S-5	iso88595	SP	381	-	HP	セルビア / モンテネグロ
1251	S-5	1251	SP	381	-	WIN	セルビア / モンテネグロ
1283	S-5	1283	SP	381	-	Mac	セルビア / モンテネグロ
1025	S-5	IBM-1025	SP	381	-	HOST	セルビア / モンテネグロ
852	S-2	IBM-852	SK	422	-	OS2	スロバキア
912	S-2	ISO8859-2	SK	422	sk_SK	AIX	スロバキア
912	S-2	iso88592	SK	422	sk_SK.iso88592	HP	スロバキア
912	S-2	ISO8859-2	SK	422	sk_SK.ISO8859-2	SCO	スロバキア
1250	S-2	1250	SK	422	-	WIN	スロバキア
1282	S-2	1282	SK	422	-	Mac	スロバキア
870	S-2	IBM-870	SK	422	-	HOST	スロバキア
852	S-2	IBM-852	SI	386	-	OS2	スロベニア
912	S-2	ISO8859-2	SI	386	sl_SI	AIX	スロベニア
912	S-2	iso88592	SI	386	sl_SI.iso88592	HP	スロベニア
912	S-2	ISO8859-2	SI	386	sl_SI.ISO8859-2	SCO	スロベニア
912	S-2	ISO-8859-2	SI	386	sl_SI	Linux	スロベニア
1250	S-2	1250	SI	386	-	WIN	スロベニア
1282	S-2	1282	SI	386	-	Mac	スロベニア
870	S-2	IBM-870	SI	386	-	HOST	スロベニア

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
437	S-1	IBM-437	ZA	27	-	OS2	南アフリカ
850	S-1	IBM-850	ZA	27	-	OS2	南アフリカ
819	S-1	ISO8859-1	ZA	27	en_ZA	AIX	南アフリカ
850	S-1	IBM-850	ZA	27	En_ZA	AIX	南アフリカ
819	S-1	iso88591	ZA	27	-	HP	南アフリカ
1051	S-1	roman8	ZA	27	-	HP	南アフリカ
819	S-1	ISO8859-1	ZA	27	-	Sun	南アフリカ
819	S-1	ISO8859-1	ZA	27	en_ZA.ISO8859-1	SCO	南アフリカ
1252	S-1	1252	ZA	27	-	WIN	南アフリカ
1275	S-1	1275	ZA	27	-	Mac	南アフリカ
285	S-1	IBM-285	ZA	27	-	HOST	南アフリカ
1146	S-1	IBM-1146	ZA	27	-	HOST	南アフリカ
437	S-1	IBM-437	ES	34	-	OS2	スペイン
850	S-1	IBM-850	ES	34	-	OS2	スペイン
819	S-1	ISO8859-1	ES	34	es_ES	AIX	スペイン
850	S-1	IBM-850	ES	34	Es_ES	AIX	スペイン
819	S-1	iso88591	ES	34	es_ES.iso88591	HP	スペイン
1051	S-1	roman8	ES	34	es_ES.roman8	HP	スペイン
819	S-1	ISO8859-1	ES	34	es	Sun	スペイン
819	S-1	ISO8859-1	ES	34	es	SCO	スペイン
819	S-1	ISO8859-1	ES	34	es_ES	SCO	スペイン
819	S-1	ISO-8859-1	ES	34	es_ES	Linux	スペイン
1252	S-1	1252	ES	34	-	WIN	スペイン
1275	S-1	1275	ES	34	-	Mac	スペイン
284	S-1	IBM-284	ES	34	-	HOST	スペイン
1145	S-1	IBM-1145	ES	34	-	HOST	スペイン
437	S-1	IBM-437	SE	46	-	OS2	スウェーデン
850	S-1	IBM-850	SE	46	-	OS2	スウェーデン
819	S-1	ISO8859-1	SE	46	sv_SE	AIX	スウェーデン
850	S-1	IBM-850	SE	46	Sv_SE	AIX	スウェーデン
819	S-1	iso88591	SE	46	sv_SE.iso88591	HP	スウェーデン
1051	S-1	roman8	SE	46	sv_SE.roman8	HP	スウェーデン
819	S-1	ISO8859-1	SE	46	sv	SCO	スウェーデン
819	S-1	ISO8859-1	SE	46	sv_SE	SCO	スウェーデン
819	S-1	ISO8859-1	SE	46	sv	Sun	スウェーデン
819	S-1	ISO-8859-1	SE	46	sv_SE	Linux	スウェーデン
1252	S-1	1252	SE	46	-	WIN	スウェーデン
1275	S-1	1275	SE	46	-	Mac	スウェーデン
278	S-1	IBM-278	SE	46	-	HOST	スウェーデン
1143	S-1	IBM-1143	SE	46	-	HOST	スウェーデン

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
437	S-1	IBM-437	CH	41	-	OS2	スイス
850	S-1	IBM-850	CH	41	-	OS2	スイス
819	S-1	ISO8859-1	CH	41	de_CH	AIX	スイス
850	S-1	IBM-850	CH	41	De_CH	AIX	スイス
819	S-1	iso88591	CH	41	-	HP	スイス
1051	S-1	roman8	CH	41	-	HP	スイス
819	S-1	ISO8859-1	CH	41	de_CH	SCO	スイス
819	S-1	ISO8859-1	CH	41	fr_CH	SCO	スイス
819	S-1	ISO8859-1	CH	41	it_CH	SCO	スイス
819	S-1	ISO8859-1	CH	41	de_CH	Sun	スイス
819	S-1	ISO-8859-1	CH	41	de_CH	Linux	スイス
1252	S-1	1252	CH	41	-	WIN	スイス
1275	S-1	1275	CH	41	-	Mac	スイス
500	S-1	IBM-500	CH	41	-	HOST	スイス
1148	S-1	IBM-1148	CH	41	-	HOST	スイス
938	D-2	IBM-938	TW	88	-	OS2	台湾
948	D-2	IBM-948	TW	88	-	OS2	台湾
950	D-2	big5	TW	88	-	OS2	台湾
950	D-2	big5	TW	88	Zh_TW	AIX	台湾
964	D-2	IBM-eucTW	TW	88	zh_TW	AIX	台湾
950	D-2	big5	TW	88	zh_TW.big5	HP	台湾
964	D-2	eucTW	TW	88	zh_TW.eucTW	HP	台湾
950	D-2	big5	TW	88	big5	Sun	台湾
964	D-2	cns11643	TW	88	zh_TW	Sun	台湾
950	D-2	big5	TW	88	-	WIN	台湾
937	D-2	IBM-937	TW	88	-	HOST	台湾
874	S-20	TIS620-1	TH	66	-	OS2	タイ
874	S-20	TIS620-1	TH	66	Th_TH	AIX	タイ
874	S-20	tis620	TH	66	th_TH.tis620	HP	タイ
874	S-20	TIS620-1	TH	66	-	WIN	タイ
838	S-20	IBM-838	TH	66	-	HOST	タイ
857	S-9	IBM-857	TR	90	-	OS2	トルコ
920	S-9	ISO8859-9	TR	90	tr_TR	AIX	トルコ
920	S-9	iso88599	TR	90	tr_TR.iso88599	HP	トルコ
920	S-9	ISO8859-9	TR	90	tr_TR.ISO8859-9	SCO	トルコ
920	S-9	ISO-8859-9	TR	90	tr_TR	Linux	トルコ
1254	S-9	1254	TR	90	-	WIN	トルコ
1281	S-9	1281	TR	90	-	Mac	トルコ
1026	S-9	IBM-1026	TR	90	-	HOST	トルコ

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名
437	S-1	IBM-437	GB	44	-	OS2	イギリス
850	S-1	IBM-850	GB	44	-	OS2	イギリス
819	S-1	ISO8859-1	GB	44	en_GB	AIX	イギリス
850	S-1	IBM-850	GB	44	En_GB	AIX	イギリス
819	S-1	iso88591	GB	44	en_GB.iso88591	HP	イギリス
1051	S-1	roman8	GB	44	en_GB.roman8	HP	イギリス
819	S-1	ISO8859-1	GB	44	en_UK	Sun	イギリス
819	S-1	ISO8859-1	GB	44	en_GB	SCO	イギリス
819	S-1	ISO8859-1	GB	44	en	SCO	イギリス
819	S-1	ISO-8859-1	GB	44	en_GB	Linux	イギリス
1252	S-1	1252	GB	44	-	WIN	イギリス
1275	S-1	1275	GB	44	-	Mac	イギリス
285	S-1	IBM-285	GB	44	-	HOST	イギリス
1146	S-1	IBM-1146	GB	44	-	HOST	イギリス
819	S-1	88591	GB	44	En_GB.88591	SINIX	イギリス
819	S-1	ISO8859-1	GB	44	En_GB.6937	SINIX	イギリス
1125	S-5	IBM-1125	UA	380	-	OS2	ウクライナ
1124	S-5	IBM-1124	UA	380	uk_UA	AIX	ウクライナ
1251	S-5	1251	UA	380	-	WIN	ウクライナ
1123	S-5	IBM-1123	UA	380	-	HOST	ウクライナ
437	S-1	IBM-437	US	1	-	OS2	アメリカ合衆国
850	S-1	IBM-850	US	1	-	OS2	アメリカ合衆国
819	S-1	ISO8859-1	US	1	en_US	AIX	アメリカ合衆国
850	S-1	IBM-850	US	1	En_US	AIX	アメリカ合衆国
819	S-1	iso88591	US	1	en_US.iso88591	HP	アメリカ合衆国
1051	S-1	roman8	US	1	en_US.roman8	HP	アメリカ合衆国
819	S-1	ISO8859-1	US	1	en_US	Sun	アメリカ合衆国
819	S-1	ISO8859-1	US	1	en_US	SGI	アメリカ合衆国
819	S-1	ISO8859-1	US	1	en_US	SCO	アメリカ合衆国
819	S-1	ISO-8859-1	US	1	en_US	Linux	アメリカ合衆国
1252	S-1	1252	US	1	-	WIN	アメリカ合衆国
1275	S-1	1275	US	1	-	Mac	アメリカ合衆国
37	S-1	IBM-37	US	1	-	HOST	アメリカ合衆国
1140	S-1	IBM-1140	US	1	-	HOST	アメリカ合衆国
1163	S-11	IBM-1163	VN	84	-	OS2	ベトナム
1163	S-11	IBM-1163	VN	84	vi_VN	AIX	ベトナム
1258	S-11	1258	VN	84	-	WIN	ベトナム
1164	S-11	IBM-1164	VN	84	-	HOST	ベトナム

表 28. サポートされる言語およびコード・セット (続き)

コード・ページ	グループ	コード・セット	テリトリー	国 / 地域別コード	ロケール	OS	国 / 地域名

以下は、アラブ諸国 (AA) にマップされます。

```
 ----------------------------------------
    /* Arabic (サウジアラビア) */
    /* Arabic (イラク) */
    /* Arabic (エジプト) */
    /* Arabic (リビア) */
    /* Arabic (アルジェリア) */
    /* Arabic (モロッコ) */
    /* Arabic (チュニジア) */
    /* Arabic (オマーン) */
    /* Arabic (イエメン) */
    /* Arabic (シリア) */
    /* Arabic (ヨルダン) */
    /* Arabic (レバノン) */
    /* Arabic (クウェート) */
    /* Arabic (アラブ首長国連邦) */
    /* Arabic (バーレーン) */
    /* Arabic (カタール) */
```

以下は、英語 (US) にマップされます。

```
 --------------------------------
    /* English (ジャマイカ) */
    /* English (カリブ海) */
```

以下は、ラテンアメリカ (Lat) にマップされます。

```
 ---------------------------------------
    /* Spanish (メキシコ) */
    /* Spanish (グアテマラ) */
    /* Spanish (コスタリカ) */
    /* Spanish (パナマ) */
    /* Spanish (ドミニカ共和国) */
    /* Spanish (ベネズエラ) */
    /* Spanish (コロンビア) */
    /* Spanish (ペルー) */
    /* Spanish (アルゼンチン) */
    /* Spanish (エクアドル) */
    /* Spanish (チリ) */
    /* Spanish (ウルグアイ) */
    /* Spanish (パラグアイ) */
    /* Spanish (ボリビア) */
```

注:

1. Solaris コード・ページ 950 は、IBM 950 の以下の文字をサポートしません。

コード範囲	説明	Sun Big-5	IBM Big-5
C6A1 ～ C8FE	シンボル	予約区域	シンボル
F9D6 ～ F9FE	ETen 拡張	予約区域	ETen 拡張
F286 ～ F9A0	IBM 選択文字	予約区域	IBM 選択

2. DB2 UDB のこのバージョンには、欧州文字サポートがあります。 Microsoft Windows ANSI コード・ページは、欧州文字を 0x80 の位置に組み込んだ Microsoft 社からの最新の定義に基づいて変更されました。この位置は、以前は未定義でした。さらにコード・ページ 850 の定義も、ドットのない i 文字 (0xD5 の位置にある) から欧州文字に置き換えるように変更になりました。 DB2 UDB は、欧州文字サポートを提供するためのデフォルト値として、新しいこれらのコード・ページの定義を使用します。この実装は、現在の DB2 UDB をご使用になっていて、欧州文字サポートを必要とするお客様に適切なデフォルト値であり、他のお客様にも影響を与えません。しかし、これらのコード・ページの以前の定義を継続して使いたい場合は、次のファイルを、インストール終了後にコピーできます。

   - 12520850.cnv
   - 08501252.cnv
   - IBM00850.ucs
   - IBM01252.ucs

   次のディレクトリーから

   ```
   sqllib/conv/alt/
   ```

   次のディレクトリーへ

   ```
   sqllib/conv/
   ```

   既存のファイル `IBM01252.usc` および `IBM00850.ucs` をバックアップしてから、非欧州文字バージョンをそれらに上書きします。ファイルをコピーした後は、DB2 UDB から欧州文字サポートがなくなります。

DB2 管理サーバーのロケール設定

DB2 管理サーバーのインスタンスのロケールが、DB2 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。そうでなければ、DB2 インスタンスは DB2 管理サーバーとは通信できません。

DB2 管理サーバーのユーザー・プロファイルで、LANG 環境変数が設定されていないと DB2 管理サーバー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 管理サーバーはコード・ページ 819

で始動します。 DB2 インスタンスが DBCS ロケールの 1 つを使用し、DB2 管理サーバーがコード・ページ 819 で始動された場合、そのインスタンスは DB2 管理サーバーと通信することができません。 DB2 管理サーバーのロケールと DB2 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) システムで LANG=zh_CN は、 DB2 管理サーバーのユーザー・プロファイルで設定されている必要があります。

UNIX ベースのプラットフォームでサポートされている翻訳ロケール

UNIX ベースのプラットフォームでは、DB2 製品メッセージとライブラリーは異なる言語のインストールが可能です。DB2 インストール・ユーティリティーは、メッセージ・カタログ・ファイル・セットを、次の表で示すプラットフォームの最もよく使用されるローカル・ディレクトリーに用意しています。 表29 は AIX、HP-UX、および Solaris の情報を提供します。 244ページの表30 は Linux、Linux/390、SGI、および Dynix の情報を提供します。

ご使用のシステムで、 表29 や 244ページの表30 で示されているのと同じコード・ページで異なったロケール名を使用している場合でも、適切なメッセージ・ディレクトリーにリンクをはれば、翻訳されたメッセージを見ることができます。

たとえば、お使いの AIX マシンでデフォルトのロケールが ja_JP.IBM-eucJP、ja_JP.IBM-eucJP のコード・ページが 954 になっている場合は、次のコマンドを使用して /usr/lpp/db2_07_01/msg/ja_JP.IBM-eucJP から /usr/lpp/db2_07_01/msg/ja_JP へのリンクを作成します。

ln -s /usr/lpp/db2_07_01/msg/ja_JP /usr/lpp/db2_07_01/msg/ja_JP.IBM-eucJP

このコマンドを実行後、DB2 の全メッセージは日本語で表示されるようになります。

表 29. AIX、HP-UX、Solaris のディレクトリー・ロケール

	AIX		HP-UX		Solaris	
言語	ロケール	コード・ページ	ロケール	コード・ページ	ロケール	コード・ページ
フランス語	fr_FR Fr_FR	819 850	fr_FR.iso88591 fr_FR.roman8	819 1051	fr	
ドイツ語	de_DE De_DE	819 850	de_DE.iso88591 de_DE.roman8	819 1051	de	819

表 29. AIX、HP-UX、Solaris のディレクトリー・ロケール (続き)

	AIX		HP-UX		Solaris	
言語	ロケール	コード・ページ	ロケール	コード・ページ	ロケール	コード・ページ
イタリア語	it_IT It_IT	819 850	it_IT.iso88591 it_IT.roman8	819 1051	it	819
スペイン語	es_ES Es_ES	819 850	es_ES.iso88591 es_ES.roman8	819 1051	es	819
ブラジル・ポルトガル語	pt_BR	819			pt_BR	819
日本語	ja_JP Ja_JP	954 932	ja_JP.eucJP	954	ja	954
韓国語	ko_KR	970	ko_KR.eucKR	970	ko	970
中国語 (簡体字)	zh_CN Zh_ CN.GBK	1383 1386	zh_CN.hp15CN	1383	zh	1383
中国語 (繁体字)	zh_TW Zh_TW	964 950	zh_TW.eucTW zh_TW.big5	964 950	zh_TW zh_TW.BIG5	964 950
デンマーク語	da_DK Da_DK	819 850	da_DK.iso88591 da_DK.roman8	819 1051	da	819
フィンランド語	fi_FI Fi_FI	819 850	fi_FI.iso88591 fi_FI.roman8	819 1051	fi	819
ノルウェー語	no_NO No_NO	819 850	no_NO.iso88591 no_NO.roman8	819 1051	no	819
スウェーデン語	sv_SE Sv_SE	819 850	sv_SE.iso88591 sv_SE.roman8	819 1051	sv	819
チェコ語	cs_CZ	912				
ハンガリー語	hu_HU	912				
ポーランド語	pl_PL	912				

表 29. *AIX*、*HP-UX*、*Solaris* のディレクトリー・ロケール *(続き)*

	AIX		HP-UX		Solaris	
言語	ロケール	コード・ページ	ロケール	コード・ページ	ロケール	コード・ページ
オランダ語	nl_NL Nl_NL	819 850				
トルコ語	tr_TR	920				
ロシア語	ru_RU	915				
ブルガリア語	bg_BG	915	bg_BG.iso88595	915		
スロベニア語	sl_SI	912	sl_SI.iso88592	912	sl_SI	912

表 30. *Linux*、*Linux/390*、*SGI*、*Dynix* のディレクトリー・ロケール

	Linux		Linux/390		SGI		Dynix	
言語	ロケール	コード・ページ	ロケール	コード・ページ	ロケール	コード・ページ	ロケール	コード・ページ
フランス語	fr	819	fr	819			fr	819
ドイツ語	de	819	de	819			de	819
イタリア語							es	819
スペイン語								
ブラジル・ポルトガル語								
日本語	ja_JP.ujis	954	ja_JP.ujis	954			ja_JP.EUC	954
韓国語	ko	970	ko	970	ko_KO.euc	970		
中国語 (簡体字)	zh zh_CN.GBK	1386	zh zh_CN.GBK	1386				
中国語 (繁体字)	zh_TW.Big5	950	zh_TW.Big5	950				
デンマーク語								
フィンランド語								

表 30. *Linux*、*Linux/390*、*SGI*、*Dynix* のディレクトリー・ロケール (続き)

	Linux		Linux/390		SGI		Dynix	
言語	ロケール	コード・ページ	ロケール	コード・ページ	ロケール	コード・ページ	ロケール	コード・ページ
ノルウェー語								
スウェーデン語								
チェコ語								
ハンガリー語								
ポーランド語								
オランダ語							nl	819
トルコ語								
ロシア語								
ブルガリア語								
スロベニア語								

コントロール・センターと資料のファイル・セット

コントロール・センター、コントロール・センター・ヘルプ、および資料のファイル・セットは、ターゲット・ワークステーション上の以下のディレクトリーに置かれています。

- DB2 (AIX 版):
  - /usr/lpp/db2_07_01/cc/%L
  - /usr/lpp/db2_07_01/java/%L
  - /usr/lpp/db2_07_01/doc/%L
  - /usr/lpp/db2_07_01/qp/%L
  - /usr/lpp/db2_07_01/spb/%L
- DB2 (HP-UX 版):
  - /opt/IBMdb2/V7.1/cc/%L
  - /opt/IBMdb2/V7.1/java/%L

- /opt/IBMdb2/V7.1/doc/%L

- DB2 (Linux 版):
  - /usr/IBMdb2/V7.1/cc/%L
  - /usr/IBMdb2/V7.1/java/%L
  - /usr/IBMdb2/V7.1/doc/%L
- DB2 (Solaris 版):
  - /opt/IBMdb2/V7.1/cc/%L
  - /usr/IBMdb2/V7.1/java/%L
  - /opt/IBMdb2/V7.1/doc/%L

コントロール・センターのファイル・セットは、ユニコード・コード・ページの中にあります。資料とコントロール・センター・ヘルプのファイル・セットはブラウザーが認識するコード・セットの中にあります。お使いのシステムが提供されているものとは異なるロケール名を使用している場合は、適切な言語ディレクトリーにリンクをはることで、翻訳されたバージョンのコントロール・センターを実行し、翻訳されたバージョンのヘルプをみることができます。

たとえば、お使いの AIX マシンでデフォルトのロケールが ja_JP.IBM-eucJP になっている場合は、次のコマンドを実行して、/usr/lpp/db2_07_01/cc/ja_JP.IBM-eucJP から /usr/lpp/db2_07_01/cc/ja_JP へのリンクと、/usr/lpp/db2_07_01/doc/ja_JP.IBM-eucJP から /usr/lpp/db2_07_01/doc/ja_JP へのリンクを作成できます。

- **ln -s /usr/lpp/db2_07_01/cc/ja_JP /usr/lpp/db2_07_01/cc/ja_JP.IBM-eucJP**
- **ln -s /usr/lpp/db2_07_01/doc/ja_JP /usr/lpp/db2_07_01/doc/ja_JP.IBM-eucJP**

これらのコマンドの実行後、コントロール・センターとヘルプは日本語で表示されるようになります。

注: Web コントロール・センターは、Linux/390 または NUMA-Q でのネイティブ実行をサポートしていません。これらのプラットフォームでのデータベースを管理するため、クライアント・ワークステーションから実行できます。

コード・ページ値の取得

アプリケーション・コード・ページ は、データベースの接続が行われるときに、活動中の環境から入手されます。 DB2CODEPAGE レジストリー変数が設定されている場合は、この値は、アプリケーション・コード・ページとしてとられます。通常、DB2 がオペレーティング・システムからコード・ページ情報を自動的に取得するため、DB2CODEPAGE レジストリー変数を設定する必要はありません。 DB2CODEPAGE レジストリー変数に誤った値を設定すると、予期できない結果になります。

CLI インターフェースを使用するようにコーディングされたアプリケーションでは、DB2CODEPAGE レジストリー変数が設定されていても、CLI コード層はロケール設定を使用する場合があります。

データベース・コード・ページ は、データベースが作成される時点で指定された (明示またはデフォルトにより) 値から得られます。たとえば、各種の操作環境で*活動状態の環境* がどのように判別されるかを、以下で定義します。

UNIX UNIX ベースのオペレーティング・システムでは、活動状態の環境は、ロケール設定から判別されます。この変数には、言語、テリトリーおよびコード・セットに関する情報が入っています。

OS/2 OS/2 では、1 次および 2 次コード・ページは、CONFIG.SYS ファイルで指定されます。 **chcp** コマンドを使用して、指定のセッションの中でコード・ページを表示して、動的に変更することができます。

Macintosh Macintosh のオペレーティング・システムでは、DB2CODEPAGE レジストリー変数が設定されていない場合、 Macintosh コード・ページは、導入されたスクリプトの Regional バージョン・コードから得られます。

Windows すべての Windows 32 ビット・オペレーティング・システムでは、 DB2CODEPAGE レジストリー変数が設定されていないと、コード・ページはレジストリーにある ANSI コード・ページ設定から得られます。

コード・ページ値の環境マッピングの完全なリストについては、 226ページの表28 を参照してください。

文字セット

一般に、データベース・マネージャーはアプリケーションで使用できる文字セットを制約しません。 DB2 でサポートされているマルチバイト文字セット (MBCS) について詳しくは、 *アプリケーション開発の手引き* を参照してください。

ID 用の文字セット

データベース名の中で使用できる基本文字セットは、単一バイトの大文字および小文字のローマ字 (A...Z、a...z)、アラビア数字 (0...9) および下線文字 (_) から構成されます。ホストのデータベース製品との互換性を保つために、 3 つの特殊文字 (#、@、および $) がこのリストに加えられています。特殊文字 #、@、および $ は NLS ホスト

(EBCDIC) 不変文字セットに組み込まれていないため、これらを NLS 環境で使用する場合は注意してください。使用されているコード・ページに応じて、拡張文字セットの文字も使用できます。複数コード・ページ環境でデータベースを使用している場合、使用する拡張文字セットのどのエレメントもすべてのコード・ページによってサポートされていることを確認する必要があります。

データベース・オブジェクト (表や視点など) に命名するときは、プログラム・ラベル、ホスト変数、カーソル、および拡張文字セットからの要素 (たとえば、区別的発音符付きの文字) も使用することができます。厳密にどの文字を使用できるかは、使用しているコード・ページによって異なります。複数のコード・ページ環境でデータベースを使用する場合には、すべてのコード・ページが、使用を計画している拡張文字セットからの要素をすべてサポートするようにする必要があります。拡張文字セット以外の文字を含むにもかかわらず SQL ステートメントで使用できる区切り ID については、*SQL 解説書* を参照してください。

DBCS ID 用の拡張文字セットの定義

DBCS 環境では、拡張文字セットは、基本文字セットにあるすべての文字、および次のような文字から構成されます。

- おのおのの DBCS コード・ページにある 2 バイト文字は、すべて (2 バイトのスペースは除く) 有効な文字です。
- 2 バイトのスペースは、特殊文字です。
- 混合コード・ページのおのおので使用できる単一バイト文字は、次のように様々なカテゴリーに割り当てられます。

カテゴリー	各混合コード・ページにある有効なコード・ポイント
数字	x30-39
文字	x23-24、x40-5A、x61-7A、 xA6-DF (コード・ページ 932 および 942 の場合のみ A6-DF)
特殊文字	その他のすべての有効な単一バイト文字のコード・ポイント

SQL ステートメントのコーディング

SQL ステートメントのコーディングは、言語によって異なることはありません。 SQL キーワードは、大文字、小文字、または大文字小文字混合のいずれでも入力できます。データベース・オブジェクトやホスト変数の名前、および SQL ステートメント内のプログラム・ラベルの名前には、 『DBCS ID 用の拡張文字セットの定義』で説明されている拡張文字セット以外の文字を含めることはできません。

双方向 CCSID サポート

以下の BiDi 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。

```
- Text type (LOGICAL or VISUAL)
- Shaping (SHAPED or UNSHAPED)
- Orientation (RIGHT-TO-LEFT or LEFT-TO-RIGHT)
- Numeral shape (ARABIC or HINDI)
- Symmetric swapping (YES or NO)
```

プラットフォームごとにデフォルト値が異なるため、 DB2 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、 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 ユニバーサル・データベース (OS/390 版) 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。

双方向特定の CCSID

DB2 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。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。

CCSID (dec)	CCSID (hex)	コードページ	ストリングタイプ
00420	x'01A4'	420	4
00424	x'01A8'	424	4
08612	x'21A4'	420	5
08616	x'21A8'	424	10
00856	x'0358'	856	5
00862	x'035E'	862	4
00864	x'0360'	864	5
00916	x'0394'	916	5
01046	x'0416'	1046	5
01089	x'0441'	1089	5
01255	x'04E7'	1255	5
01256	x'04E8'	1256	5
62208	x'F300'	856	4
62209	x'F301'	862	10
62210	x'F302'	916	4
62211	x'F303'	424	5
62213	x'F305'	862	5
62215	x'F307'	1255	4
62218	x'F30A'	864	4
62220	x'F30C'	856	6
62221	x'F30D'	862	6
62222	x'F30E'	916	6
62223	x'F30F'	1255	6
62224	x'F310'	420	6
62225	x'F311'	864	6
62226	x'F312'	1046	6
62227	x'F313'	1089	6
62228	x'F314'	1256	6

```
62229   x'F315'   424      8
62230   x'F316'   856      8
62231   x'F317'   862      8
62232   x'F318'   916      8
62233   x'F319'   420      8
62234   x'F31A'   420      9
62235   x'F31B'   424      6
62236   x'F31C'   856     10
62237   x'F31D'  1255      8
62238   x'F31E'   916     10
62239   x'F31F'  1255     10
62240   x'F320'   424     11
62241   x'F321'   856     11
62242   x'F322'   862     11
62243   x'F323'   916     11
62244   x'F324'  1255     11

62245   x'F325'   424     10
62246   x'F326'  1046      8
62247   x'F327'  1046      9
62248   x'F328'  1046      4
62249   x'F329'  1046     12
62250   x'F32A'   420     12
```

CDRA ストリング・タイプは以下のように定義されます。

ストリング・タイプ	テキスト・タイプ	数値型	方向	型	対称交換
4	Visual	Passthru	LTR	Shaped	OFF
5	Implicit	Arabic	LTR	Unshaped	ON
6	Implicit	Arabic	RTL	Unshaped	ON
7(*)	Visual	Arabic	Contextual(*)	Unshaped-Lig	OFF
8	Visual	Arabic	RTL	Shaped	OFF
9	Visual	Passthru	RTL	Shaped	ON
10	Implicit	Passthru	Contextual-L	Unshaped	ON
11	Implicit	Passthru	Contextual-R	Unshaped	ON
12	Implicit	Arabic	RTL	Shaped	ON

注: (*) 最初のアルファベット文字がローマ字なら、フィールドの方向は左から右 (left-to-right (LTR)) になります。双方向 (RTL) 文字の場合は、右から左 (right-to-left (RTL)) になります。文字が Unshaped になっていても、LamAlef リガチャーは保たれており、構成の中では壊れていません。

DB2 ユニバーサル・データベースにおける双方向サポートの実装

双方向レイアウト変換は、新しい CCSID 定義を使って DB2 ユニバーサル・データベースで実装されます。新しい BiDi 特定の 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、この値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。

非 DRDA 環境で特定の双方向 CCSID を指定するには、クライアントの特性と一致する CCSID を (上記の表から) 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なおす必要があります。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。

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。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。 CCSID オーバーライドの設定について詳しくは、*DB2 コネクト 使用者の手引き* を参照してください。

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。

双方向の DB2 コネクト実装のサポート

DB2 コネクト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 コネクトは、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 コネクト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 ホスト・データベースに送る予定のデータに対して、 DB2 コネクト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 ホスト・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。

サポートされている双方向 CCSID とそのストリング・タイプのリストは、249ページの『双方向特定の CCSID』 にあります。

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 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 ホスト・データベースにアクセスしたいとします。しかし、DB2 ホスト・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 ホスト・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 ホスト・データベースは、 DB2 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 ホスト・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 ホスト・データベースから受け取るデータに対して両方項変換を実行しなければならないことを、 DB2 コネクトに指示する必要があります。 DB2 ホスト・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 ホスト・データベースの CCSID 00424 を 08616 でオーバーライドするよう DB2 コネクトに指示します。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 コネクトは CCSID 00862 を使用して DB2 ホスト・データベースへ接続します。
2. DB2 コネクトは、DB2 ホスト・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 コネクトは、DB2 ホスト・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 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。

照合順序

データベース・マネージャーは、*照合順序* を使用して、文字データを比較します。これは、ある文字がほかの文字とくらべて、高いか、低いか、または同じかを判別するための文字セットの順序付けです。たとえば、照合順序を使用して、ある文字についての大文字・小文字のバージョンは、同じものとしてソートされます。

照合順序はデータベース作成の時点で指定します。あとでこれを変更することはできません。

データベース・マネージャーでは、アプリケーション・プログラミング・インターフェース (API) を使用して、カスタムの照合順序でデータベースを作成することができます。カスタムの照合順序表の実装については、*アプリケーション開発の手引き* を参照してください。

注: FOR BIT DATA 属性で定義された文字ストリング・データ、および BLOB データは、2 進数のソート順序を用いてソートされます。

一般的な問題

いったん照合順序が定義されると、そのデータベースに対するその後の文字の比較はすべて、その照合順序で行われます。 FOR BIT DATA または BLOB データとして定義された文字データの場合を除いて、すべての SQL 比較および ORDER BY 文節についても、また索引や統計の設定においても、この照合順序が使用されます。データベースの照合順序の使用方法についての詳細は、 *SQL 解説書* の『ストリングの比較』を参照してください。

以下のような場合には、問題が起こる可能性があります。

- アプリケーションが、データベースのソート済みデータを、異なる照合順序を用いてソートされたアプリケーション・データとマージしている。

- アプリケーションが、あるデータベースのソート済みデータを、別のデータベースのソート済みデータとマージしているが、これらのデータベースの照合順序が異なっている。
- アプリケーションがソート済みデータについてある想定をしているが、その想定が、使用された照合順序とはちがっている。たとえば、特定の照合順序では、英字より数字の方が照合順序が低い場合もあります。そうではない場合もあります。

最後に、文字コード・ポイントの直接比較にもとづくソートの結果が一致するのは、一致照合順序を用いて順序づけられた照合の結果だけであることに注意してください。

連合データベースでの問題

データベース照合順序の選択は、連合システムのパフォーマンスに影響することがあります。あるデータ・ソースが、 DB2 連合データベースと同じ照合順序を使う場合、DB2 は、文字データが関係する順序依存の処理を、そのデータ・ソースにプッシュダウンすることができます。データ・ソース照合順序が DB2 の照合順序と一致しない場合、データが取り出され、文字データについてのすべての順序依存の処理がローカルに行われます (パフォーマンスが低下することがあります)。

データ・ソースと DB2 の照合順序が同じかどうかを判別するには、以下の要因を考慮してください。

- 各国語サポート

  照合順序は、サーバーでサポートされている言語に関連しています。 DB2 の NLS 情報を、データ・ソースの NLS 情報と比較してください。
- データ・ソースの特性

  データ・ソースによっては、大文字小文字を区別しない照合順序で作成されるものもありますが、その場合、順序依存の操作をすると、DB2 から異なる結果を受け取る可能性があります。
- カスタマイズ。

  データ・ソースの中には、照合順序のためのオプションをいくつか備えたもの、あるいは照合順序をカスタマイズできるものがあります。

DB2 連合データベースからアクセスするさまざまなデータ・ソースを考慮した上で、そのデータベースに適した照合順序を選択してください。たとえば、次のようにします。

- ほとんどの場合に、 DB2 と同じコード・ページ (NLS) の Oracle データベースに DB2 データベースがアクセスするのであれば、データベース作成時に一致順序を指定します (Oracle データベースは同等の照合順序を使います)。
- DB2 データベースが DB2 UDB データベースだけにアクセスするのであれば、それぞれの値が照合順序の値と一致するようにします。

MVS の照合順序のセットアップ方法については、 *管理 API 解説書* の **sqlecrea** (データベース作成) API の説明に記載されている例を参照してください。この例には、EBCIDIC 500、37、および 5026/5035 コード・ページの照合表が載せられています。

DB2 データベースに照合順序を設定したら、データ・ソースのサーバーごとに、*collating_sequence* サーバー・オプションを設定してください。このオプションは、所定のデータ・ソース・サーバーの照合順序が DB2 データベースの照合順序と一致するかどうかを指定します。

照合順序が一致する場合は、collating_sequence オプションを 「Y」にセットしてください。このように設定すると、DB2 最適化プログラムは、データ・ソースでの順序依存の処理を選ぶことができます。これにより、パフォーマンスが改善されます。しかし、データ・ソースの照合順序が DB2 データベースの照合順序と同じでないと、正しくない結果を受け取ることがあります。たとえば、マージ結合を使う計画の場合、DB2 最適化プログラムは、順序付けの操作をできる限りデータ・ソースへプッシュダウンします。データ・ソースの照合順序が同じでなければ、結合の結果セットが正しくない可能性があります。

照合順序が一致しない場合は、collating_sequence オプションを「N」にセットしてください。この値は、データ・ソースの照合順序が DB2 とは異なるとき、あるいはデータ・ソースの照合順序が大文字小文字を区別しないときに使用します。たとえば、英語のコード・ページでの大文字小文字を区別しないデータ・ソースでは、 TOLLESON、ToLLeSoN、および tolleson はすべて同じものであると見なされます。データ・ソースでの照合順序が DB2 の照合順序と同じかどうか分からないときには、この collating_sequence オプションを 「N」 にセットしてください。

タイ文字のソート

特殊母音 (″長母音″) や声調記号その他の特殊なタイ文字を正しくソートするためには、データベースを作成する際に CREATE DATABASE コマンドに COLLATE USING NLSCHAR 文節を加える必要があります。構文については、*コマンド解説書* を参照してください。

日時値

日時値のデータ・タイプを次に説明します。日時値は、特定の算術計算やストリング演算に使用することができ、特定のストリングとも互換性がありますが、この値はストリングでも数字でもありません。

日付

日付 は、3 つの部分の値 (年、月、および日) です。年の部分の範囲は 0001 から 9999 です。月の部分の範囲は、1 から 12 です。日の部分の範囲は、1 から *x* で、*x* は、月によって決まります。

日付の内部表記は 4 バイトのストリングです。それぞれのバイトは、2 つのパック 10 進数から構成されます。最初の 2 バイトは年を表し、3 番目のバイトは月、最後のバイトは日を表します。

DATE 列の長さは、SQLDA に記述されているように 10 バイトですが、これは、この値の文字ストリング表示として適切な長さです。

時刻

時刻 は、3 つの部分の値 (時、分、および秒) で、 24 時間時計の時刻を示します。時間の部分の範囲は 0 から 24 で、他の部分の範囲は、0 から 59 です。時間が 24 の場合は、分と秒の指定はゼロになります。

時刻の内部表記は 3 バイトのストリングです。それぞれのバイトは、2 つのパック 10 進数です。最初のバイトは時間を表し、2 番目のバイトは分を、最後のバイトは秒を表します。

TIME 列の長さは、SQLDA に記述されているように 8 バイトですが、これは、この値の文字ストリングとして適切な長さです。

タイム・スタンプ

タイム・スタンプ とは、 7 つの部分からなる値 (年、月、日、時、分、秒、およびマイクロ秒) で上と同じように日時を指定するものですが、上記との違いは、時刻にマイクロ秒の指定が含まれる点です。

タイム・スタンプの内部表記は 10 バイトのストリングです。それぞれのバイトは、2 つのパック 10 進数です。最初の 4 バイトは日付を表し、次の 3 バイトは時刻を表し、最後の 3 バイトはマイクロ秒を表します。

TIMESTAMP 列の長さは、SQLDA に記述されているように 26 バイトですが、これは、この値の文字ストリング表示として適切な長さです。

日時値のストリング表示

データ・タイプが DATE、TIME、または TIMESTAMP である値は、 SQL ユーザーにとって透過であるような内部形式で表示されます。しかし、日付、時刻、およびタイム・スタンプは、文字ストリングでも表示でき、そのような表示は SQL ユーザーに直接かかわってきます。データ・タイプが DATE、TIME、または TIMESTAMP であるような定数や変数はないからです。したがって、検索をするときは、日時値を文字ストリング変数に割り当てる必要があります。文字ストリング表示は、通常、クライアントの国 / 地域別コードと関連づけられた日時値のデフォルト形式になりますが、プログラムが事前にコンパイルされたか、データベースにバインドされたときに、 ″F″ 形式オプションの指定によって指定変更することもできます。各種の国 / 地域別コードのストリング形式のリストは、259ページの表33 を参照してください。

日時値の有効なストリング表示が内部的な日時値を用いる演算に使用されるときは、ストリング表示は、演算が行われる前に日付、時刻、またはタイム・スタンプの内部形式に変換されます。次のセクションで、日時値の有効なストリング表示を定義します。

日付ストリング

日付のストリング表示は、数字で始まり、最低 8 文字の長さをもつ文字ストリングです。後書きブランクを含めることもできます。また、日付の月および日の部分の先行ゼロを省略することができます。

日付の有効なストリング形式が表31 にリストされています。それぞれの形式は名前で識別され、省略形と使用例も含めてあります。

表 31. 日付のストリング表示の形式

形式名	省略形	日付形式	例
国際標準化機構	ISO	yyyy-mm-dd	1991-10-27
IBM USA 標準規格	USA	mm/dd/yyyy	10/27/1991
IBM 欧州標準規格	EUR	dd.mm.yyyy	27.10.1991
日本工業規格 (JIS)	JIS	yyyy-mm-dd	1991-10-27
地域別定義 (ローカル)	LOC	データベース国 / 地域別コードによる	—

時刻ストリング

時刻のストリング表示は、数字で始まり、最低 4 文字の長さをもつ文字ストリングです。後書きブランクを含めることができます。また、時刻の時間の部分の先行ゼロを省略したり、秒全体を省略することができます。秒を省略するよう選択すると、0 秒が指定されたものと暗黙に想定されます。したがって、`13.30` は `13.30.00` に相当します。

時刻の有効なストリング形式が表32 にリストされています。それぞれの形式は名前で識別され、省略形と使用例も含めてあります。

表 32. 時刻のストリング表示の形式

形式名	省略形	時刻形式	例
国際標準化機構	ISO	hh.mm.ss	13.30.05
IBM USA 標準規格	USA	hh:mm AM または PM	1:30 PM
IBM 欧州標準規格	EUR	hh.mm.ss	13.30.05
日本工業規格 (JIS)	JIS	hh:mm:ss	13:30:05
地域別定義 (ローカル)	LOC	アプリケーションの国 / 地域別コードによる	—

注:

1. ISO、EUR または JIS 時刻ストリング形式では、 .ss (または :ss) はオプションです。
2. USA 時刻ストリング形式の場合は、分の指定を省略することができます (00 分を暗黙に指定したことになります)。したがって、1 PM は 1:00 PM に相当します。
3. USA 時刻形式では、時間指定は 12 より大きくすることはできません。また、00:00 AM という特殊な場合を除いて、0 にすることはできません。 24 時間時計の ISO 形式を使用する場合は、USA 形式と 24 時間時計の対応は、次のようになります。
   - 12:01 AM から 12:59 AM は、00.01.00 から 00.59.00 に相当します。
   - 01:00 AM から 11:59 AM は、01.00.00 から 11.59.00 に相当します。
   - 12:00 PM (正午) から 11:59 PM は、12.00.00 から 23.59.00 に相当します。
   - 12:00 AM (深夜) は、24.00.00 に相当し、00:00 AM (深夜) は 00.00.00 に相当します。

タイム・スタンプ・ストリング

タイム・スタンプのストリング表示は、数字で始まり、最低 16 文字の長さをもつ文字ストリングです。タイム・スタンプの完全なストリング表示は、*yyyy-mm-dd-hh.mm.ss.nnnnnn* という形式です。後書きブランクを含めることもできます。また、タイム・スタンプの月、日、および時刻の部分の先行ゼロを省略することができます。さらに、マイクロ秒を切り捨てるか、または全体を省略することができます。マイクロ秒の部分を省略するよう選択すると、0 が指定されたものと暗黙に想定されます。したがって、1991-3-2-8.30.00 は 1991-03-02-08.30.00.000000 に相当します。

文字セットに関する考慮事項

日付およびタイム・スタンプのストリングの内容は、単一バイトの文字および数字でなければなりません。

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国 / 地域別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、 *F* 形式オプションの指定によって指定変更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、すべての国 / 地域別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33 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33 に示されています。

注: 表33 には、各種の国 / 地域別コードのストリング形式のリストが示されています。

表 33. 国 / 地域別コードによる日時の形式

国 / 地域別コード	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
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

表 33. 国 / 地域別コードによる日時の形式 (続き)

国 / 地域別コード	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
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
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

表 33. 国 / 地域別コードによる日時の形式 (続き)

国 / 地域別コード	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
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 33. 国 / 地域別コードによる日時の形式 (続き)

国 / 地域別コード	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、国 / 地域別コード 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)。				

DB2 UDB でのユニコード・サポート

ここでは、DB2 UDB におけるユニコード・サポートのレベルを説明します。

ユニコードの紹介

ユニコード文字コード化規格は、固定長の文字コード化方式です。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。

ユニコードでは、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、 65000 のコード要素があれば十分ですが、ユニコード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 1 つの外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。

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、サロゲートのないユニコード 16 ビット形式と同一です。 ISO/IEC 10646 ではまた、一部の UCS-4 文字を 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、サロゲートのあるユニコード 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 ビットのユニコード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。ユニコードのデフォルトのバイト配列はビッグ・エンディアンです。

ユニコードに関する詳細は、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。

UTF-8

16 ビットのユニコード文字には、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、ユニコード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、固定長のユニコード文字を可変長の 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 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34から判断できます。

表 34. UTF-8 ビットの配布

コード値 (2 進数)	UTF-16 (2 進数)	最初のバイト (2 進数)	2 番目のバイト (2 進数)	3 番目のバイト (2 進数)	4 番目のバイト (2 進数)
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 2 進数で 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 2 進数で 11110000 10010000 10000000 10000000 になります。

DB2 UDB でのユニコードのインプリメンテーション

DB2 UDB は、UCS-2、すなわちサロゲートのないユニコードをサポートしています。

バージョン 7.2 FixPak 4 以前の DB2 UDB では、サロゲート・ペアの 2 つの文字が 2 つの独立したユニコード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 (263ページの表34の下から 2 行目を参照してください)。 DB2 UDB バージョン 7.2 FixPak 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 (263ページの表34の最後の行を参照してください)。

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、それぞれのユニコード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。

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ユニコードを表します。

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、漢字ストリング・データを euc-Japan および euc-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	1 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、データベース内の漢字ストリング・データのコード・ページです。ユニコード・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 で格納され、GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、および DBCLOB データは UCS-2 で格納されます。

ユニコード・データベースの作成

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、ユニコード (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)、データベースはユニコード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 UDB によってサポートされている任意の地域コードを使用できます。

たとえば、米国の地域コードを指定してユニコード・データベースを作成する場合は、次のコマンドを実行します。

```
DB2 CREATE DATABASE dbname USING CODESET UTF-8 TERRITORY US
```

sqlecrea API を使用してユニコード・データベースを作成するには、それぞれの値を *sqledbcountryinfo* 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。

UCS-2 ユニコード・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、すべてのユニコード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。ほとんどのユニコード文字の場合、 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外字が含まれている場合は、 UTF-8 での文字のエンコードは後ろ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行われます。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 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、同じ外字が 2 つの UCS-2 サロゲート文字としてエンコードされ、 DB2 UDB はこの各サロゲート文字を別個に扱い、照合します。

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。

ユニコード・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。クライアントのコード・ページと UTF-8 との間の

コード・ページ文字変換は、データベース・マネージャーによって自動的に行われます。漢字ストリング・タイプのデータは常に UCS-2 内に存在し、コード・ページ変換は行われません。ただしコマンド行プロセッサー (CLP) 環境は例外です。 CLP から漢字ストリング (UCS-2) データを意図的に選択すると、戻される漢字ストリング・データは、 CLP によって UCS-2 からそれぞれのクライアント環境のコード・ページに変換されます。

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 UTF-8 に変換すると、バイト数が増大する可能性があります。 ASCII 文字の場合はこのような増大はありませんが、他の UCS-2 文字は、いくつかの要因のために増大します。 UTF-8 形式におけるそれぞれの UCS-2 文字のバイト数は、 263ページの『UTF-8』の表を見て判断することができます。

データ・タイプ

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、漢字ストリング・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 の漢字ストリング・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、漢字ストリング 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 漢字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 スカラー関数 (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。このことは、UCS-2 データベースを UTF-8 で定義する場合に特に当てはまります。単一バイト文字の中には 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 指標は、 UCS-2 データベースには有効ではない場合があります。正しくない指標を指定すると、SQLCODE -191 (SQLSTATE 22504) が戻されます。これらの関数の動作についての詳細は、*SQL 解説書* を参照してください。

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 との間で変換されますが、漢字ストリング・データは常に UCS-2 になります。

ID

UCS-2 データベースでは、ID はすべてマルチバイトの UTF-8 です。ですから、DB2 UDB によって拡張文字セット (たとえば、アクセント記号、マルチバイト文字) での文字の使用が許可されている箇所では、 ID として任意の UCS-2 文字を使用することができます。拡張文字を使用できる ID について詳しくは、 187ページの『付録A. 命名規則』を参照してください。

クライアントは、それぞれの環境でサポートされている任意の文字を入力することができます。その後、ID に入れられたすべての文字がデータベース・マネージャーによって UTF-8 へ変換されます。 UCS-2 データベースで ID に各国語の文字を指定するときには、以下の 2 つの点を考慮する必要があります。

- 非 ASCII 文字にはそれぞれ 2 から 4 バイトが必要。そのため、*n* バイトの ID は、ASCII 文字と非 ASCII 文字の比率に応じて、 *n*/4 文字から *n* 文字の範囲を保持できます。非 ASCII 文字が 1 文字か 2 文字しか含まれない場合 (たとえば、アクセント記号の場合)、その上限は *n* 文字に近くなりますが、完全に非 ASCII である ID の場合 (たとえば、日本語の場合)、使用できるのは *n*/4 から *n*/3 文字だけです。
- ID を別のクライアント環境から入力するのであれば、そのクライアントで使用できる文字の共通サブセットを使用して、 ID を定義しなければならない。たとえば、UCS-2 データベースが Latin-1、アラビア語、および日本語環境からアクセスされる場合、すべての ID を実際に ASCII に限定する必要があります。

ユニコード・リテラル

ユニコード・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'....' 形式を使用した漢字ストリング定数として指定する (*SQL 解説書* の『言語要素』の章の『漢字ストリング定数』を参照)。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・ユニコードに変換されます。
- UX'....' または GX'....' 形式を使用したユニコード 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・ユニコード・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 つ必要であることを覚えておいてください。

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)。

UCS-2 データベースにおけるパターン照合

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 UDB の MBCS データベースの現在の動作を考える場合、突き合わせ式に MBCS データが含まれていれば、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。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 下線は 1 つの SBCS 文字を表す。
- DBCS 下線は 1 つの MBCS 文字を表す。
- パーセント記号 (SBCS あるいは DBCS のいずれか) は、ゼロ以上の SBCS 文字または非 SBCS 文字のストリングを表す。

ユニコード・データベースでは、「単一バイト」文字と「2 バイト」文字とは区別されません。 16 ビット文字はそれぞれ 2 バイトを使います。 UTF-8 形式では、ユニコード文字が「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 MBCS 文字との間の区別はありません。それぞれの文字は、UTF-8 形式でのバイト数に関係なくユニコード文字です。ある文字ストリングあるいは漢字ストリングの式を指定する場合、下線は 1 つのユニコード文字を表し、パーセント記号はゼロ以上のユニコード文字のストリングを表します。サロゲートはペアになっているので、下線は 2 つ必要であることに注意してください。

クライアントでは、この文字ストリング式は、クライアントのコード・ページに含まれており、データベース・マネージャーによって UTF-8 へ変換されます。 SBCS クライアントのコード・ページには DBCS パーセント記号や DBCS 下線は含まれていませんが、サポートされている各コード・ページには、単一バイトのパーセント記号 (U+0025

に相当) および単一バイトの下線 (U+005F に相当) が含まれています。 UCS-2 データベースでの特殊文字の解釈は以下のとおりです。

- SBCS 下線 (U+0025 に相当) は、漢字ストリング式においては 1 つの UCS-2 文字を表し、文字ストリング式においては 1 つの UTF-8 文字を表す。
- SBCS パーセント記号 (U+005F に相当) は、漢字ストリング式においてはゼロ以上の UCS-2 文字のストリングを表し、文字ストリング式においてはゼロ以上の UTF-8 文字のストリングを表す。

さらに DBCS コード・ページは、 1 つの DBCS パーセント記号 (U+FF05 に相当) と 1 つの DBCS 下線 (U+FF3F に相当) をサポートしています。これらの文字は、UCS-2 データベースでは特別な意味がありません。

任意指定の ″エスケープ式″ を使い、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、サポートされるのは、ASCII 文字、あるいは 2 バイトの UTF-8 シーケンスに展開される文字だけです。エスケープ文字を指定して、3 バイトの UTF-8 値に展開すると、エラー・メッセージ (エラー SQL0130N、SQLSTATE 22019) が戻されます。

インポート / エクスポート / ロードについての考慮事項

このセクションで説明されているように、UCS-2 データベースでは、DEL、ASC、および PC/IXF ファイル形式だけがサポートされています。 WSF 形式はサポートされていません。

UCS-2 データベースから ASCII 区切り (DEL) ファイルへエクスポートすると、文字データはすべてアプリケーション・コード・ページに変換されます。文字ストリング・データと漢字ストリング・データはどちらも、クライアントの同じ SBCS または MBCS コード・ページに変換されます。これは、いずれのデータベースをエクスポートするときに予想される動作であり、区切り付き ASCII ファイル全体には 1 つのコード・ページしか含められないので、この動作を変更することはできません。したがって、区切り付き ASCII ファイルへエクスポートするときには、アプリケーション・コード・ページに存在する UCS-2 文字だけが保管されます。他の文字については、アプリケーション・コード・ページのデフォルトの置換文字に置き換えられます。 UTF-8 クライアント (コード・ページ 1208) の場合、 UTF-8 クライアント側ですべての UCS-2 文字がサポートされているので、データが失われることはありません。

ASCII ファイル (DEL または ASC) から UCS-2 データベースへインポートする場合、文字ストリング・データはアプリケーション・コード・ページから UTF-8 に変換され、漢字ストリング・データはアプリケーション・コード・ページから UCS-2 に変換されます。失われるデータはありません。別のコード・ページに保管された ASCII データをインポートする場合は、インポートのコマンドを発行する前に、データ・ファイルのコード・ページを変更する必要があります。そのための方法の 1 つは、ASCII データ・ファイルのコード・ページに、 DB2CODEPAGE をセットすることです。

SBCS および MBCS クライアントで有効な ASCII 区切り文字の範囲は、 DB2 UDB によってこれらのクライアント向けに現在サポートされている、有効な ASCII 区切り文字の範囲と同じになります。 UTF-8 クライアントでの有効な区切り文字の範囲は X'01' から X'7F' で、通常の制約があります。これらの制約の完全なリストは、 *データ移動ユーティリティー手引きおよび解説書* の付録『エクスポート / インポート / ロード・ユーティリティーのファイル形式』を参照してください。

UCS-2 データベースから PC/IXF ファイルにエクスポートする場合、文字ストリング・データはクライアントの SBCS/MBCS コード・ページに変換されます。漢字ストリング・データは変換されず、UCS-2 (コード・ページ 1200) で格納されます。失われるデータはありません。

PC/IXF ファイルから UCS-2 データベースへインポートする場合、文字ストリング・データは、 PC/IXF ヘッダーに示されている SBCS/MBCS コード・ページに含まれるものと見なされ、漢字ストリング・データは、 PC/IXF ヘッダーに示されている DBCS コード・ページに含まれるものと見なされます。文字ストリング・データは、インポート・ユーティリティーにより、 PC/IXF ヘッダーで指定されたコード・ページからクライアントのコード・ページに変換され、その後、(INSERT ステートメントにより) クライアントのコード・ページから UTF-8 へ変換されます。漢字ストリング・データは、インポート・ユーティリティーにより、 PC/IXF ヘッダーで指定された DBCS コード・ページから UCS-2 (コード・ページ 1200) へ直接に変換されます。

ロード・ユーティリティーはデータを直接にデータベースへ入れ、デフォルトで、 ASC または DEL ファイルのデータはデータベースのコード・ページであると見なします。したがって、ASCII ファイルについては、デフォルトではコード・ページ変換は行われません。 (`codepage` 修飾子を使用して) データ・ファイルのコード・ページを明示的に指定しておけば、ロード・ユーティリティーはその情報を利用してコード・ページをデータベースのコード・ページに変換し、それからデータをロードします。 PC/IXF の場合、ロード・ユーティリティーは常に IXF ヘッダーで指定したコード・ページからデータベースのコード・ページ (CHAR の場合は 1208、 GRAPHIC の場合は 1200) へ変換します。

DBCLOB ファイルのコード・ページは、UCS-2 では必ず 1200 です。 CLOB ファイルのコード・ページは、インポート、ロード、エクスポートされるデータ・ファイルのコード・ページと同じです。たとえば、PC/IXF 形式を使用してデータをロードまたはインポートするときには、 CLOB ファイルは、PC/IXF ヘッダーで指定したコード・ページであると見なされます。 DBCLOB ファイルが ASC または DEL 形式である場合、ロード・ユーティリティーは、(`codepage` 修飾子を使って別のコード・ページを明示的に指定しない限り) CLOB データがデータベースのコード・ページであると見なし、インポート・ユーティリティーは、クライアントのアプリケーション・コード・ページであると見なします。

UCS-2 データベースでは、以下の理由で `nochecklengths` 修飾子を常に指定します。

- DBCS コード・ページが存在しないデータベースに対して、いずれの SBCS も接続できる。
- UTF-8 形式の文字ストリングは、通常、クライアント・コード・ページの文字ストリングの長さとは異なる。

ロード、インポート、およびエクスポート・ユーティリティーについての詳細は、 *データ移動ユーティリティー手引きおよび解説書* を参照してください。

非互換性

UCS-2 データベースに接続されるアプリケーションの場合、漢字ストリング・データは常に UCS-2 (コード・ページ 1200) です。 UCS-2 以外のデータベースに接続されるアプリケーションの場合、漢字ストリング・データはアプリケーションの DBCS コード・ページにあります。ただし、アプリケーション・コード・ページが SBCS であれば、これは許可されません。たとえば、932 クライアントが日本語の非 UCS-2 データベースに接続されている場合、漢字ストリング・データはコード・ページ 301 になります。 UCS-2 データベースに接続された 932 クライアント・アプリケーションの場合、漢字ストリング・データは UCS-2 に存在します。

付録E. DB2 ライブラリーの使用法

DB2 ユニバーサル・データベース ライブラリーは、オンライン・ヘルプ、ブック (PDF および HTML)、および HTML 形式のサンプル・プログラムから成っています。このセクションでは、ユーザーに提供される情報について紹介し、その入手方法を示します。

オンライン製品情報をご利用になるには、インフォメーション・センターを使用することができます。詳細については、289ページの『インフォメーション・センターを使用した情報へのアクセス』を参照してください。ここではタスク情報、DB2 ブック、トラブルシューティング情報、サンプル・プログラム、および Web の DB2 情報を見ることができます。

DB2 PDF ファイルおよびハードコピー版資料

DB2 情報

以下に示す表では、DB2 ブックを 4 つのカテゴリーに分類しています。

DB2 の手引きおよび解説書
これらの資料は、すべてのプラットフォームに共通の DB2 情報を含んでいます。

DB2 のインストールおよび構成の情報
これらの資料は、特定のプラットフォーム上の DB2 ごとに用意されています。たとえば、 OS/2、Windows、および UNIX ベースのプラットフォームで稼働するそれぞれの DB2 用に、別個の*概説*および*インストール* 資料が用意されています。

プラットフォーム共通のサンプル・プログラム (HTML 形式)
これらのサンプルは、アプリケーション開発クライアントとともにインストールされるサンプル・プログラムの HTML 版です。これらのサンプルは参考用であり、実際のプログラムに代わるものではありません。

リリース情報
これらのファイルには、DB2 ブックには含められなかった最新の情報が記載されています。

インストール情報、リリース情報、およびチュートリアルは、製品 CD-ROM から HTML 形式で参照することができます。ほとんどの資料は、製品 CD-ROM から HTML 形式で表示できますし、 DB2 の資料 CD-ROM から Adobe Acrobat (PDF) 形

式で表示し印刷することができます。 IBM にハードコピー版の資料を注文したい場合は、 285ページの『印刷資料の注文方法』を参照してください。注文可能な資料については、以下の表をご覧ください。

OS/2 および Windows プラットフォームの場合、 HTML ファイルは `sqllib¥doc¥html` ディレクトリーにインストールできます。 DB2 情報はいくつかの言語で提供されています。しかし、すべての言語に翻訳されているわけではありません。ある言語で情報が提供されていない場合は、英語版の情報が提供されます。

UNIX プラットフォームの場合、言語ごとに異なる複数の HTML ファイルを `doc/%L/html` ディレクトリーにインストールできます。ここで、*%L* は地域を表しています。詳細については、適切な*概説およびインストールの手引き* を参照してください。

DB2 ブックを入手して情報を利用するには、次のようなさまざまな方法があります。

- 288ページの『オンライン情報の表示』
- 292ページの『オンライン情報の検索』
- 285ページの『印刷資料の注文方法』
- 285ページの『PDF 資料の印刷』

表 35. DB2 情報

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
	DB2 の手引きおよび解説書情報		
管理の手引き	*管理の手引き: 計画* は、データベース概念について概説し、設計 (たとえば、論理および物理データベース設計) に関する情報を提供し、高い可用性について解説しています。	SC88-8513 db2d1x70	db2d0
	管理の手引き: インプリメンテーション は、設計、データベースへのアクセス、監査、バックアップ、およびリカバリーなどのインプリメンテーションについて説明しています。	SC88-8511 db2d2x70	
	管理の手引き: パフォーマンス は、データベース環境について解説し、さらにアプリケーションのパフォーマンスの評価と調整の方法について説明しています。	SC88-8512 db2d3x70	
管理 API 解説書	データベースの管理に使用できる DB2 アプリケーション・プログラミング・インターフェース (API) およびデータ構造について説明します。また、この資料は、アプリケーションから API を呼び出す方法も示します。	SC88-8514 db2b0x70	db2b0
DB2 アプリケーション構築の手引き	環境設定に関する情報を提供し、Windows、OS/2、および UNIX ベースのプラットフォームでの DB2 アプリケーションのコンパイル、リンク、実行の各ステップについて説明します。	SC88-8515 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	DB2 ユニバーサル・データベース製品をご使用中に発生する可能性のあるセンス・コード APPC、CPI-C、および SNA についての一般情報を提供します。 HTML 形式でのみご利用いただけます。	資料番号なし db2apx70	db2ap

表 35. DB2 情報 (続き)

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
アプリケーション開発の手引き	DB2 データベースにアクセスするアプリケーションを、組み込み SQL または Java (JDBC および SQLJ) を使用して開発する方法について説明します。さらに、ストアード・プロシージャーの作成方法、ユーザー定義関数の作成方法、ユーザー定義タイプの作成方法、トリガーの使用方法、区画化されている環境または統合されているシステムでのアプリケーションの開発方法などについて解説されています。	SC88-8516 db2a0x70	db2a0
コール・レベル・インターフェースの手引きおよび解説書	DB2 データベースにアクセスするアプリケーションを、 DB2 コール・レベル・インターフェース (Microsoft ODBC 仕様互換の呼び出し可能 SQL) を使用して開発する方法について説明します。	SC88-8517 db2l0x70	db2l0
コマンド解説書	コマンド行プロセッサーの使用法について説明し、データベースの管理に使用できる DB2 コマンドについて解説しています。	SC88-8518 db2n0x70	db2n0
コネクティビティー 補足	DB2 (AS/400 版)、DB2 (OS/390 版)、DB2 (MVS 版)、または DB2 (VM 版) を DRDA アプリケーション・リクエスターとして DB2 ユニバーサル・データベース とともに使用するためのセットアップ情報および参照情報を提供します。また、この資料は DRDA アプリケーション・サーバーを DB2 コネクト アプリケーション・リクエスターとともに使用する方法の詳細を示します。 HTML と PDF でのみ利用可能	資料番号なし db2h1x70	db2h1
データ移動ユーティリティー手引きおよび解説書	データの移動を行う DB2 ユーティリティー (インポート、エクスポート、ロード、AutoLoader、および DPROP など) の使用法について説明しています。	SC88-8522 db2dmx70	db2dm

表 35. DB2 情報 (続き)

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
データウェアハウスセンター 管理の手引き	データウェアハウスセンターを使用してデータウェアハウスを構築および保守する方法を説明します。	SC88-8545 db2ddx70	db2dd
データウェアハウスセンター アプリケーション統合の手引き	プログラマーがアプリケーションをデータウェアハウスセンターおよび情報カタログ・マネージャーと統合するのに役立つ情報を提供します。	SC88-8546 db2adx70	db2ad
DB2 コネクト 使用者の手引き	DB2 コネクト製品の概念、プログラミング、および一般的な使用方法に関する情報を提供します。	SC88-8521 db2c0x70	db2c0
DB2 クエリー・パトローラー 管理の手引き	DB2 クエリー・パトローラー・システムの運用の概説を行い、運用および管理に関する詳細情報、および管理用グラフィカル・ユーザー・インターフェース・ユーティリティーについてのタスク情報を提供します。	SC88-8525 db2dwx70	db2dw
DB2 クエリー・パトローラー 使用者の手引き	DB2 クエリー・パトローラーのツールや関数の使用方法を説明します。	SC88-8527 db2wwx70	db2ww
用語集	DB2 およびそのコンポーネントで使用される用語の定義を示します。 HTML 形式と SQL 解説書 で利用可能	資料番号なし db2t0x70	db2t0
イメージ、オーディオ、およびビデオ・エクステンダー 管理およびプログラミング	DB2 エクステンダーの一般情報について提供し、画像、音声、およびビデオ (IAV) エクステンダーの管理と構成について、および IAV エクステンダーを使用したプログラミングについて説明しています。さらに、参照情報、診断情報 (メッセージ解説)、およびサンプルも収録されています。	SC88-8609 dmbu7x70	dmbu7
情報カタログ・マネージャー 管理の手引き	情報カタログを管理するためのガイドです。	SC88-8547 db2dix70	db2di
情報カタログ・マネージャー プログラミングの手引きおよび解説書	情報カタログ・マネージャー用の体系化されたインターフェースの定義を示します。	SC88-8549 db2bix70	db2bi

表 35. DB2 情報 (続き)

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
情報カタログ・マネージャー 使用者の手引き	情報カタログ・マネージャー・ユーザー・インターフェースの使用に関する情報を提供します。	SC88-8548 db2aix70	db2ai
インストールおよび構成補足	プラットフォーム固有の DB2 クライアントの計画、インストール、およびセットアップのガイドです。この補足資料には、バインド、クライアント / サーバー通信の設定、 DB2 GUI ツール、DRDA AS、分散インストール、分散要求の構成、および異種データ・ソースへのアクセスについても説明されています。	GC88-8524 db2iyx70	db2iy
メッセージ解説書	DB2、情報カタログ・マネージャー、およびデータウェアハウスセンターから出されるメッセージとコードをリストし、取るべき処置を解説しています。	第 1 巻 GC88-8543 db2m1x70 第 2 巻 GC88-8544 db2m2x70	db2m0
OLAP Integration Server Administration Guide	OLAP Integration Server の Administration Manager コンポーネントの使用方法を説明します。	SC27-0787 db2dpx70	n/a
OLAP Integration Server Metaoutline User's Guide	標準の OLAP Metaoutline インターフェースを使用して (Metaoutline Assistant を使用するのではなく) OLAP metaoutline を作成しデータを取り込む方法を説明しています。	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	(Model Assistant ではなく) 標準的な OLAP Model Interface を使用して OLAP モデルを作成する方法を説明します。	SC27-0783 db2lpx70	n/a
OLAP のセットアップおよびユーザーズ・ガイド	OLAP スターター・キットの構成およびセットアップに関する情報を提供します。	SC88-8652 db2ipx70	db2ip
Hyperion Essbase スプレッドシート アドイン ユーザーズ ガイド for Excel	Excel 作表計算プログラムを使用して OLAP データを分析する方法を説明します。	SC88-8724 db2epx70	db2ep

表 35. DB2 情報 (続き)

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
Hyperion Essbase スプレッドシート アドイン ユーザーズ ガイド for Lotus 1-2-3	ロータス 1-2-3 作表計算プログラムを使用して OLAP データを分析する方法を説明します。	SC88-8723 db2tpx70	db2tp
DB2 レプリケーションの手引きおよび解説書	DB2 に付属の IBM レプリケーション・ツールの計画、構成、管理、および使用方法に関する情報を提供します。	SC88-8550 db2e0x70	db2e0
地理情報エクステンダー使用者の手引きおよび解説書	地理情報エクステンダーのインストール、構成、管理、プログラミング、およびトラブルシューティングに関する情報を提供します。また、地理情報データの概念についての重要事項を示し、地理情報エクステンダー固有の参照情報 (メッセージおよび SQL) を提供します。	SC88-8624 db2sbx70	db2sb
SQL 概説	SQL の概念を紹介し、構造体とタスクの例を多数提供しています。	SC88-8539 db2y0x70	db2y0
SQL 解説書	SQL の構文、セマンティクス、および言語規則について説明します。また、この資料には、各リリース間の互換性、製品の制限事項、およびカタログ・ビューも含まれます。	第 1 巻 SC88-8540 db2s1x70 第 2 巻 SC88-8657 db2s2x70	db2s0
システム・モニター 手引きおよび解説書	データベースおよびデータベース・マネージャーに関連したさまざまな情報を収集する方法を示します。この資料は、この情報を利用して、データベース活動の把握、パフォーマンス向上、および問題原因の判別を行う方法を説明しています。	SC88-8523 db2f0x70	db2f0

表 35. DB2 情報 (続き)

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
テキスト・エクステンダー *管理およびプログラミング*	DB2 エクステンダーの一般情報、テキスト・エクステンダーの管理および構成情報、およびテキスト・エクステンダーを使用したプログラミングの方法について解説します。この資料には、参照情報、診断情報 (メッセージ解説)、およびサンプルが含まれています。	SC88-8610 desu9x70	desu9
問題判別の手引き	エラーの原因の判別、問題からの回復、および DB2 カスタマー・サービスの支援の下での診断ツールの使用法を記載しています。	GD88-7271 db2p0x70	db2p0
新機能	DB2 ユニバーサル・データベース バージョン 7 の新しい機能および拡張機能について説明します。	SC88-8541 db2q0x70	db2q0
DB2 のインストールおよび構成の情報			
DB2 コネクト エンタープライズ・エディション (OS/2 および Windows 版) 概説およびインストール	OS/2 および Windows 32 ビット・オペレーティング・システム版の DB2 コネクト エンタープライズ・エディションで、計画、移行、インストール、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8520 db2c6x70	db2c6
DB2 コネクト エンタープライズ・エディション (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 コネクト エンタープライズ・エディションの計画、移行、インストール、構成、およびタスクに関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8519 db2cyx70	db2cy

表 35. DB2 情報 (続き)

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
DB2 コネクト パーソナル・エディション 概説およびインストール	OS/2 および Windows 32 ビット・オペレーティング・システムの DB2 コネクト パーソナル・エディションで、計画、移行、インストール、および構成を行う場合のタスク情報を提供します。また、この資料はサポートされているすべてのクライアントのインストールおよびセットアップについても説明します。	GC88-8533 db2c1x70	db2c1
DB2 コネクト パーソナル・エディション (Linux 版) 概説およびインストール	サポートされる Linux 配布プログラムの DB2 コネクト パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8528 db2c4x70	db2c4
DB2 データ・リンク・マネージャー 概説およびインストール	AIX および Windows 32 ビット オペレーティング システムの DB2 データ・リンク・マネージャーで、計画、インストール、構成を行う場合の情報を提供します。	GC88-8532 db2z6x70	db2z6
DB2 エンタープライズ拡張エディション (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 エンタープライズ拡張エディションの計画、インストール、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8530 db2v3x70	db2v3
DB2 エンタープライズ拡張エディション (Windows 版) 概説およびインストール	Windows 32 ビット オペレーティング・システムの DB2 エンタープライズ拡張エディションで、計画、インストール、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8529 db2v6x70	db2v6

表 35. DB2 情報 (続き)

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
DB2 ユニバーサル・データベース (OS/2 版) 概説およびインストール	OS/2 オペレーティング・システムでの DB2 ユニバーサル・データベースの計画、インストール、移行、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8534 db2i2x70	db2i2
DB2 ユニバーサル・データベース (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 ユニバーサル・データベースの計画、インストール、移行、および構成に関する情報を提供します。 また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8536 db2ixx70	db2ix
DB2 ユニバーサル・データベース (Windows 版) 概説およびインストール	Windows 32 ビット・オペレーティング・システムの DB2 ユニバーサル・データベースで、計画、インストール、移行、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8537 db2i6x70	db2i6
DB2 パーソナル・エディション 概説およびインストール	OS/2 および Windows 32 ビット・オペレーティング・システム版の DB2 ユニバーサル・データベース パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8535 db2i1x70	db2i1
DB2 パーソナル・エディション (Linux 版) 概説およびインストール	サポートされる Linux 配布プログラムの DB2 ユニバーサル・データベース パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8538 db2i4x70	db2i4
DB2 クエリー・パトローラー インストールの手引き	DB2 クエリー・パトローラーのインストール情報を提供します。	GC88-8526 db2iwx70	db2iw

表 35. DB2 情報 (続き)

資料名	記述	資料番号 PDF ファイル名	HTML ディレクトリー
ウェアハウス・マネージャー インストールの手引き	ウェアハウス・エージェント、ウェアハウス・トランスフォーマー、および情報カタログ・マネージャーのインストール情報を提供します。	GC88-8572 db2idx70	db2id
プラットフォーム共通のサンプル・プログラム (HTML 形式)			
サンプル・プログラム (HTML)	DB2 のサポートするすべてのプラットフォームでのプログラム言語用に、サンプル・プログラム (HTML 形式) を提供します。これらのサンプル・プログラムは、参照用としてのみ提供されています。サンプルは、すべてのプログラミング言語で利用できるわけではありません。 HTML サンプルが利用できるのは、 DB2 アプリケーション開発クライアントがインストールされている場合だけです。 プログラムの詳細については、*DB2 アプリケーション構築の手引き* を参照してください。	資料番号なし	db2hs
リリース情報			
DB2 コネクト リリース情報	DB2 コネクトの資料には含められなかった最新の情報が収録されています。	注 #2 を参照してください。	db2cr
DB2 インストール情報	DB2 ブックには含められなかったインストールに関する最新の情報が収録されています。	製品 CD-ROM からのみ利用できます。	
DB2 リリース情報	DB2 ブックには含められなかった DB2 製品とその機能に関する最新の情報が収録されています。	注 #2 を参照してください。	db2ir

注:

1. ファイル名の 6 桁目の文字 *x* は、その資料の言語を表します。たとえば、ファイル名 `db2d0e70` は、*管理の手引き* の英語版であることを示し、ファイル名 `db2d0f70` は同じ資料のフランス語版を示します。資料の言語を表すためにファイル名の 6 桁目で使用されている文字は以下のとおりです。

言語	識別子
ブラジル・ポルトガル語	b
ブルガリア語	u
チェコ語	x
デンマーク語	d
オランダ語	q
英語	e
フィンランド語	y
フランス語	f
ドイツ語	g
ギリシャ語	a
ハンガリー語	h
イタリア語	i
日本語	j
韓国語	k
ノルウェー語	n
ポーランド語	p
ポルトガル語	v
ロシア語	r
簡体字中国語	c
スロベニア語	l
スペイン語	z
スウェーデン語	s
繁体字中国語	t
トルコ語	m

2. DB2 ブックには含められなかった最新の情報が、「リリース情報」で HTML 形式および ASCII ファイルとして利用できます。 HTML 版は、インフォメーション・センターおよび製品 CD-ROM からご利用になれます。 ASCII ファイルの参照方法:
   - UNIX ベースのプラットフォームでは、ファイル `Release.Notes` を参照してください。このファイルは `DB2DIR/Readme/%L` ディレクトリーにあります。ここで `%L` は地域名を、 `DB2DIR` は以下のものを表します。
     - `/usr/lpp/db2_07_01` (AIX の場合)
     - `/opt/IBMdb2/V7.1` (HP-UX、DYNIX/ptx、Solaris、および Silicon Graphics IRIX の場合)
     - `/usr/IBMdb2/V7.1` (Linux の場合)
   - これ以外のプラットフォームでは、ファイル `RELEASE.TXT` を参照してください。このファイルは、製品がインストールされているディレクトリーにあります。OS/2 プラットフォームでは、 **IBM DB2** フォルダーをダブルクリックし、**Release Notes** アイコンをダブルクリックすることもできます。

PDF 資料の印刷

資料のハードコピー版が必要な場合、 DB2 の資料 CD-ROM にある PDF ファイル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することができます。ライブラリー内の各資料のファイルについては、275ページの表35 を参照してください。

Adobe Acrobat Reader の最新版は、Adobe の Web サイト http://www.adobe.co.jp/ から入手できます。

PDF ファイルは、DB2 の資料 CD-ROM に収録されており、ファイル拡張子 PDF が付いています。 PDF ファイルにアクセスするには以下のようにします。

1. DB2 の資料 CD-ROM を挿入します。 UNIX ベースのプラットフォームの場合は、DB2 資料 CD-ROM をマウントします。マウントの手順については、*概説およびインストール* を参照してください。
2. Acrobat Reader を起動します。
3. 以下に示すいずれかの位置から必要な PDF ファイルを開きます。
   - OS/2 および Windows プラットフォームでは:

     x:¥doc¥*language* ディレクトリー。ここで、*x* は CD-ROM ドライブを、*language* は 2 桁の言語を表す国コード (たとえば、EN は英語) を示します。
   - UNIX ベースのプラットフォームでは:

     CD-ROM の */cdrom*/doc/*%L* ディレクトリー。ここで、*/cdrom* は CD-ROM のマウント・ポイントを、 *%L* は地域名を表します。

さらに、PDF ファイルを CD-ROM からローカル・ドライブまたはネットワーク・ドライブにコピーし、そこから参照することもできます。

印刷資料の注文方法

ハードコピー版の DB2 ブックは、個別に注文することができます。資料を注文するには、IBM 承認の販売業者または営業担当員に連絡してください。

DB2 オンライン文書

オンライン・ヘルプへのアクセス

すべての DB2 コンポーネントで、オンライン・ヘルプを利用できます。以下の表に、さまざまな種類のヘルプを示します。

ヘルプの種類	内容	利用方法
コマンド・ヘルプ	コマンド行プロセッサーのコマンド構文について説明します。	コマンド行プロセッサーの対話モードから、次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 コマンドのヘルプが表示されます。
クライアント構成アシスタントのヘルプ コマンド・センターのヘルプ コントロール・センターのヘルプ データウェアハウスセンターのヘルプ イベント・アナライザーのヘルプ 情報カタログ・マネージャーのヘルプ サテライト管理センターのヘルプ スクリプト・センターのヘルプ	そのウィンドウまたはノートブックで実行できるタスクについて説明します。このヘルプは、知っておく必要のある概説および前提条件に関する情報を含みます。また、ウィンドウやノートブックの制御の使用方法を示します。	ウィンドウまたはノートブックから、**「ヘルプ (Help)」**押しボタンをクリックするか、または **F1** キーを押します。

ヘルプの種類	内容	利用方法

メッセージ・ヘルプ	メッセージの原因、および取るべき処置を説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。
		`? XXXnnnnn`
		ここで、*XXXnnnnn* は有効なメッセージ識別子を表します。
		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。
		一度に 1 画面分のメッセージ・ヘルプを表示させるには、次のように入力します。
		`? XXXnnnnn | more`
		メッセージ・ヘルプをファイルに保管するには、次のように入力します。
		`? XXXnnnnn > filename.ext`
		ここで、*filename.ext* はメッセージ・ヘルプを保管するファイルを表します。

SQL ヘルプ	SQL ステートメントの構文について説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `help statement` ここで、*statement* は SQL ステートメントを表します。 たとえば、`help SELECT` と入力すると、 SELECT ステートメントのヘルプが表示されます。 **注:** UNIX ベースのプラットフォームでは、SQL ヘルプを利用できません。
SQLSTATE ヘルプ	SQL 状態およびクラス・コードについて説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `? sqlstate or ? class code` ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。 たとえば、`? 08003` によって SQL 状態 08003 のヘルプが表示され、 `? 08` によってクラス・コード 08 のヘルプが表示されます。

オンライン情報の表示

この製品に付属のブックは、ハイパーテキスト・マークアップ言語 (HTML) ソフトコピー形式です。ソフトコピー形式では情報を検索または表示したり、ハイパーテキスト・リンクを利用して関連情報に移動したりすることができます。また、1 つの端末を超えてライブラリーを容易に共用することができます。

オンライン・ブックやサンプル・プログラムは、 HTML バージョン 3.2 仕様に準拠するすべてのブラウザーを使って表示できます。

オンライン・ブックまたはサンプル・プログラムは、次のようにして表示します。

- DB2 管理ツールを実行している場合、インフォメーション・センターを使用します。
- ブラウザーで、**ファイル (File) —>ページを開く (Open Page)** をクリックします。次のようなページを開いて、DB2 情報に関する説明とリンクを表示してください。
  - UNIX ベースのプラットフォームでは、以下のページを開きます。

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    ここで *%L* はロケール名です。
  - その他のプラットフォームでは、以下のページを開きます。

    sqllib¥doc¥html¥index.htm

    パスは DB2 がインストールされているドライブです。

    インフォメーション・センターをインストールしていない場合、 **DB2 Information** アイコンをダブルクリックしてページを開くことができます。このアイコンは、ご使用のシステムに応じて、製品のメイン・フォルダー内または Windows 「スタート」メニューにあります。

Netscape ブラウザーのインストール

システムに Web ブラウザーがインストールされていない場合、製品の箱の中にある Netscape CD-ROM から Netscape をインストールすることができます。インストールに関する詳細な説明については、以下を参照してください。

1. Netscape CD-ROM を挿入します。
2. UNIX ベースのプラットフォームでは、CD-ROM をマウントします。マウントの手順については、*概説およびインストール* を参照してください。
3. インストールの手順については、 CDNAV*nn*.txt ファイルを参照します。ここで、*nn* は 2 桁の言語識別子を表します。ファイルは CD-ROM のルート・ディレクトリーにあります。

インフォメーション・センターを使用した情報へのアクセス

インフォメーション・センターを使用すると、DB2 製品情報にすばやくアクセスすることができます。インフォメーション・センターは、DB2 管理ツールを使用できるすべてのプラットフォームで利用できます。

インフォメーション・センターは「インフォメーション・センター (Information Center)」アイコンをダブルクリックすることによってオープンできます。このアイコンのある場所はシステムによって異なります。メイン・プロダクト・フォルダーか Windows の**「スタート」**メニューのどちらかです。

Windows プラットフォームの DB2 では、ツールバーおよび**ヘルプ**・メニューを使用して、インフォメーション・センターにアクセスすることもできます。

インフォメーション・センターは 6 種類の情報を提供します。適切なタブをクリックすると、種類ごとに提供されているトピックが表示されます。

タスク (Tasks) DB2 を使用して実行できる主要なタスク。

参照 (Reference)
DB2 参照情報 (キーワード、コマンド、API など)。

ブック (Books) DB2 ブック。

トラブルシューティング (Troubleshooting)
エラー・メッセージのカテゴリーと、メッセージに対する回復処置。

サンプル・プログラム (Sample Programs)
DB2 アプリケーション開発クライアントに付属のサンプル・プログラム。 DB2 アプリケーション開発クライアントをインストールしていない場合、このタブは表示されません。

Web WWW 上にある DB2 情報。この情報にアクセスするには、ご使用のシステムから Web への接続が必要です。

リストから項目を 1 つ選択すると、インフォメーション・センターはビューアーを立ち上げて情報を表示します。選択した情報の種類に応じて、ビューアーはシステム・ヘルプ・ビューアー、エディター、または Web ブラウザーです。

インフォメーション・センターには検索機能が備わっており、リストを参照せずに特定のトピックを探すことができます。

テキストの全検索を行うには、インフォメーション・センター内のハイパーテキスト・リンク「**DB2 オンライン情報の検索 (Search DB2 Online Information)**」検索フォームに従います。

通常、HTML 検索サーバーは自動的に始動します。 HTML 情報の検索がうまくいかない場合は、以下の方法の 1 つを使用して、検索サーバーを始動しなければならない場合もあります。

Windows では

「スタート」をクリックし、**「プログラム」—>「IBM DB2」—>「Information」—>「Start HTML Search Server」**を選択します。

OS/2 では

「DB2 (OS/2 版)」フォルダーをダブルクリックして、**「Start HTML Search Server」**アイコンをダブルクリックします。

HTML 情報の検索でこの他の問題が発生した場合は、リリース情報を参照してください。

注: 検索機能は、Linux、DYNIX/ptx、および Silicon Graphics IRIX 環境では利用できません。

DB2 ウィザードの使用

ウィザードを使用すると、各タスクをステップごとに進めることによって、さまざまな管理タスクを遂行することができます。ウィザードは、コントロール・センターおよびクライアント構成アシスタントを通して使用できます。以下の表では、ウィザードとその目的をリストしています。

注: データベース作成、索引作成、マルチサイト更新の構成、およびパフォーマンス構成ウィザードは、区分データベース環境で使用できます。

ウィザード	内容	利用方法
データベース追加 (Add Database)	クライアント・ワークステーション上にデータベースのカタログを作成します。	クライアント構成アシスタントから、「**追加 (Add)**」をクリックします。
データベース・バックアップ *(Back up Database)*	バックアップ計画を決定、作成、およびスケジュールします。	「コントロール・センター (Control Center)」からバックアップするデータベースを右クリックし、**「バックアップ (Backup)」—>「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。
マルチサイト更新の構成 (Configure Multisite Update)	マルチサイト更新、分散トランザクション、または 2 フェーズ・コミットを構成します。	「コントロール・センター (Control Center)」から、**「データベース (Databases)」**フォルダーを右クリックして、**「マルチサイト更新 (Multisite Update)」**を選択します。

ウィザード	内容	利用方法
データベース作成 (Create Database)	データベースを作成し、いくつかの基本的な構成タスクを実行します。	「コントロール・センター (Control Center)」から、**「データベース (Databases)」**フォルダーを右クリックして、**「作成 (Create)」—> 「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。
表作成 (Create Table)	基本的なデータ・タイプを選択して、表の基本キーを作成します。	「コントロール・センター (Control Center)」から、**「表 (Tables)」**アイコンを右クリックして、**「作成 (Create)」—>「ウィザードを使用する表 (Table Using Wizard)」**を選択します。
表スペース作成 (Create Table Space)	新しい表スペースを作成します。	「コントロール・センター (Control Center)」から、**「表スペース (Table Spaces)」**アイコンを右クリックして、**「作成 (Create)」—> 「ウィザードを使用する表スペース (Table Space Using Wizard)」**を選択します。
索引作成 (Create Index)	すべての照会について、作成すべき索引および除去すべき索引を提案します。	「コントロール・センター (Control Center)」から、**「索引 (Index)」**アイコンを右クリックして、**「作成 (Create)」—>「ウィザードを使用する索引 (Index Using Wizard)」**を選択します。
パフォーマンス構成 (Performance Configuration)	ビジネス要件に適合するように構成パラメーターを更新して、データベースのパフォーマンスを調整します。	「コントロール・センター (Control Center)」から、調整したいデータベースを右クリックして、**「ウィザードを使用するパフォーマンスの構成 (Configure Performance Using Wizard)」**を選択します。 区分データベース環境では、「Database Partitions」視点から、調整したい最初のデータベース区画を右クリックして、**「ウィザードを使用するパフォーマンスの構成 (Configure Performance Using Wizard)」**を選択します。

ウィザード	内容	利用方法
データベース復元 *(Restore Database)*	障害の後、データベースを回復します。どのバックアップを使用し、どのログを再生するかの判別を支援します。	「コントロール・センター (Control Center)」から復元するデータベースを右クリックし、**「復元 (Restore)」—>「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。

文書サーバーのセットアップ

デフォルトでは、DB2 情報はローカル・システムにインストールされます。つまり、DB2 情報にアクセスする必要のある各担当者が同じファイルをインストールする必要があります。 DB2 情報を 1 か所に格納するには、次のようにします。

1. ¥sqllib¥doc¥html のすべてのファイルとサブディレクトリーを、ローカル・システムから Web サーバーにコピーします。各ブックには独自のサブディレクトリーがあり、そのブックを構成する必要な HTML および GIF ファイルが入っています。ディレクトリー構造は常に同じ状態に保つ必要があります。
2. Web サーバーを構成して、ファイルを新しい場所で検索するようにします。さらに詳しい情報については、 *インストールおよび構成 補足* の NetQuestion 付録を参照してください。
3. インフォメーション・センターの Java バージョンをご使用の場合は、すべての HTML ファイルのベース URL を指定できます。この URL はブックのリストに使用してください。
4. 資料ファイルが表示されるようになったなら、よく使うトピックにはブックマークを付けておいてください。ブックマークを付けるページは、たとえば以下のものがあります。
   - ブックのリスト
   - 頻繁に使用されるブックの目次
   - 頻繁に参照する情報 (たとえば、ALTER TABLE トピックなど)
   - 検索フォーム

中央のマシンから DB2 ユニバーサル・データベース オンライン文書ファイルを提供する方法については、 *インストールおよび構成 補足* の NetQuestion 付録を参照してください。

オンライン情報の検索

HTML ファイルの情報を検索するには、以下の方法のどれか 1 つを使用してください。

- 最上部にある**「検索 (Search)」**をクリックします。検索フォームを使用して特定のトピックを見つけます。この機能は、 Linux、DYNIX/ptx、または Silicon Graphics IRIX 環境ではご利用になれません。
- 最上部にある**「索引 (Index)」**をクリックします。索引を使用して、ブック内の特定のトピックを見つけます。
- HTML 資料またはヘルプの目次あるいは索引を表示してから、 Web ブラウザーの検索機能を利用してブック内の特定のトピックを見つけます。
- Web ブラウザーのブックマーク機能を使用して、特定のトピックにすばやく戻ります。
- インフォメーション・センターの検索機能を使用して、特定のトピックを検索します。詳しくは、289ページの『インフォメーション・センターを使用した情報へのアクセス』を参照してください。

付録F. 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、プログラムまたはサービスの操作性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。実施権の許諾については、下記の宛先に書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
AP 事業所
IBM World Trade Asia Corporation
Intellectual Property Law & 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。

本書は定期的に見直され、必要な変更 (たとえば、技術的に不適切な表現や誤植など) 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 Ltd.
Office of the Lab Director
1150 Eglinton Avenue East
Toronto, Ontario
M3C 1H7
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・プログラムのすべての部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

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

商標

アスタリスク (*) 付きの以下の用語は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

以下は、それぞれ各社の商標または登録商標です。

Tivoli および、NetView は、Tivoli Systems, Inc. の商標です。

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。

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。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名などはそれぞれ各社の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]

C

D

G

H

I

L

M

N

P

R

S

T

U

W

X

IBM と連絡をとる

技術上の問題がある場合は、時間をとって*問題判別の手引き* に定義されている処置を検討し、それらの提案を実行した後で、DB2 顧客サービスに連絡をとってください。この資料には、DB2 顧客サービスがお客さまを支援するために必要とする情報が説明されています。

製品情報

以下の情報は英語で提供されます。内容は英語版製品に関する情報です。

http://www.ibm.com/software/data/

DB2 World Wide Web ページには、ニュース、製品説明、研修スケジュールなどの DB2 に関する最新情報が提供されています。ただし、提供されている情報は英語です。

http://www.ibm.com/software/data/db2/library/

「DB2 Product and Service Technical Library」では、よくされる質問 (FAQ)、修正内容、資料、および最新の DB2 技術情報などの情報へのアクセスが提供されています。

注: この情報のご提供は英語のみとなりますのでご注意ください。

http://www.elink.ibmlink.ibm.com/pbl/pbl/

「International Publications」注文用 Web サイトでは、マニュアルの注文方法についての情報を提供しています。ただし、提供されている情報は英語です。

http://www.ibm.com/education/certify/

IBM の「Professional Certification Program」Web サイトでは、 DB2 を含むさまざまな IBM 製品の認証テストの情報を提供しています。ただし、提供されている情報は英語です。

ftp.software.ibm.com

匿名でログオンしてください。ディレクトリー `/ps/products/db2` には、 DB2 および多数の他製品に関連したデモ、修正プログラム、情報、およびツールがあります。ただし、提供されている情報は英語です。

comp.databases.ibm-db2, bit.listserv.db2-l

これらのインターネット・ニュースグループは、ユーザーが DB2 製品に関する自分の経験について話し合うために利用できます。ただし、提供されている情報は英語です。

Compuserve: GO IBMDB2

このコマンドを入力すると、IBM DB2 Family forum にアクセスできます。すべての DB2 製品が、このフォーラムでサポートされています。ただし、提供されている情報は英語です。

米国以外の国で IBM に連絡する方法については、 *IBM Software Support Handbook* の Appendix A を参照してください。この資料にアクセスするには、 Web ページ: http://www.ibm.com/support/ にアクセスし、ページの最下部にある「IBM Software Support Handbook」リンク・ボタンを選択します。

注: 国によっては、 IBM が承認している販売業者が、 IBM サポート・センターの代わりにそれら販売業者のサポート・センターに連絡する場合があります。

Printed in Japan

SC88-8513-01

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