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

SC88-8513-00
(英文原典：SC09-2946-00)

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

SC88-8513-00
(英文原典：SC09-2946-00)

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、 473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。

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミング、またはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミング、またはサービスを、日本で発表する意図があることを必ずしも示すものではありません。

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:	SC09-2946-00 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 2000.6

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・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・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回復、ジョブのスケジューリング、および媒体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、機密保護管理者、およびシステム操作員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、回復オブジェクト、記憶域オブジェクト、およびシステム・オブジェクトを含むデータベース・オブジェクトの概要を説明します。

- 『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 で実現される並行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 (データ記憶域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 高可用性の設計』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。

高可用性システム

- 『第12章 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)』では、 DB2 による AIX での高可用性フェールオーバー・サポートについて説明します。
- 『第13章 Windows NT 環境での高可用性』では、 DB2 による Windows NT での高可用性フェールオーバー・サポートについて説明します。
- 『第14章 DB2 と、Sun Cluster 2.2 での高可用性』では、 DB2 による Sun Solaris オペレーティング・システムでの高可用性フェールオーバー・サポートについて説明します。

付録

- 『付録B. 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C. データベース移行の計画』では、第 7 版へのデータベースの移行について説明します。
- 『付録D. リリース間の非互換性』では、第 7 版までに導入された、互換性のない機能について説明します。
- 『付録E. 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A. 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、データベースを管理するのに使用するグラフィカル・ユーザー・インターフェース (GUI) ツールを紹介します。

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。

データベースの機密保護

- 『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。

回復

- 『データベースの回復』では、データベースおよび表スペースの回復方式 (データベースおよび表スペースのバックアップおよび回復、およびロールフォワード回復方式の使用を含む) を選択する場合の考慮事項について説明します。

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。
- 『データベース回復用のユーザー出口』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 機密保護を処理する方法』では、DB2 が Windows NT 機密保護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 機能が使用する表についての情報を示し、それらの表の作成方法について説明します。
- 『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 (369 ページの『付録A. 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 で実現される並行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構成要素は、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。
- 『高可用性システム』。このセクションでは、 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。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、あるプロセッサーから別のプロセッサーにワークロードを自動的に移すことが可能になります。

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、 *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空間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移行計画、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、データベース機密保護、データベース回復、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『回復オブジェクトの概要』
- 14ページの『記憶域オブジェクトの概要』
- 20ページの『システム・オブジェクトの概要』
- 22ページの『データに関する業務規則』
- 26ページの『データベースの回復』
- 55ページの『データベースでの表の再編成』
- 56ページの『DB2 の機密保護の概説』

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。

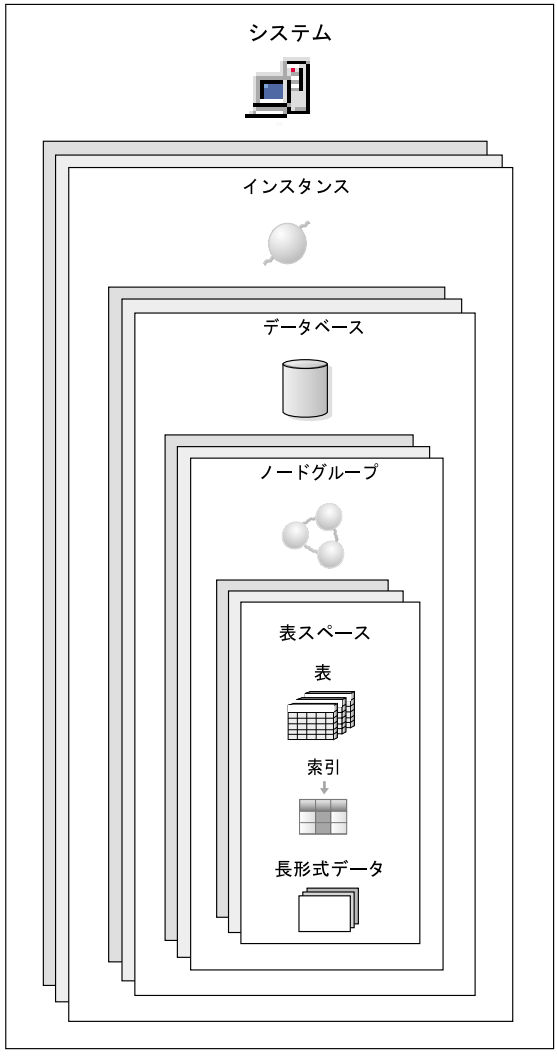


図 1. いくつかのデータベース・オブジェクトの関係

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。

(65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機密保護も持っています。

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回復ログが含まれます。

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、66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、65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、14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、14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、構造化照会言語 (SQL、*SQL 解説書*を参照) (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) 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、また永続的な記憶域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。

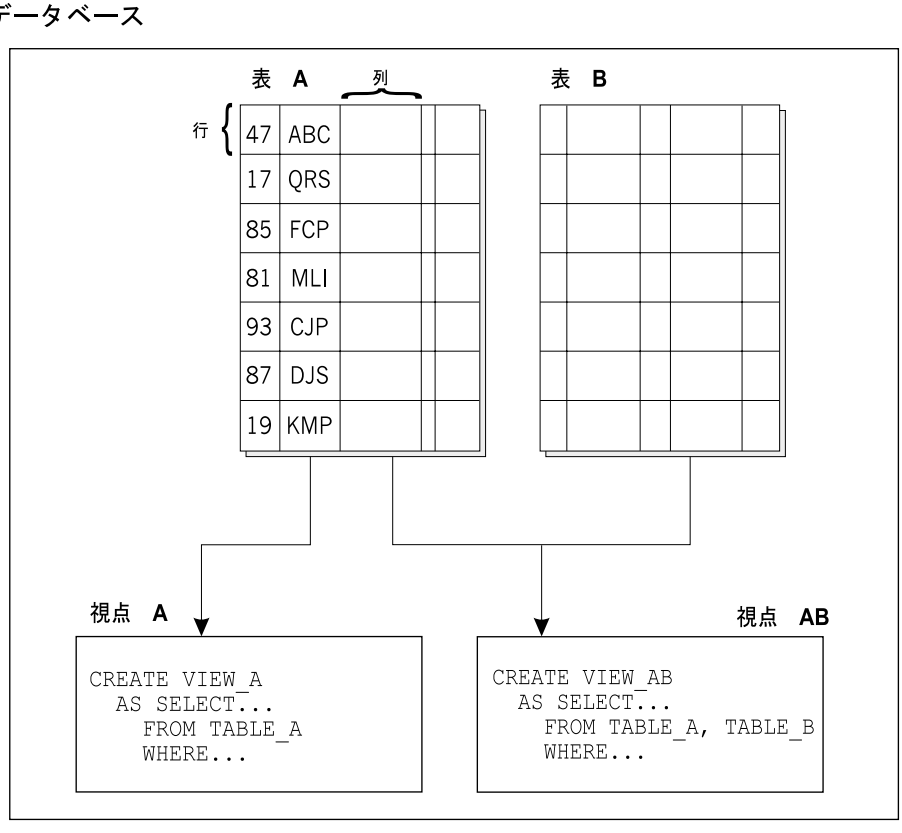


図 2. 表と視点の関係

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、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。 22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。

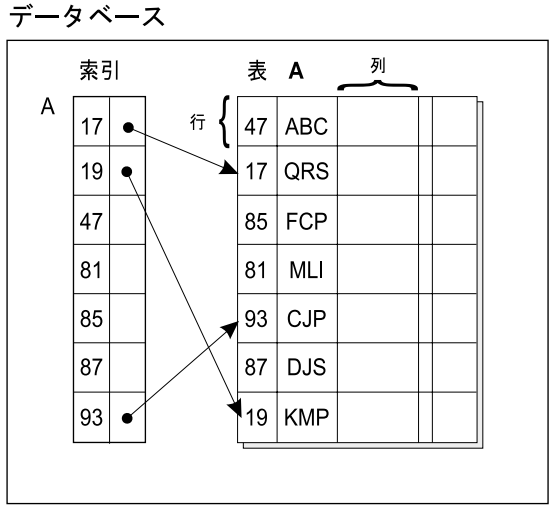


図 3. 索引と表の関係

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 などの識別子です。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、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機密保護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。

回復オブジェクトの概要

データベースを作成すると、ログ・ファイルと回復活動記録ファイルが自動的に作成されます (13ページの図4)。ログ・ファイルや回復活動記録ファイルは直接変更できません。これらは、消失または損傷したデータを回復するためにデータベース・バックアップ・イメージを使用する必要があるときに重要になります。

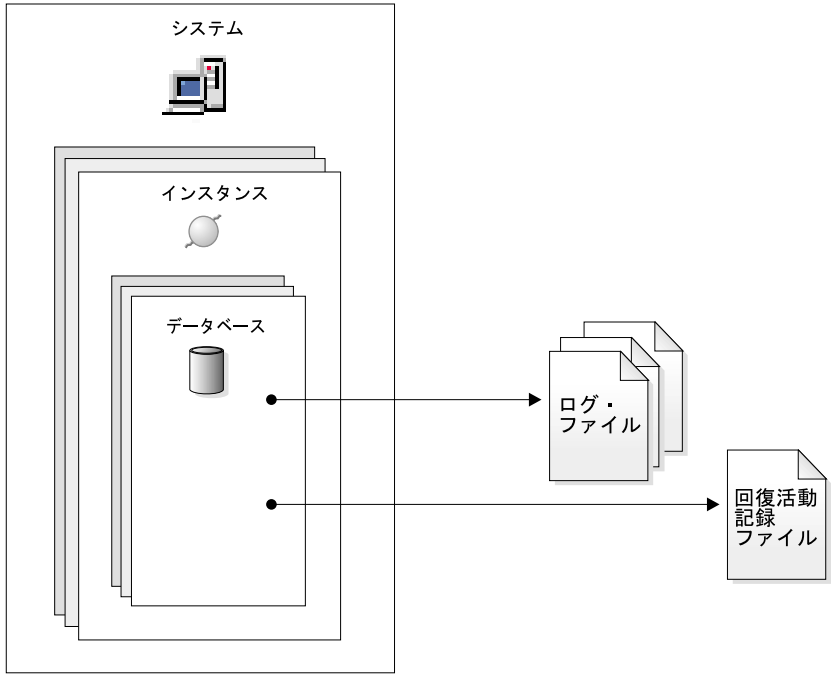


図 4. ログ・ファイルおよび回復活動記録ファイル

回復ログ・ファイル

それぞれのデータベースには*回復ログ* が含まれており、これはアプリケーションまたはシステム・エラーから回復するときに使用します。データベース・バックアップと組み合わせて、これらはデータベースの整合性をエラーが生じた時点まで回復するために使用されます。データベース回復の詳細については、26ページの『データベースの回復』を参照してください。

回復活動記録ファイル

回復活動記録ファイル 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回復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回復関連のイベントを追跡するために使用します。データベースのバックアップおよび復元の

手順については、 26ページの『データベースの回復』を参照してください。ロード・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* で説明しています。

記憶域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記憶域装置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。

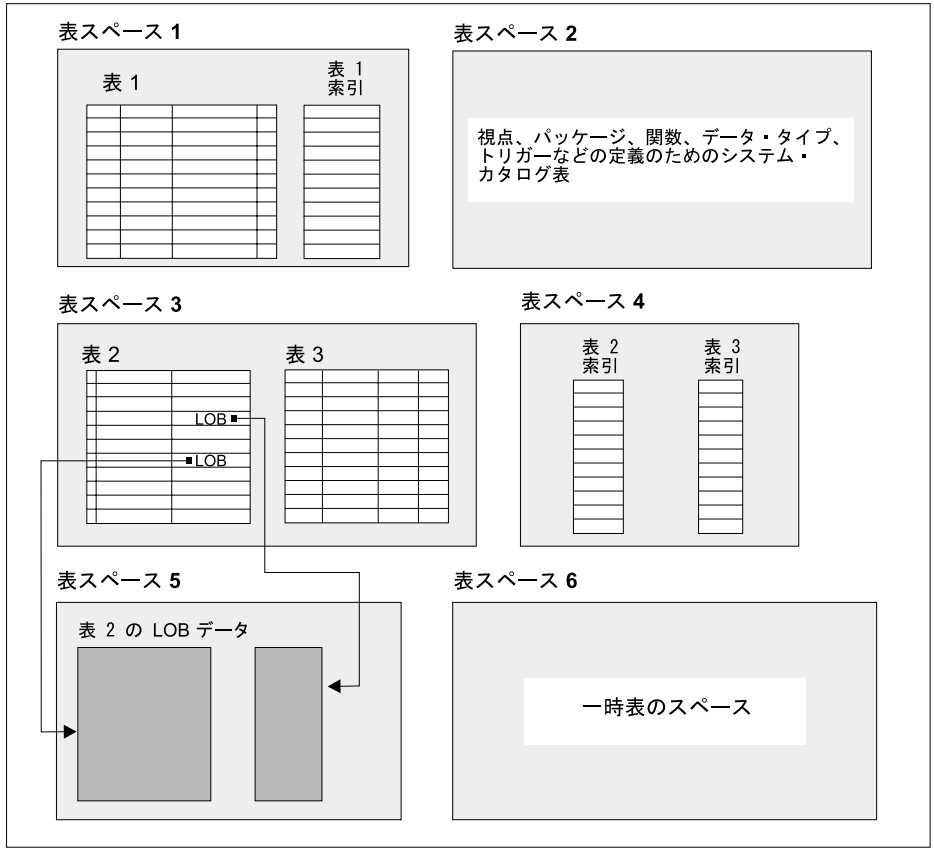


図5. 表スペース

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、割り振られた物理記憶域 (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記憶域を制御します。 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・管理プログラムが記憶域を制御します。

図6 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。

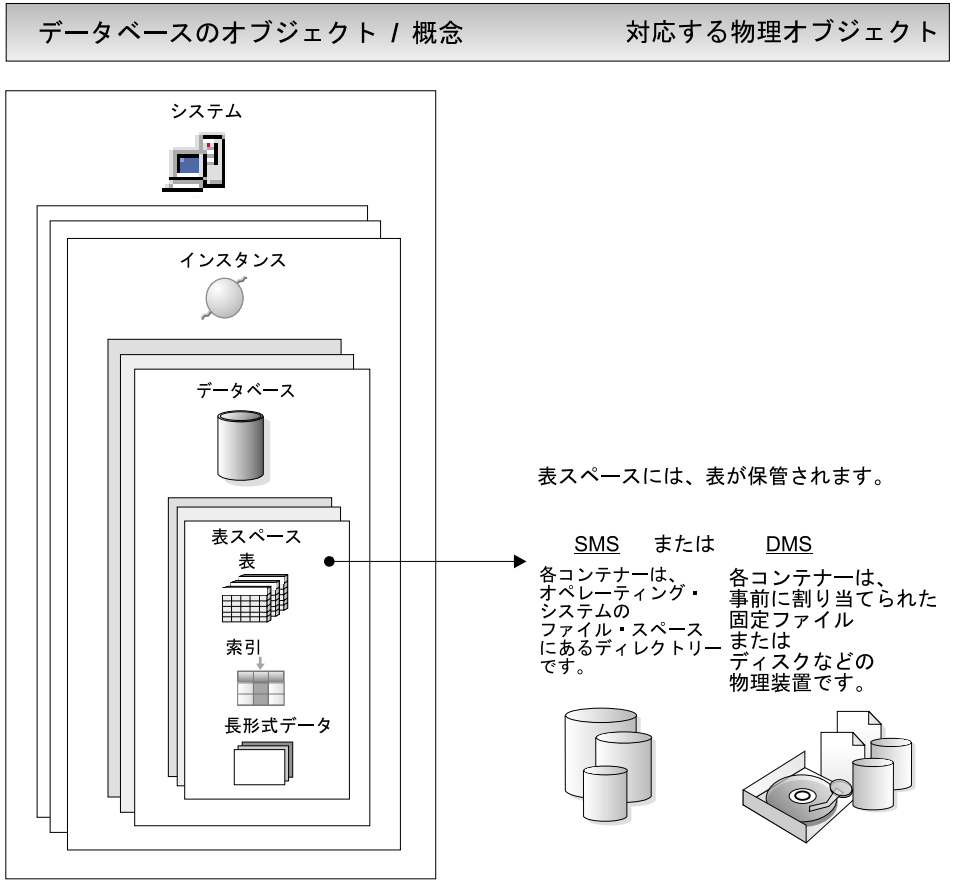


図 6. 表スペースと表

17ページの図7 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分類されます。システム一時表スペース は、ソート、表の再編成、索引の作成、および表の結合などの SQL 操作中に必要となる内部一時データを保管するために使用します。システム一時表スペースはいくつでも作成できますが、大多数の表が使用するページ・サイズを使用して 1 つだけ作成することをお勧めします。デフォルトのシステム一時表スペースは TEMPSPACE1 と呼ばれています。ユーザー一時表スペース は、アプリケーション一時データを保管する宣言済み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、省略時にはデータベース作成の時点で作成されません。

図 7. 3 つの表スペース・タイプ

コンテナー

コンテナー は、物理記憶域装置です。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8 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。

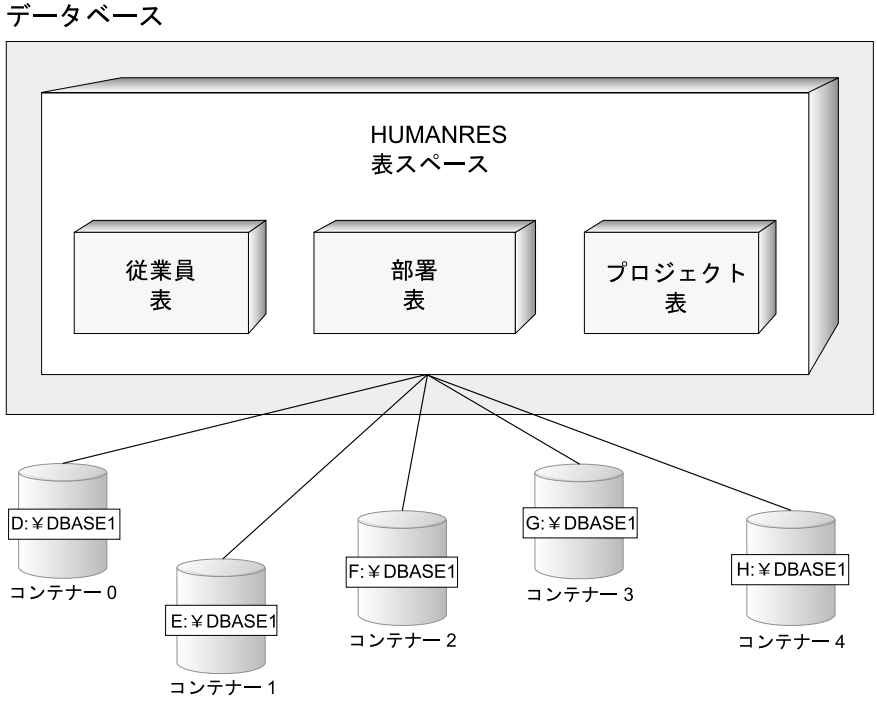


図 8. データベース内の表スペースと表

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。

20ページの図9 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。

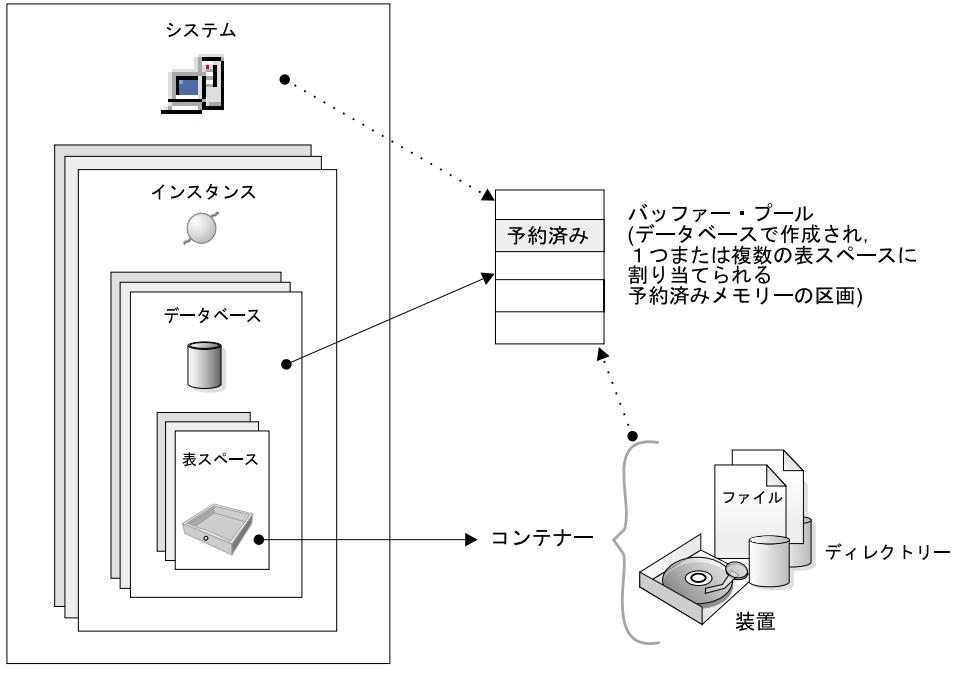


図 9. バッファー・プールとコンテナー

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 (22ページの図10 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。

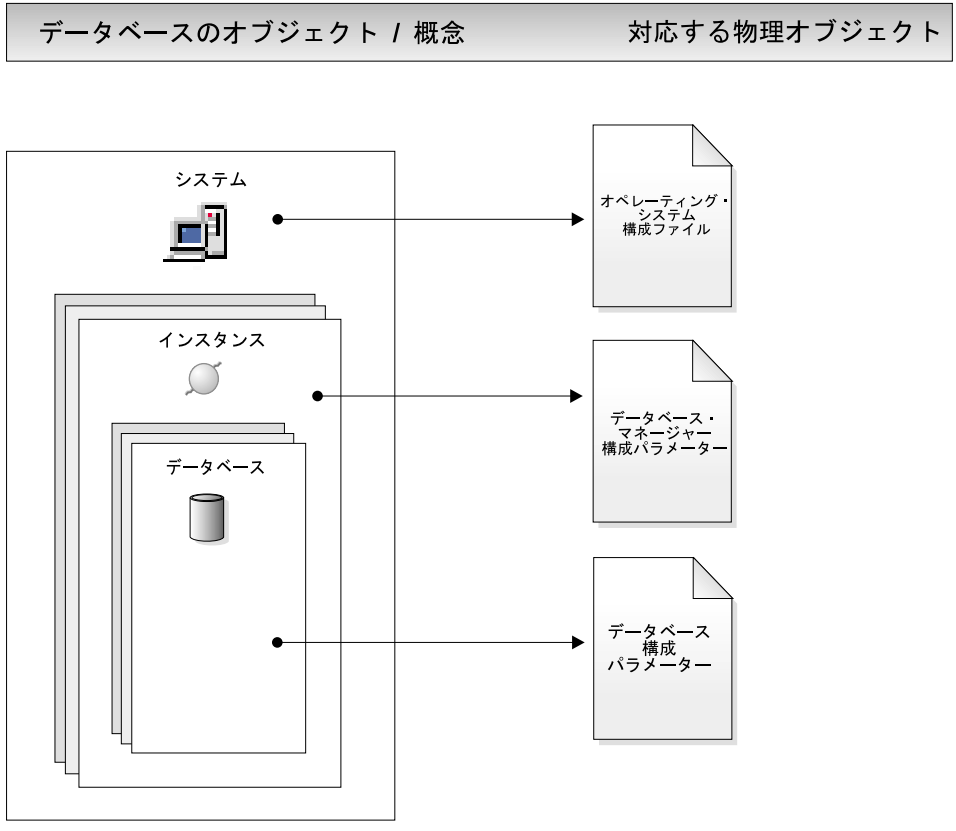


図 10. 構成パラメーター・ファイル

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。

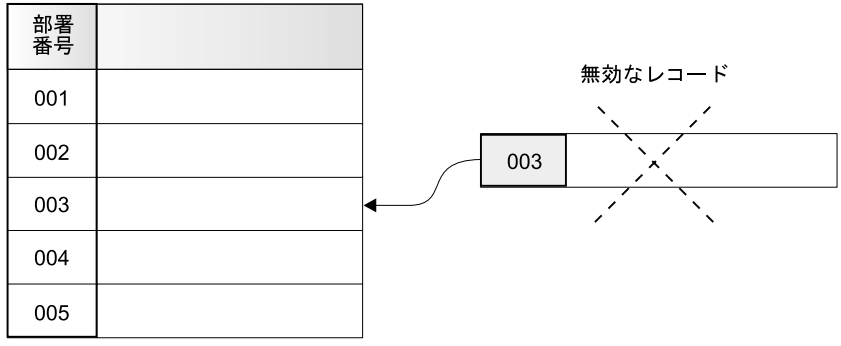


図 11. 固有制約は重複データを防ぐ

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

表 1. DEPARTMENT 表 (続き)

DEPTNO (基本キー)	DEPTNAME	MGRNO
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。

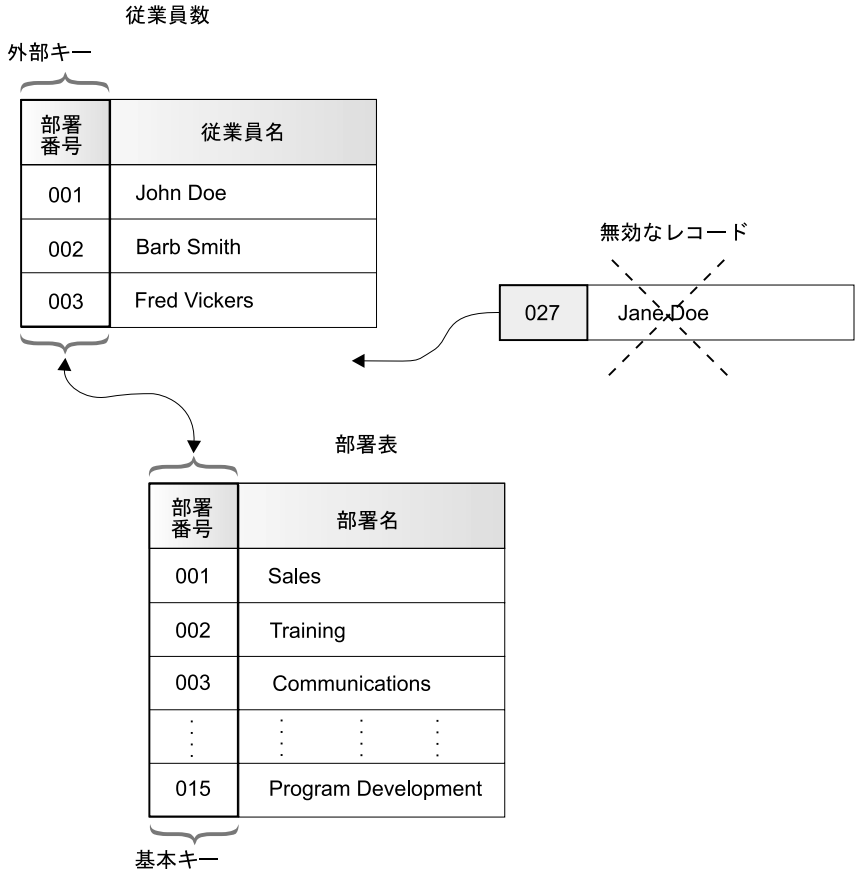


図 12. 外部キー制約および基本キー制約は関係を定義してデータを保護する

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、指定した基礎表に対する 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。

データベースの回復

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回復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。

このセクションでは、各種の回復方式について説明し、業務環境に最適な回復方式を判別する方法を示します。この部分では、次の点について説明します。

- 『回復とは何か』
- 33ページの『回復に影響する要素』
- 50ページの『災害時回復の考慮事項』
- 51ページの『媒体障害の影響の緩和』
- 53ページの『トランザクション障害の影響の緩和』
- 54ページの『区分データベース・システムにおけるシステム・クロックの同期』

回復とは何か

データベース上の問題が発生した場合は、使用可能な戦略についての知識が必要になります。これらには、媒体や記憶域の問題、電源停止、およびアプリケーション上の障害が含まれます。データベースまたは個々の表スペースをバックアップして、データベースが壊れた場合にそれらを再作成することができます。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、異なる媒体に保管します。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。

このデータベースの再作成のことを*回復* といいます。*破損回復* は、障害が発生した後にデータベースの自動的な回復を試みます。損傷したデータベースを回復するには、*バージョン回復* と*ロールフォワード回復* の 2 通りの方法があります。

回復不能データベースでは、 *logretain* と *userexit* の両方のデータベース構成パラメーターが使用不能になっています。これは、保管されているログのみが破損回復に必要なログであることを意味します。これらのログは、*アクティブ・ログ* と呼ばれ、現在のトランザクション・データを含んでいます。オフライン・バックアップを使ってバージョン回復を行うことは、基本的には回復不能データベースの回復を行うことを意味します。 (オフライン・バックアップは、バックアップ操作の進行中は、他のアプリケーションがこのデータベースを使用できないという意味です。) このようなデータベースは、オフラインでのみ復元できます。復元すると、バックアップ・イメージが取られたときの状態に戻ります。

回復可能データベースでは、 *logretain*データベース構成パラメーターが ″RECOVERY″ に設定されているか、 *userexit* データベース構成パラメーターが使用可能になっています。または、これらの両方が行われています。アクティブ・ログを破損回復で使用できますが、*アーカイブ・ログ* もあり、これにはコミット済みトランザクション・データが含まれています。このようなデータベースは、オフラインでのみ復元できます。復元すると、バックアップ・イメージが取られたときの状態に戻ります。ただし、ロールフォワード回復では、アクティブ・ログおよびアーカイブ・ログを使用することによって、特定の時点またはアクティブ・ログの最後までデータベースをロール・フォワード する (つまり、バックアップ・イメージが取られたときよりも進める) ことができます。

回復可能データベースのバックアップ操作はオフラインでもオンライン でも実行できます (オンラインとは、バックアップ操作中に他のアプリケーションがそのデータベースに接続できるという意味です)。データベースの復元とロールフォワード操作は、常にオフラインで実行しなければなりません。バックアップ操作中にロールフォワード回復は、すべての 表の変更がキャプチャーされ、バックアップが復元された時に再適用されることを保証します。

回復可能データベースがある場合は、データベース全体の代わりに、個々の表スペースをバックアップ、復元、およびロールフォワードすることができます。表スペースをオンラインでバックアップするとき、その表スペースは依然として使用可能であり、同時に行われる更新がロギングされます。表スペースに対してオンライン復元またはロールフォワード操作を実行するときは、表ス

ページ自体は操作が完了するまで使用できませんが、ユーザーが他の表スペースにアクセスできないということはありません。

破損回復は、データベースが、矛盾したまたは使用不能な状態のままになることを防ぎます。データベースに対するトランザクション (つまり作業単位) 予期しない割り込みを受けることがあります。たとえば、作業単位の一部となるすべての変更内容が完了しコミットされる前に、障害が発生すると、データベースは矛盾した、または使用不能な状態のままになっています。

この場合は、データベースを一貫した使用可能な状態にする必要があります。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 (図13)。

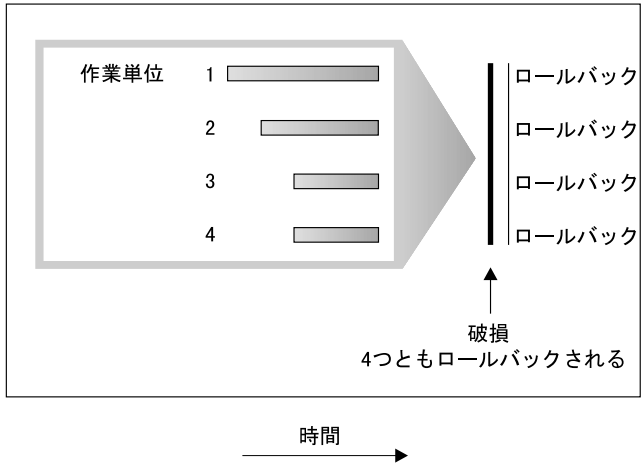


図 13. 作業単位のロールバック

データベースが整合性があり使用可能な状態の場合には、これは ″一貫性ポイント″ にあることになります。オフライン・データベース・バックアップは、一貫性ポイントを表しています。一貫性ポイントに達したとき、すべてのトランザクションは解決されており、他のユーザーまたはアプリケーションでデータが使用できます。

破損の後には、RESTART DATABASE コマンド呼び出して、一貫性ポイントに移動できます (*コマンド解説書* を参照)。障害が発生するたびにこの処理を実行したい場合は、**自動再始動可能** (*autorestart*) 構成パラメーターの使用を考慮してください。このデータベース構成パラメーターのデフォルト動作は、必要なときにはいつでも RESTART DATABASE コマンドを呼び出すことです。

autorestart が使用可能になっていると、障害後にデータベースに対し次の接続要求が出された時点で、 RESTART DATABASE コマンドが呼び出されます。

破損回復を行うと、データベースは一貫性のある、使用可能な状態になります。ただし、順方向回復が使用可能になっている (つまり、 *logretain* 構成パラメーターが ″RECOVERY″ にセットされているか、または *userexit* 構成パラメーターが使用可能になっている) データベースで破損回復を実行する場合に、個別の表スペースが原因で破損回復時にエラーが発生すると、その表スペースはオフラインにならなければならず、修復されるまでアクセスできなくなります。破損回復は続行します。破損回復が完了した時点で、データベースに入っている他の表スペースは使用可能であり、データベースへの接続は確立できます。 (一時表またはシステム・カタログ表が入っている表スペースに関しては例外があります。それらの表については、ロールフォワード回復の部分で説明します。)

前述のように、 DB2 には損傷したデータベースを回復させる方法が 2 つあります。

- *バージョン回復* は、以前のバージョンのデータベースの回復であり、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。

  データベース復元操作では、以前に作成されたデータベースのバックアップを使用して、データベース全体が再作成されます。データベースのバックアップにより、データベースを、バックアップをとった時点と同じ状態に復元することができます。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ています (30ページの図14を参照)。

  バージョン回復方式を使用して、データベースの完全なバックアップの計画を立てて、定期的に実行してください。

  区分データベース環境では、データベースは複数のデータベース区画サーバー (またはノード) にまたがって存在します。すべての区画を復元することと、データベース復元操作に使用するすべてのバックアップ・イメージを同時に作成することが必要です。 (各データベース区画は、別々にバックアップされ、復元されます。) 同時に作成される各データベース区画のバックアップは、バージョン・バックアップ と呼ばれます。

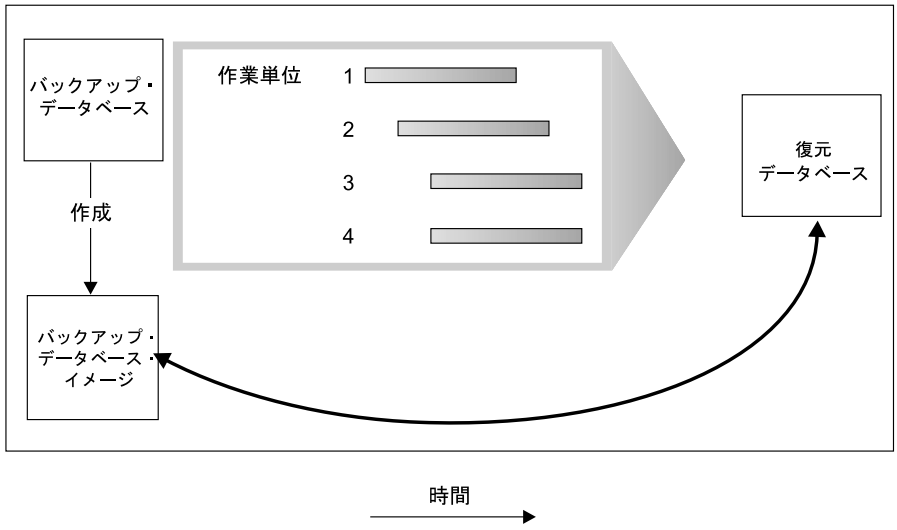


図 14. データベースの復元

- *ロールフォワード回復* 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おき、ログをアーカイブする必要があります (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 (またはその両方) を使用可能にすることで実行できます。使用するロギング手順について詳しくは、35ページの『データベース・ログ』を参照してください。) データベースを復元して、 WITHOUT ROLLING FORWARD オプションを指定することは、バージョン回復方式を使用することと同じです。データベースはオフライン・バックアップ・イメージをとった時点と同一の状態に復元されます。データベースを復元する際、データベース復元操作で WITHOUT ROLLING FORWARD オプションを指定していない 場合、そのデータベースは復元操作が終了するまでロールフォワード保留状態になります。これで、ロールフォワード回復を行えるようになります。

  考慮する 2 つのロールフォワード回復は次のとおりです。

  – *データベースのロールフォワード回復*。このタイプのロールフォワード回復では、データベース・ログに記録されているトランザクションがデータベース復元操作の後に適用されます (31ページの図15 を参照)。データベース・ログには、データベースへの変更がすべて記録されています。この方式では、データベースが特定の時点の状態に、または障害が生じる直前 (アクティブ・ログの最後) の状態に回復されます。

    区分データベース環境では、データベースは多数のデータベース区画にまたがって存在します。時刻指定ロールフォワード回復を実行する場合は、すべてのデータベース区画をロールフォワードし、すべての区画が同じレ

ベルになるようにする必要があります。単一のデータベース区画を復元しなければならない場合は、ログの最後までロールフォワード回復を実行して、その区画をデータベース内の他の区画と同じレベルにすることができます。

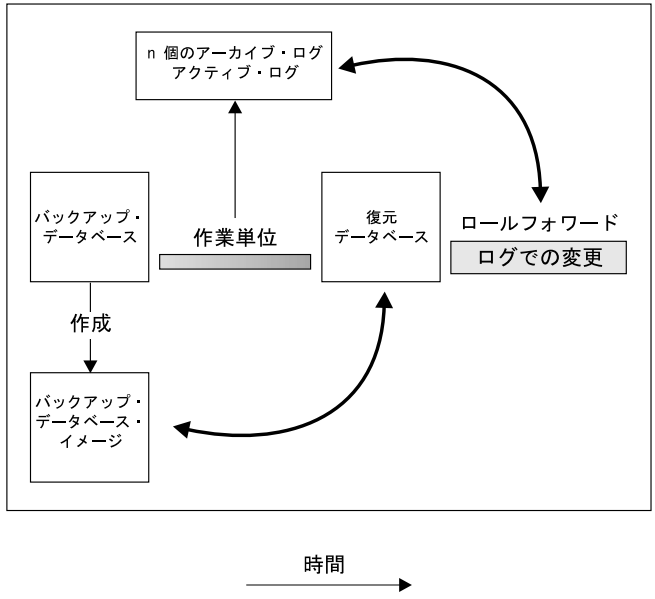


図 15. データベースのロールフォワード回復

– *表スペースの復元とロールフォワード.* データベースの順方向回復が可能であれば、表スペースのバックアップ、復元、およびロールフォワードも可能です。表スペースの復元およびロールフォワードを行う場合、データベース全体 (つまり、すべての表スペース)、または 1 つまたは複数の個別表スペースのバックアップが必要です。さらに、回復する表スペースに影響を与えるログ・レコードが必要です。以下の 2 つのポイントのうちの 1 つにログをロールフォワードできます。

  - ログの終わった時点。または、
  - 特定の時点 (*時刻指定* 回復と呼ばれる)。

  注:

  1. バックアップ操作時に選択されなかった表スペースは、復元された表スペースと同じ状態にはなりません。

2. 表スペースに対してロールフォワード回復方式を使用するとき、回復するデータベースの「キー」表スペースを識別し、このデータベースまたは「キー」表スペースのバックアップのスケジュールおよび実行を定期的に行う必要があります。

表スペース・ロールフォワード回復は、以下の 2 つの場合に使用されます。

– 表スペース復元操作の後、表スペースは常にロールフォワード保留状態で、ロールフォワードする必要があります。 ROLLFORWARD DATABASE コマンド (*コマンド解説書* を参照) を呼び出し、特定の時点またはログの最後まで表スペースにログを適用してください。
– 破損回復の後で 1 つまたは複数の表スペースが *ロールフォワード保留* 状態の場合、まず表スペースの問題を訂正します。場合によっては、表スペースの問題を訂正するのにデータベース復元操作の実行が関係しない場合もあります。たとえば、電源が切れると、表スペースはロールフォワード保留状態になります。破損回復の前にこの問題が訂正された場合、データベースを整合性のある使用可能状態にするのに破損回復で十分です。この場合にはデータベース復元操作は必要ありません。表スペースの問題が解決したら、ROLLFORWARD DATABASE コマンドを使用して、特定の時点かログの最後までのログを、表スペースに適用することができます。

  注: エラーが発生した表スペースにシステム・カタログ表が含まれている場合は、データベースを開始することはできません。SYSCATSPACE 表スペースを復元した後、ログの終わりまでロールフォワード回復を実行する必要があります。

区分データベース環境では、表スペースを *特定の時点まで* ロールフォワードする場合、表スペースが常駐するノード (データベース区画) のリストを指定する必要はありません。 DB2 は、すべての区画にロールフォワード要求を実行依頼します。これは、表スペースが常駐するすべてのデータベース区画で表スペースを復元しなければならないことを意味します。

区分データベース・システムでは、*ログの最後に向けて* 表スペースをロールフォワードする場合で、すべての区画で表スペースをロールフォワードしたくない 場合は、データベース区画のリストを提供する必要があります。すべての区画上にある、ロールフォワード保留状態のすべての表スペースをログの最後までロールフォワードしたい場合には、データベース区画のリストを指定する必要はありません。省略時解釈では、ロールフォワード要求はすべての区画に送信されます。

回復に影響する要素

どのデータベース回復方式を使用するかを決定するには、以下の基本的な要素を考慮してください。

- データベースは回復可能か、それとも回復不能か。
- 障害発生時のどれほど近いところまでデータベースを回復する必要があるか (回復点)。
- データベース回復にどれ位時間がかかるか。これには、以下の要因が関係しています。
  - バックアップ間の間隔 (ロールフォワード回復に影響します)
  - データベースが使用可能な時間、またはデータベースにアクセス可能な時間 (バックアップをオンラインで行うかオフラインで行うかは、データの可用性の必要性により決まります)
- バックアップ・コピーおよびアーカイブ・ログのために割り振ることができる記憶スペース量
- 表スペースのレベルでのバックアップか、データベース全体のレベルのバックアップか

通常は、データベース保守および回復方針では、データベース回復のために必要になった時点ですべての情報を使用できるようにしておく必要があります。この方針には、データベース・バックアップをとるための定期スケジュールを組み込み、さらにデータベース作成時、または区分データベース・システムの場合はデータベース区画サーバー (ノード) の追加あるいは削除によるシステム規模の変更時の予定バックアップ作成操作を組み込む必要があります。これらの基本要件以外にも、データベース障害の発生を抑制したりその影響を緩和する要素を組み込んでおくと、方針はさらに完全なものになります。

以下の部分で、さらに詳しく説明します。

- 34ページの『回復可能データベースおよび回復不能データベース』
- 35ページの『データベース・ログ』
- 40ページの『作業表におけるロギングの低減』
- 41ページの『回復目標点』
- 42ページの『バックアップの頻度と所要時間』
- 44ページの『必要な回復時間』
- 44ページの『記憶域についての考慮事項』
- 45ページの『関連データの一括保持』

- 46ページの『異なるオペレーティング・システムの使用についての制約事項』
- 46ページの『損傷を受けた表スペースの回復』
- 48ページの『回復のパフォーマンスに関する考慮事項』

このセクションでは主にデータベースについて述べていますが、全体的な回復計画には次の項目の回復も組み込む必要があります。

- オペレーティング・システムおよび DB2 実行可能ファイル
- アプリケーション、UDF、およびオペレーティング・システム・ライブラリーに含まれるストアード・プロシージャー・コード
- DB2 インスタンスおよび非 DB2 リソースを作成するためのコマンド
- オペレーティング・システムの機密保護
- ロード操作からのロード・コピー (LOAD コマンドに COPY YES を指定した場合)

回復可能データベースおよび回復不能データベース

データを容易に再作成できる場合、そのデータが含まれているデータベースは回復不能データベースにすることができます。次に例を示します。

- 読み取り専用アプリケーションに使用される外部ソースからのデータ (および、既存のデータと混在していないデータ) が含まれる表は、回復不能データベースに入れることを考慮します。
- データ量の少ない表。この場合、回復が問題なのではありません。むしろ、データのロギングを十分に行っていないため、ログ・ファイルの管理および復元後のロールフォワードの複雑さに対応できないということです。
- 少量の行が定期的に追加される大きな表。この場合も、揮発性が低いため、ログ・ファイルの管理および復元操作後のロールフォワードに十分対応できません。

データの再作成が困難な場合、そのデータを保持するデータベースは回復可能データベースでなければなりません。回復可能データベースの一部になる必要のあるデータの例を以下に挙げます。

- 再作成できないデータ。これには、ロード後にソースが破棄されるデータや、表の中に手操作で入力するデータが含まれます。
- データベースにロードした後に、アプリケーション・プログラムまたはワークステーション・ユーザーによって修正されるデータ。

データベース・ログ

すべてのデータベースには、それに伴うログがあります。これらのログには、データベース変更の記録が保持されています。データベースを最後の全オフライン・バックアップよりも後の時点に復元する必要がある場合は、データを障害発生時点までロールフォーワードするためにログが必要です。

DB2 ログには 2 つのタイプがあります。*循環* とアーカイブ であり、それぞれは、異なるレベルの回復機能を提供します。

循環 ロギングは、新規のデータベースが作成されるときの省略時動作です。このタイプのロギングでは、データベースの全オフライン・バックアップのみが有効です。名前から分かるとおり、循環ログはオンライン・ログの「輪」を使用して、トランザクション障害およびシステム破損からの回復を提供します。ログは、現行トランザクションの保全性を保証するためにのみ使用および保存されます。循環ロギングでは、最後に行った全バックアップより後のトランザクションを使用してデータベースをロールフォワードすることはできません。媒体障害および災害からの回復は、全オフライン・バックアップからの復元によって行います。最後のバックアップ以降の変更はすべて失われます。全バックアップを作成するとき、データベースはオフライン (ユーザーからアクセス不能) でなければなりません。このタイプの復元では、データが全バックアップの時点まで回復されるため、これは、*バージョン回復* と呼ばれています。

36ページの図16 は、循環ロギングが活動状態のときにアクティブ・ログがログ・ファイルの輪を使用する様子を示しています。

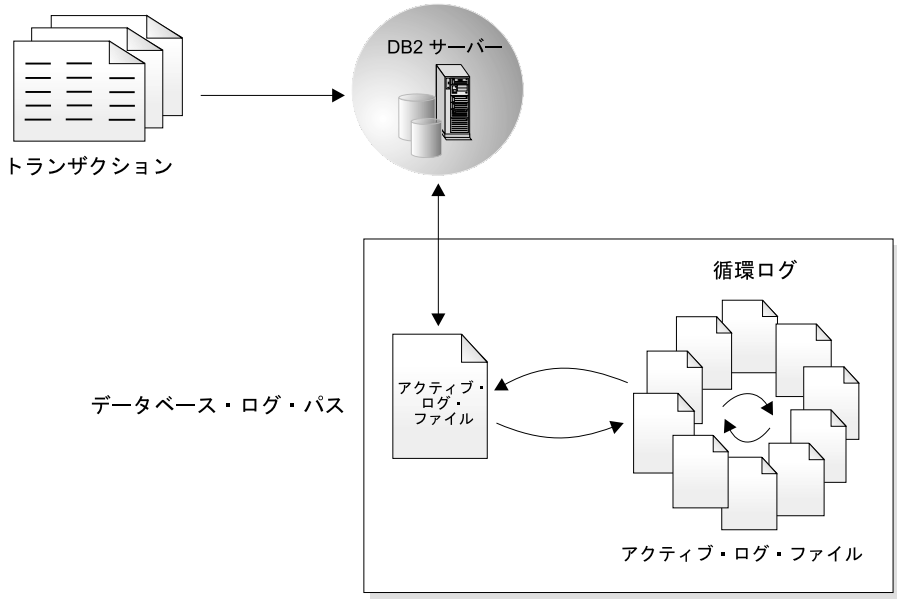


図 16. 循環ロギング

アクティブ・ログは、障害 (システム電源またはアプリケーション・エラー) が原因でデータベースが一貫性のない状態になるのを防ぐために、破損回復処理中に使用されます。 RESTART DATABASE コマンドは必要に応じてアクティブ・ログを使用し、データベースを、一貫性のある使用可能な状態にします。破損回復の際、障害が原因でコミットされていないログ内の変更内容は、ロールバックされます。コミットされてはいるもののメモリー (バッファー・プール) からディスク (データベース・コンテナー) へ物理的にまだ書き出されていない変更は、再実行されます。こうした処置によって、データベースの保全性が保たれます。また、特定の時点での回復またはログの最後までの回復を実行するときに、 ROLLFORWARD DATABASE コマンドもアクティブ・ログを使用します。アクティブ・ログは、データベース・ログ・パス・ディレクトリーにあります。

アーカイブ・ログがロールフォワード回復に使用されます。次のとおりです。

オンライン・アーカイブ・ログ

アクティブ・ログに入っている変更内容が通常の処理で必要なくなると、そのログはクローズされて、アーカイブ・ログになります。アーカイブ・ログがデータベース・ログ・パス・ディレクトリーに入っている場合、そのログはオンライン であるといいます (37ページの図17 を参照)。

オフライン・アーカイブ・ログ

アーカイブ・ログがデータベース・ログ・パス・ディレクトリーに入っていない場合、そのログはオフライン であるといいます (38ページの図18を参照)。ユーザー出口プログラムを使用して、アーカイブ・ログをデータベース・ログ・パス・ディレクトリー以外の位置に保管することもできます。 (追加情報については、 *管理の手引き: インプリメンテーション* の『データベース回復用のユーザー出口』を参照してください。)

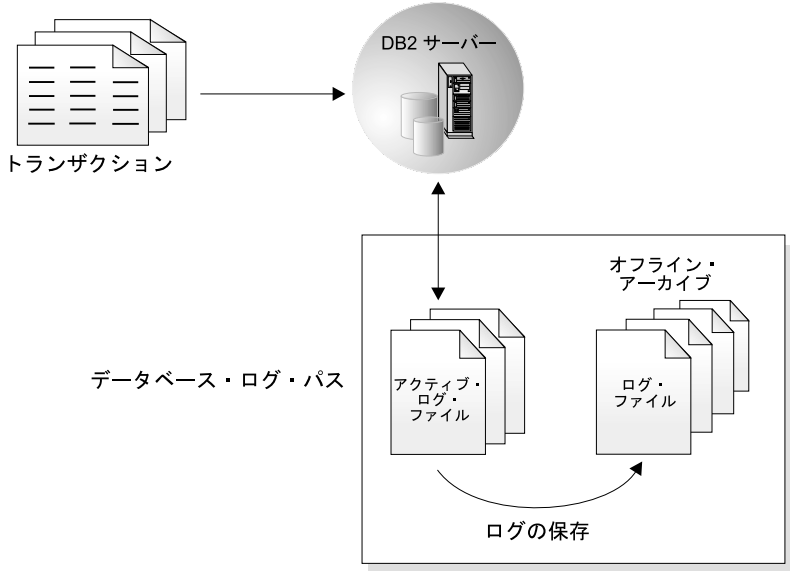


図 17. アーカイブ・ログ

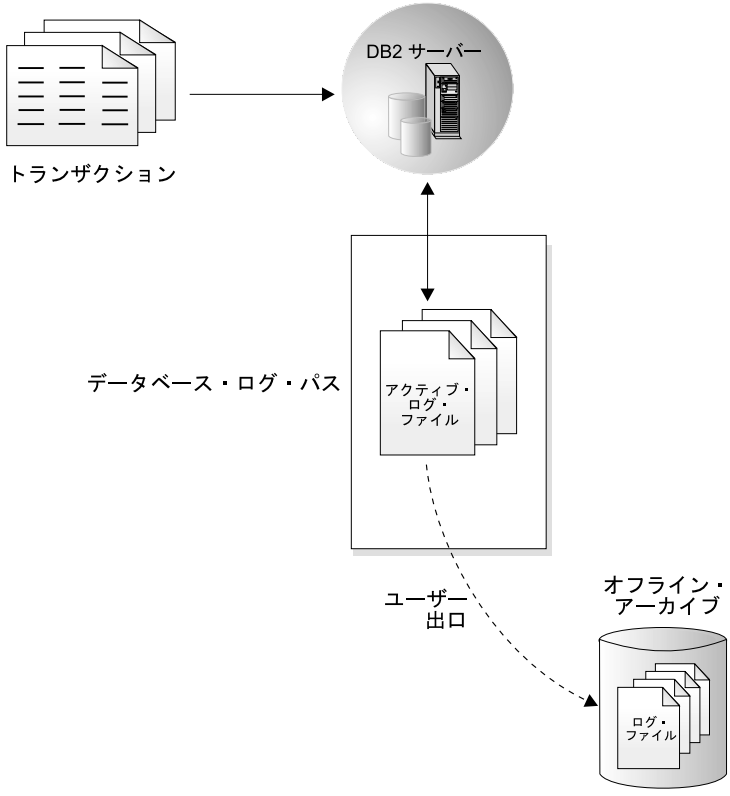


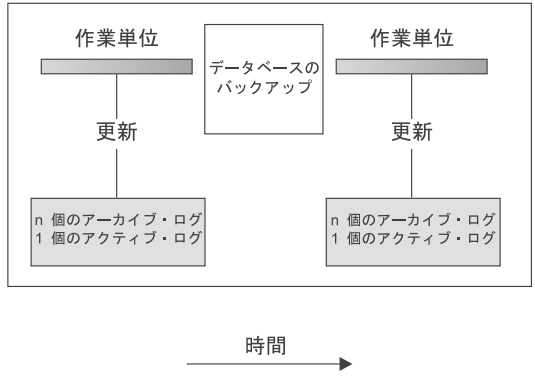
図 18. オフライン・アーカイブ・ログ

ロールフォワード回復では、アーカイブ・ログとアクティブ・ログの両方を使用して、ログの終わりまでまたは特定の時点まで、データベースを再作成することができます。ロールフォワード機能は、アーカイブ・ログとアクティブ・ログで検出されるコミットされた変更項目を、復元されるデータベースに再適用することで、この処理を実行します。

ロールフォワード回復は、アーカイブ・ログとアクティブ・ログの両方にあるコミットされた更新を再適用することにより、ログを使用して表スペースを再作成することもできます。表スペースは、ログの最後まで、または特定の時点まで回復可能です。

オンライン・バックアップ時には、データベースに対するすべての活動がログに記録されます。オンライン・バックアップが復元されるとき、これらのログは少なくともバックアップが完了した時点までロールフォワードされなければなりません。このことが生じるようにするために、ログをアーカイブしておき、データベースが復元されるときに使用可能にしなければなりません。操作

が完了したかなり後でも、バックアップ時に使用されたログ・ファイルがオープン状態のままになっている場合があります。 BACKUP DATABASE コマンドでオンライン・バックアップの FLUSH LOG オプションを指定すれば、オンライン・バックアップの完了時にアクティブ・ログがクローズされます。これにより、アクティブ・ログをアーカイブして、完全なバックアップと、そのバックアップの復元に必要なログすべてが備えられるようにできます。



データベースへの変更を確認するためにバックアップ間でログが使用される

図 19. ロールフォワード回復での活動およびアーカイブ・データベース・ログ

アーカイブ・ログの記憶位置は、*newlogpath* と *userexit* の 2 つのデータベース構成パラメーターによって変更できます。 *newlogpath* パラメーターを変更すると、アクティブ・ログの記憶位置も変更されます。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。

データベース・ログ・パス・ディレクトリーの中のどのログ・エクステント (18ページの『コンテナー』を参照) がアーカイブ・ログかを判別するには、*loghead* データベース構成パラメーターの値を調べてください。このパラメーターには、最も低い番号のアクティブ・ログが示されています。 *loghead* よりも小さい順序番号のログはアーカイブ・ログであり、それらは移動可能なログです。このパラメーターの値は、コントロール・センターを使用して検査できます。あるいは、コマンド行プロセッサーおよび GET DATABASE CONFIGURATION コマンドを使用して、「最初のアクティブ・ログ・ファイル」を参照します。こ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。

注:

1. アクティブ・ログを消去すると、データベースは使用不能になってしまい、再び使用できるようにするには復元することが必要になります。また、消去された最初のログまでしかロールフォワードできません。
2. アクティブ・ログが (ディスクが壊れた結果) 損傷するかもしれないことが心配な場合は、ログの保管先のボリュームをミラーリングすることを考慮する必要があります。

作業表におけるロギングの低減

アプリケーションでマスター表から作業表を作成してその中にデータを入れる場合に、マスター表から簡単に再作成できるという理由でこれら作業表の回復可能性について心配する必要がない場合、作業表は CREATE TABLE ステートメントに NOT LOGGED INITIALLY パラメーターを指定して作成することができます。 NOT LOGGED INITIALLY パラメーターを使用する利点は、表が作成された作業単位と同じ作業単位で表に対し行われた変更 (挿入、削除、更新、または索引作成操作を含む) をロギングしなくてもよい点です。これにより、実行されるロギングが低減されるだけでなく、アプリケーションのパフォーマンスも向上します。 NOT LOGGED INITIALLY パラメーターを指定した ALTER TABLE ステートメントを使用して、既存の表について同じ結果を確保できます。

注:

1. 同じ作業単位で NOT LOGGED INITIALLY パラメーターを使用し複数の表を作成できます。
2. カタログ表および他のユーザー表に対する変更は、ロギングの対象になります。

表に対する変更がロギングされないため、 NOT LOGGED INITIALLY パラメーターの使用を決定するときには次の点を考慮する必要があります。

- 表に対するすべての 変更項目は、コミット時に書き出されます。つまり、コミットには時間がかかります。
- 表が作成される作業単位の操作についてエラーが戻されると、作業単位全体がロールバックされます (SQLCODE -1476、SQLSTATE 40506)。
- ロールフォワード時には、これらの表を回復できません。ロールフォワード操作実行時に NOT LOGGED INITIALLY オプションにより作成された表が検出されると、この表には使用不能のマークが付けられます。データベースの回復後にこの表にアクセスしようとすると、 SQL1477N が戻されます。

注: 表が作成されると、COMMIT が実行されるまでカタログ表には行ロックが保持されます。ロギングが行われないことを利用するためには、作成される作業単位と同じ作業単位の表にデータを入れる必要があります。これは、並行操作を意味します。詳細については、*管理の手引き: パフォーマンス* の『並行性』を参照してください。

表の作成についての詳細は、*SQL 解説書* を参照してください。

宣言済み一時表を作業表として使用することを計画している場合は、次の点に注意してください。

- 宣言済み一時表はカタログでは作成されません。したがって、ロックは保留されません。
- 宣言済み一時表に対しては、最初の COMMIT の後でもロギングは実行されません。
- COMMIT の後に表の行を保持するため、ON COMMIT PRESERVE オプションを使用してください。これを行わないと、すべての行が削除されます。
- 宣言済み一時表を作成したアプリケーションだけが、その表のインスタンスにアクセスできます。
- データベースのアプリケーション接続が除去されるときに、宣言済み一時表は暗黙的に除去されます。
- 宣言済み一時表を使用する作業単位の途中で操作にエラーが生じると、作業単位が完全にはロールバックされなくなります。ただし、宣言済み一時表の内容を変更するステートメントで操作にエラーが生じると、表の行がすべて削除されます。作業単位 (または保管ポイント) のロールバックでは、その作業単位 (または保管ポイント) で変更された宣言済み一時表にあるすべての行が削除されます。

宣言済み一時表とその制限についての詳細は、 *SQL 解説書* で DECLARE GLOBAL TEMPORARY TABLE ステートメントを参照してください。

回復目標点

バージョン回復とロールフォワード回復とでは、回復の目標時点が異なります。バージョン回復の場合には、計画された各時点で、オフラインの完全なデータベース・バックアップ・コピーをとります。この方法では、回復されたデータベースは、復元されたバックアップ・コピーより最新にはなりません。たとえば、一日の終わりにバックアップ・コピーをとり、翌日に作業中途でデータベースを失った場合は、半日分の変更内容を失うことになります。

ロールフォワード回復方式の場合、データベースに対する変更はログの中に記録されています。この方法では、まずバックアップ・コピーを使ってデータベ

ースまたは表スペースを復元した後、ログを使用して、バックアップ・コピーが作成されて以来のデータベース変更を適用し直します。

ロールフォワード回復を使えるようにすると、オンライン・バックアップと表スペース・レベルのバックアップの両方の長所を活用できます。データベースまたは表スペース全体のロールフォワード回復を行うためには、ログの最後まで、または指定された時点までの回復を選択できます。たとえば、データベースがアプリケーションによって破壊された場合は、復元されたデータベース・コピーから開始して、アプリケーション開始直前までの変更をロールフォワードすることができます。指定した時間以降に書き込まれた作業単位は再適用されません。

また、ログの最後まで、または特定の時点まで、表スペースをロールフォワードできます。

バックアップの頻度と所要時間

データベースのバックアップには時間もシステム・リソースも必要となるため、回復計画では、定期的なバックアップも含める必要があります。

ログを保管する場合など (これにより、ロールフォワード回復が可能になる)、データベース全体のバックアップを定期的にとるようにしてください。回復方針にロールフォワード回復が組み込まれている場合は、最近のデータベース全体のバックアップを作成しておくと、データベースに適用しなければならないアーカイブ・ログが少なくて済むことになります。これにより、データベースを回復するために実行する ROLLFORWARD ユーティリティーの時間が短縮されます。

また、バックアップおよびログを上書きせずに、安全を考慮してデータベース全体および関連ログを複数個保管することも考慮してください。

頻繁に使用されるデータベースの回復とロールフォワードでアーカイブ・ログを適用するのに必要な時間について心配な場合には、頻繁にデータベースのバックアップをとるためのコストを考慮してください。これにより、ロールフォワード時に適用する必要のあるアーカイブ・ログの数を減らすことができます。

バックアップは、データベースがオンライン でもオフライン でも実行できます。オンラインの場合、他のアプリケーションまたはプロセスは、バックアップ操作の実行中もデータベースに接続し続けたり、データの読み取りや修正を行うことができます。バックアップがオンラインで実行される場合、データベ

ースに接続できるのはバックアップ操作だけです。バックアップ・タスクの実行中は組織内の他の部署はデータベースに接続できません。

データベースが使用できなくなる時間を短くするには、オンライン・バックアップの使用が考えられます。ロールフォワード回復が可能である場合にのみ、オンライン・バックアップがサポートされます。ロールフォワード回復を使用でき、ログの完全セットがある場合は、必要が生じたときにデータベースを再作成できます。

注:

1. バックアップ操作が行われた時間に関係しているデータベース・ログがある場合、使用できるのはオンライン・バックアップのみです。
2. オフライン・バックアップは、オンライン・バックアップより高速です。

データベースの中に長形式フィールドと LOB のデータが大量に含まれている場合には、データベースのバックアップ作業に非常に多くの時間がかかります。 BACKUP コマンドでは、特定の表スペースのバックアップをとることができます。 DMS 表スペースを使用すると、その表スペースに異なったタイプのデータを格納でき、バックアップ操作に必要な時間を短縮できます。表データをある表スペースに保持し、長形式フィールドおよび LOB データを別の表スペースに保持し、索引を別の表スペースに保持することができます。長形式フィールドと LOB データを別個の表スペースに保管してあるなら、長形式フィールドと LOB データが入っている表スペースのバックアップはとらないように選択することによって、バックアップの完了にかかる時間を少なくすることができます。長形式フィールドおよび LOB データがビジネスにとって重要である場合には、これらの表スペースのバックアップの作成と、これらの表スペースの復元操作の完了にかかる時間との関係を考慮してください。 LOB データが別のソースから再作成可能である場合は、 LOB 列を含む表の作成または変更時には、NOT LOGGED オプションを選択してください。

表を再編成する場合、操作完了後に関連した表スペースのバックアップを作成する必要があります。これにより、表スペースを復元しなければならない場合でも、データ再編成によりロールフォワードを実行しなくても済みます。

注: 表データの一部が入っていない表スペースをバックアップする場合、その表スペースに対して時刻指定ロールフォワード回復は実行できません。表に関する任意のデータ・タイプが含まれているすべての表スペースは、同じ時点まで同時にロールフォワードする必要があります。

必要な回復時間

データベースの回復に必要な時間は、次の 2 つの要素で構成されます。つまり、バックアップの復元を完了するために必要な時間と、ロールフォワード操作時にログを適用するために必要な時間 (これは、データベースが順方向回復について使用可能である場合) です。回復計画を公式化するときは、これらの回復費用とそれらが業務操作に与える影響を考慮しなければなりません。全般的な回復計画をテストすることで、データベースを回復するために必要な時間が、業務上の要件を考慮して正当かどうかを判別することができます。各テストを実施した後、バックアップを作成する頻度を増やすことができます。回復方針の一部としてロールフォワード回復を実行する場合は、これによりバックアップ間でアーカイブされるログ数が減少し、その結果、復元操作後にデータベースをロールフォワードするための時間が短縮されます。

注: 区画内の並行処理 (*intra_parallel*) を使用可能に設定すると、データベース・マネージャー構成パラメーターはバックアップと復元操作のパフォーマンスに影響を与えません。 *intra_parallel* パラメーターの設定に関係なく、それぞれの操作で複数のプロセスが使用されます。

記憶域についての考慮事項

どの回復方式を使用するかを決定する際には、記憶スペースの要件を考慮する必要があります。

バージョン回復方式の場合は、データベースと復元後のデータベースを入れるスペースが必要になります。ロールフォワード回復方式の場合は、データベースまたは表スペースのバックアップ・コピー、復元後のデータベース、およびアーカイブ・データベース・ログを入れるだけのスペースが必要になります。

表の中に長形式フィールドとラージ・オブジェクト (LOB) の列が含まれている場合は、そのデータを別の表スペースに入れることを考慮してください。このことは、回復の計画に関係するだけでなく、記憶スペースにも影響します。長形式フィールドと LOB データを別の表スペースに取り分けておき、長形式フィールドと LOB データのバックアップにかかる時間が分かっているなら、表スペースのバックアップ頻度を少なくした回復計画にするよう決定できるかもしれません。表を作成または更新して LOB 列を組み込む際に、これらの列に対する変更内容を記録しないことを選択することもできます。このように選択すると、必要なログ・スペースおよび対応するログ・アーカイブ・スペースのサイズを減らせます。

LOB が入っている SMS 表スペースのバックアップは、元の表スペースのサイズより大きくなっている可能性があります。表スペースに含まれる LOB データ・サイズによっては、バックアップは最大 40 パーセント大きくなる可能性

があります。たとえば、1 GB の SMS 表スペース (LOB が入っている) のバックアップを取る場合には、復元する際に 1 GB より大きいディスク容量が必要になります。このことが起こるのは、予備割り振りをサポートするファイル・システム (たとえば、 UNIX ベースのオペレーティング・システム) だけです。

媒体障害が発生したときにデータベースが破壊され、それを再作成できなくなってしまうことがないようにするため、データベース・バックアップ、データベース・ログ、およびデータベース自身を別々の装置に保持するようにしてください。この理由で、データベース作成時に、データベース・ログは、*newlogpath* 構成パラメーターを使うことによって、必ず別の装置に保管しておくようにしてください。 (このパラメーターも含め、ロギング関連の構成パラメーターについては、 *管理の手引き: インプリメンテーション* の『データベースでの変更のロールフォワード』で説明します。)

データベース・ログには、大量の記憶域を使用できます。ロールフォワード回復方法を使用する場合は、アーカイブ・ログをどのように管理するかを決定する必要があります。選択肢は次のとおりです。

- データベース・ログ・パス・ディレクトリーの中に、ログを入れるためのスペースを十分にとる。
- すでにアクティブ・セットではなくなったログを、データベース・ログ・パス・ディレクトリー以外の記憶装置またはディレクトリーに手動でコピーする。
- ユーザー出口プログラムを使用して、これらのログを別の記憶装置にコピーする。 (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『データベース回復用のユーザー出口』を参照してください。)

注: OS/2 の場合は、標準および非標準装置に存在するデータベースおよびデータベース・ログの両方のデータベース・バックアップ・イメージの記憶域を処理するためのユーザー出口プログラムが、 DB2 でサポートされています。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『データベース回復用のユーザー出口』を参照してください。

関連データの一括保持

データベースを設計するとき、各表間の関係が分かります。これらの関係はアプリケーション・レベルで表現することができ、この場合は、トランザクションは複数の表を更新します。またデータベース・レベルで表現することができ、この場合は、表間に参照保全が存在するかまたはある表のトリガーが別の表に影響を与えます。回復計画を作成するときは、これらの関係を考慮に入れる必要があります。関連するデータ・セットは、まとめてバックアップできま

す。そのようなセットは、表スペース・レベルまたはデータベース・レベルのいずれかで作成できます。関連するデータのセットをまとめて保持することで、すべてのデータの一貫性が保証されている時点まで、回復処理を実行できます。これは、表スペースに対し特定の時点のロールフォワード回復を実行できるようにしたい場合、特に重要です。

異なるオペレーティング・システムの使用についての制約事項
複数のオペレーティング・システムのある環境で作業する場合、バックアップおよび回復の計画を統合できないことを考慮しなければなりません。つまり、一方のオペレーティング・システムで BACKUP DATABASE コマンドを使用し、別のオペレーティング・システムで RESTORE DATABASE コマンドを使用することはできません。それぞれのオペレーティング・システムの回復計画は、別個に独立して保持しなければなりません。

一方のオペレーティング・システムから他のオペレーティング・システムに表を移動しなければならない場合には、**db2move** コマンドを使用します。あるいは、IMPORT または LOAD コマンドと共に EXPORT を使用します。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。

損傷を受けた表スペースの回復
損傷を受けた表スペースには、アクセスできない 1 つまたは複数のコンテナーがあります。これは、永続的な媒体 (たとえば、ディスクに障害がある) または一時的な媒体 (ディスクがオフラインになっている、またはファイル・システムがマウントされていない) の問題が原因です。

損傷を受けた表スペースがシステム・カタログ表スペースの場合には、データベースを再始動することはできません。元のデータをそのままの状態にしてコンテナーの問題を修正できない場合には、以下の選択肢しか実行できません。

- データベースを復元する、または
- カタログ表スペースを復元する。

注: データベースをロールフォワードしなければならないので、表スペースの復元は回復可能なデータベースについてのみ有効です。

損傷を受けた表スペースがシステム・カタログ表スペースでない場合には、DB2 はできるかぎりデータベースを使用できるようにします。この場合に成功するかどうかはロギング戦略によって決まります。

損傷を受けた表スペースが唯一の一時表スペースである場合には、データベースに接続されたらすぐに新しい一時表スペースを作成しなければなりません。

一度作成されると、新しい一時表スペースが使用できるようになり、一時表スペースが必要な通常のデータベース操作を再開できます。それから、任意でオフライン一時表スペースを除去します。システム一時表スペースを使用する表の再編成については、特別な考慮事項があります。

- データベースまたはデータベース・マネージャー構成パラメーター *indexrec* が「RESTART」に設定されている場合、すべての無効な索引はデータベースの活動化中に再作成されなければなりません。これには、組み立てフェーズで破損した再編成からの索引も含まれます。
- 損傷を受けた一時表スペースで完了していない再編成の要求がある場合には、 *indexrec* 構成パラメーターを「ACCESS」に設定して、再始動の障害を避ける必要があります。

回復可能データベースの表スペース回復: 破損回復が必要であるため、損傷を受けた表スペースは、アクセス不能状態でオフラインに入れられ、さらにロールフォワード保留状態になります。他に問題がなければ再始動操作は成功します。損傷を受けた表スペースは、次の場合に再び使用できます。

- 元のデータを失うことなく損傷のあるコンテナーを修正し、それから表スペースのロールフォワード操作を完了した。 (ロールフォワード操作は、最初に表スペースをオフラインから通常の状態にする操作を行います。)
- 損傷のあるコンテナーを修正した (元のデータが失われる場合も失われない場合もある) 後に、表スペースの復元を実行し、それからロールフォワード操作を実行した。

回復不能データベースの表スペース回復: 破損回復は必要であり、ログは永久に保持されるわけではないため、ユーザーが損傷を受けた表スペースを除去するのをいとわない場合にのみ、再始動操作を成功させることができます。 (正常な回復とは、損傷を受けた表スペースを回復して整合性のある状態に戻す必要のあるログ・レコードがなくなることを意味します。したがって、そのような表スペースで有効なアクションは、これらを除去することだけです。)

これを行うには、修飾されていないデータベース再始動操作を呼び出します。損傷を受けた表スペースがない場合には、これは成功します。失敗した (SQL0290N) 場合には、 `db2diag.log` を調べて、現在損傷を受けている表スペースの完全なリストを参照できます。

- データベースの再始動操作が完了したときにこれらの表をすべて除去したい場合は、別のデータベース再始動操作を開始し、 DROP PENDING TABLESPACES オプションを使用して損傷を受けたすべての表スペースをリストできます。損傷を受けた表スペースが DROP PENDING TABLESPACES リストにある場合、表スペースは除去保留状態に入れられているため、回復

後の唯一のオプションは、表スペースの除去になります。再始動操作は、この表スペースを回復することなく継続されます。損傷を受けた表スペースが DROP PENDING TABLESPACES リストにない 場合、データベース再始動の操作は SQL0290N を出して失敗します。

- これらの表スペースを除去したくない (そこにあるデータを失いたくない) 場合は、以下のいずれかを実行します。
  - 待機して、損傷を受けているコンテナーを修正し (元のデータを失わないようにする)、それからデータベース再始動の操作を行います。
  - データベースの復元操作を実行します。

注: DROP PENDING TABLESPACES リストに表スペース名を入れても、この表スペースが DROP PENDING 状態になったことにはなりません。これは、表スペースが再始動操作中に損傷を受けた場合にのみ起こることです。再始動操作が成功したら、 TABLESPACE ステートメントを出して除去保留状態にある各表スペースを除去する必要があります (LIST TABLESPACES コマンドを使用して、どの表スペースがこの状態であるかを調べてください)。このようにして、スペースを再利用するか、または表スペースを再作成できます。

回復のパフォーマンスに関する考慮事項

回復のパフォーマンスについて考慮する際には、以下の点についても考慮に含めてください。

- ログを別の装置に置くことによって、頻繁に更新されるデータベースのパフォーマンスを改善できます。トランザクション処理 (OLTP) 環境では、多くの場合、ログにデータを書き込むには、データ行を保管するよりも入出力が多くなります。ログを別の装置に入れるなら、データベース・ファイルとログとの間で移動するのに必要なディスク・アーム移動量を最小限にとどめることができます。

  他のどのファイルをそのディスクに入れるかについても考慮する必要があります。たとえば、実メモリーが不足しているシステムで、システム・ページングのために使用しているディスクにログを移動すると、調整は台なしになってしまいます。
- 復元操作を完了するために必要な時間を短縮するためには、次の方法が有効です。
  - 復元バッファー・サイズを調整します。バッファー・サイズは、バックアップ操作時に使用されたバッファー・サイズの倍数でなければなりません。
  - バッファー数を増加させる。

複数のバッファーと入出力チャネルを使用する場合、少なくともチャネルの 2 倍のバッファーを使用し、チャネルがデータを待つ状態にならないようにします。使用するバッファーのサイズも、復元操作のパフォーマンスに影響を与えます。理想的な復元バッファーのサイズは、表スペースのエクステント・サイズの倍数です。

複数の表スペースがありそれぞれエクステント・サイズが異なる場合は、最大エクステント・サイズの倍数の値を指定します。

推奨する最小の バッファー数は、メディア装置またはコンテナーの数に、 PARALLELISM オプションに指定される数を加えたものです。

- 複数のソース装置を使用する。
- 復元操作の PARALLELISM オプションを、ソース装置の数より少なくとも 1 つ多くなるように設定する。

- 表に大量の長形式フィールド・データおよび LOB データが含まれている場合は、それらの復元には長時間かかります。データベースをロールフォワード回復可能にしておくと、 RESTORE コマンドでは特定の表スペースを復元できるようになります。長形式フィールド・データおよび LOB データが業務にとって重要な場合は、これらの表スペースを復元するときは、これらの表スペースのバックアップ・タスクを完了するために必要な時間を考慮する必要があります。したがって、長形式フィールドのデータおよび LOB データを別々の表スペースに保管させておけば、長形式フィールド・データおよび LOB データが含まれている表スペースの復元を選択しないことで、復元操作を完了するための時間を短縮させることができます。 LOB データが別のソースから再作成可能である場合は、 LOB 列を含む表の作成または変更時には、NOT LOGGED オプションを選択してください。長フィールドおよび LOB データを含む表スペースは復元しないが、この表が含まれている表スペースのロールフォワードを希望する場合は、ログの最後までロールフォワードを実行し、表データが含まれるすべての表スペース間で一貫性を保証するようにしてください。

  注: 表データが入っている表スペースのバックアップ時に、関連する LONG または LOB フィールドがないと、その表スペースの時刻指定ロールフォワード回復は実行できません。表に関するすべての表スペースは、同じ時点まで同時にロールフォワードする必要があります。
- バックアップ操作と復元操作には、以下の事柄が適用されます。
  - 複数の入出力バッファーと装置を使用してください。
  - 使用する装置の少なくとも 2 倍のバッファーを割り振ってください。
  - 入出力装置の制御装置の処理速度が過負荷にならないようにしてください。

- いくつかの大きなバッファーを使用する代わりに、より多くの小さなサイズのバッファーを使用してください。
- システム・リソースに従い、バッファー数とサイズを調整してください。

災害時回復の考慮事項

災害時回復 という用語は、火災、地震、暴力行為、または他の災害事象が発生した場合にデータベースを復元するために、実行しなければならない活動を記述するために使用されます。災害時回復の計画には、以下の 1 つまたは複数が含まれます。

- 非常事態発生時に使用されるサイト
- データベースを回復するための別のマシン
- データベース・バックアップおよびアーカイブ・ログのオフサイト記憶

災害時回復計画が別のマシンでのデータベース全体の回復を意味する場合は、少なくとも完全なデータベース・バックアップとデータベースに関するすべてのアーカイブ・ログが必要です。ログをアーカイブする際にそれらを予備のデータベースに適用することによって、そのデータベースを最新の状態にしておくことができます。あるいは、データベース・バックアップとログ・アーカイブを予備のサイトに保持し、災害発生後にだけ復元およびロールフォワードを実行するようにもできます。 (この場合、最新のデータベース・バックアップがぜひとも必要です。) しかし、災害時の場合は、すべてのトランザクションを災害発生時まで復元することは通常は不可能です。

災害時回復に表スペースのバックアップが役に立つかどうかは、障害の範囲によって決まります。通常、災害時回復ではデータベース全体を復元する必要があるので、待機サイトにデータベースの全バックアップを保持する必要があります。すべての表スペースの別々のバックアップ・イメージがあるとしても、データベースを復元するためにそれらを使用することはできません。その災害がディスクの損傷である場合には、表スペースのバックアップをそのディスクにある表スペースごとに回復に使用できます。ディスク障害 (または他の理由) によりコンテナーにアクセスできない場合は、コンテナーを別の場所に復元できます。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『RESTORE 実行時の表スペース・コンテナーの再定義』を参照してください。

表スペースのバックアップと完全なデータベースのバックアップの両方があれば、どのような災害時回復計画でも役に立ちます。バックアップ、復元、およ

びデータのロールフォワードのために用意されている DB2 機能は、災害時回復計画の基本となります。業務を保護するために、準備した回復手順をテストしておくようにしてください。

媒体障害の影響の緩和

媒体障害の発生率を緩和し、またこの障害タイプからの回復処理を簡単に実行できるようにするためには、次の操作を実行します。

- 重要なデータベースのデータおよびログが含まれているディスクについて、ミラー処理を実行するか複写を行う。
- 区分データベース環境では、カタログ・ノードのデータおよびログを操作するための厳密な手順を設定する。データベースを保守するのにこのノードが重要であるため、次のことを行ってください。
  - 必ず信頼できるディスクに常駐させる
  - 複製を作成する
  - バックアップを頻繁に取る
  - そこにユーザー・データは入れない

ディスク障害に対する保護

ディスクに障害が発生したためにデータまたはログが損傷を受ける危険性がある場合、考慮すべきことは、ディスク障害に対する何らかの許容度を持つ方策を講じておくことです。通常は、これは、*ディスク・アレイ* を使用することで行われます。ディスク・アレイはいくつかのディスク・ドライブをまとめたもので、アプリケーションからは 1 つの大きなディスクのように見えます。

ディスク・アレイには、*ディスク・ストライピング* (複数ディスク間でファイルを分散すること)、ディスクのミラーリング、およびデータ・パリティー検査が関係してきます。

ディスク・アレイは、単に RAID (Redundant Array of Independent Disks) と呼ばれることがあります。用語 RAID は、通常ハードウェア・ディスク・アレイにだけ適用されます。ただし、ディスク・アレイはオペレーティング・システムまたはアプリケーション・レベルのソフトウェアによっても提供されています。ハードウェア・ディスク・アレイとソフトウェア・ディスク・アレイの相違点は、入出力要求を CPU がどのように処理するかという点です。ハードウェア・ディスク・アレイの場合、ディスク制御装置が入出力活動を管理するのに対し、ソフトウェア・ディスク・アレイの場合は、オペレーティング・システムまたはアプリケーションにより実行されます。

ハードウェア・ディスク・アレイ (RAID): RAID ディスク・アレイでは、ディスク制御装置により複数のディスクが使用され管理されていて、独自の CPU も備えています。アレイを構成しているディスクの管理に必要なすべてのロジックはディスク制御装置に含まれています。したがって、この実行はオペレーティング・システムから独立して行われます。

RAID アーキテクチャーには RAID-1 から RAID-5 までの 5 種類があり、それぞれがディスク・フォールト・トレランスを提供しています。それぞれの機能とパフォーマンスは異なります。一般に、RAID は冗長アレイを指します。データ・ストライピングのみ (障害許容度の冗長度ではない) を提供する RAID-0 は、この説明からは除外されています。 RAID の仕様では 5 つのアーキテクチャーを定義していますが、今日では通常 RAID-1 と RAID-5 だけが使用されます。

RAID-1 は、ディスクのミラーリングまたはデュプレキシングとも呼ばれます。ディスク・ミラーリングは、単一のディスク制御装置を使用し、データ (完全なファイル) をあるディスクから別のディスクに複写します。ディスク・デュプレキシングはディスク・ミラーリングと同じですが、ディスクは 2 番目のディスク制御装置に接続されています (2 つの SCSI アダプターと同じ)。データの保護機能は良好です。つまり、どちらのディスクに障害が発生しても、データは他のディスクからアクセス可能です。デュプレキシングでは、データ保護の妥協がないと、ディスク制御装置に障害が発生することがあります。RAID-1 に関するパフォーマンスも良好ですが、この実現方法で考慮しなければならないトレードオフは、データをドライブのペアで複写しなければならないため、必要なディスク容量が実際のデータ量の 2 倍になることです。

RAID-5 は、すべてのディスクのセクター単位のデータ・ストライピングおよびパリティー・ストライピングに関係しています。パリティーは専用ドライブに保管される代わりに、データ情報とインターリーブされます。データの保護機能は良好です。ディスク障害が発生しても、他のディスクからの情報およびストライプされたパリティー情報を使用しアクセス可能です。書き込みパフォーマンスは RAID-1 または通常のディスクと比べてかなり低いですが、読み取りパフォーマンスは良好です。 RAID-5 構成では、少なくとも 3 つの同一なディスクが必要です。オーバーヘッドのために必要な余分なディスク・スペースは、アレイに含まれるディスク数により異なります。 5 つのディスクで構成される RAID-5 構成の場合は、スペース・オーバーヘッドは 20% です。

RAID ディスク・アレイ (RAID-0 ではない) を使用する場合、ディスクに障害が発生してもユーザーはアレイ上のデータにアクセスできます。常時交換可能または常時スワップ可能ディスクをアレイに使用すると、アレイ使用中に交換ディスクを障害ディスクとスワップすることが可能です。 RAID-5 の場合、2

つのディスクで同時に障害が発生すると、すべてのデータは失われます (しかし、同様のディスク障害が発生する可能性はごくまれです)。

RAID-1 またはソフトウェア・ミラー・ディスクをログに使用できます (『ソフトウェア・ディスク・アレイ』を参照)。これは、障害点までの回復可能性があり、書き込みパフォーマンスが高いためで、これはログにとって重要です。(ディスク障害発生後にただちに) データを回復できるように高信頼性が必要であるが、書き込みパフォーマンスはそれほど重要でない場合は、 RAID-5 ディスクの使用を考慮してください。あるいは、書き込みパフォーマンスが重要で、追加ディスク・スペースによるコストが大きくなってもこの状態を達成した場合は、データおよびログに RAID-1 を考慮してください。

ソフトウェア・ディスク・アレイ: ソフトウェア・ディスク・アレイはハードウェア・ディスク・アレイとほぼ同じ操作を実行しますが (52ページの『ハードウェア・ディスク・アレイ (RAID)』を参照)、ディスク・トラフィックの管理は、オペレーティング・システム・タスクまたはサーバーの下で実行されるアプリケーション・プログラムのいずれかが行います。他のプログラムと同様、ソフトウェア・アレイは CPU およびシステム・リソースを競合して獲得しなければなりません。したがって、CPU 制約システムには適しておらず、ディスク・アレイ全体のパフォーマンスがサーバーの CPU の負荷と容量に依存する点に注意する必要があります。

通常のソフトウェア・ディスク・アレイは、ディスク・ミラーリングを実行します (52ページの『ハードウェア・ディスク・アレイ (RAID)』を参照)。冗長性ディスクは必要ですが、高価な RAID ディスク制御装置は不要であるため、ソフトウェア・ディスク・アレイは比較的低価格で実現可能です。

注: オペレーティング・システムのブート・ドライブをディスク・アレイに設定すると、そのドライブに障害が発生した場合はシステムが始動しなくなります。ディスク・アレイが実行される前にドライブに障害が発生すると、ディスク・アレイは始動できないため、ドライブにアクセスすることはできません。ブート・ドライブは、ディスク・アレイから分離されていなければなりません。

トランザクション障害の影響の緩和

トランザクション障害の影響を緩和するためには、以下の条件が満たされているかどうか確認してください。

- 中断されない電源供給
- データベース・ログに十分なディスク・スペース

- 区分データベース環境においては、データベース区画サーバー間の高信頼性通信リンク
- 区分データベース環境では、システム・クロックの同期 (『区分データベース・システムにおけるシステム・クロックの同期』を参照)

区分データベース・システムにおけるシステム・クロックの同期

データベース区画サーバー全体で相対的な同期がとれたシステム・クロックを維持し、データベース操作がスムーズに行われ、順方向回復性に制限が加わらないようにします。データベース区画サーバー間の時間差にトランザクションの操作および通信遅延時間を加えたものは、 *max_time_diff* (ノード間最大時間差) データベース・マネージャー構成パラメーターの値より小さくしてください。

ログ・レコードのタイム・スタンプがトランザクションの順序を確実に示すようにするために、区分データベース・システムの DB2 は、ログ・レコードに記録するタイム・スタンプの基準として、各マシンのシステム・クロックを使用します。しかし、システム・クロックを進めると、ログ・クロックはそれに合わせて自動的に進みます。システム・クロックを遅らせることはできますが、ログのクロックを遅らせることはできず、システム・クロックをこの時刻に合わせるまでは、ずっと 進んだ時刻のままです。このとき、両方のクロックは同期しています。このことは、データベース・ノードで発生するシステム・クロック・エラーが短期間であっても、データベース・ログのタイム・スタンプではその影響が長く続くことがあることを意味します。

仮説的な例として、データベース区画サーバー A のシステム・クロックを、1997 年に間違って 1999 年 11 月 7 日に設定したものとします。また、その誤りは訂正されましたが、そのデータベース区画サーバーの区画で更新トランザクションがコミットされた後 であったとします。そのデータベースが連続的に使用されていてその間で定期的に更新されていると、 1997 年 11 月 7 日から 1999 年 11 月 7 日までの間の任意の時点は、ロールフォワード回復では実際には処理不能になります。データベース区画サーバー A でコミットが行われると、データベース・ログのタイム・スタンプは 1999 に設定され、データベース・ログのクロックは 1999 年 11 月 7 日のままです。この状態は、システム・クロックがこの時間と一致するまで続きます。この時間フレーム内の時点へロールフォワードしようとすると、指定された停止点 (1997 年 11 月 7 日) を超えた最初のタイム・スタンプで操作は停止します。

DB2 ではシステム・クロックに対する更新を制御することはできませんが、*max_time_diff* データベース・マネージャー構成パラメーターを指定しておくと、次のような問題が発生するのを防ぐことができます。

- このパラメーターに指定できる値は、1 分から 24 時間です。 *max_time_diff* の設定についての詳細は、*管理の手引き: パフォーマンス* を参照してください。
- 非カタログ・ノードに最初の接続要求が出されると、データベース区画サーバーはその時間をデータベースのカタログ・ノードに送信します。カタログ・ノードはその時間を受け取ると、接続を要求するノードの時間と自分自身の時間が、 *max_time_diff* パラメーターで指定された範囲内であることを検査します。この範囲を超えると、接続は拒否されます。
- 更新トランザクションがデータベース内の 3 つ以上のデータベース区画サーバーに関係している場合は、トランザクションは関係するデータベース区画サーバー間で同期がとれていることを確認した後、更新をコミットします。複数のデータベース区画サーバーの時間差が、 *max_time_diff* で指定した値を超えていると、トランザクションはロールバックされ、他のデータベース区画サーバーへ正しくない時間が伝送されるのを防ぎます。

データベース・ログ内の正しくないタイム・スタンプを訂正し、間違った値が伝搬されないようにするためには、次の操作を実行します。

1. システム・クロックを正しい時間に調整します。
2. 時間が不正に設定される前に作成したバックアップを使用し、該当するデータベース区画サーバー上でデータベース区画を復元します。
3. データベース区画のログの最後まで、変更項目をロールフォワードします。
4. 変更項目がロールフォワードされた直後、データベース区画のバックアップ・コピーを作成します。

これらの処置を完了すると、ログ時間は調整され、正しくないタイム・スタンプが伝搬されることがなく、区画について最後に作成したバックアップを使用して、データベース区画で時刻指定回復を実行することができるようになります。

データベースでの表の再編成

何度も更新を行うと表は断片化され、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、断片化されたデータに伴う空き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 の機密保護の概説

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、外部機密保護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、いくつかの機密保護検査にパスしなければなりません。データベース機密保護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 の外部の機密保護機能を使用して完了します。機密保護機能は、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、機密保護機能は、オペレーティング・システムそのものの中にあります。 DCE 機密保護サービスは、分散環境に対して機密保護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、機密保護機能はありません。

機密保護は、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、機密保護機能にユーザーを知らせます。正しいパスワード (ユーザーおよび機密保護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 機密保護サービス製品では、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、機密保護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、機密保護機能を使用して、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 は成功した機密保護システム・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機密保護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。この変数の省略時値は 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機密保護を保守します。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。 62ページの図20 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。

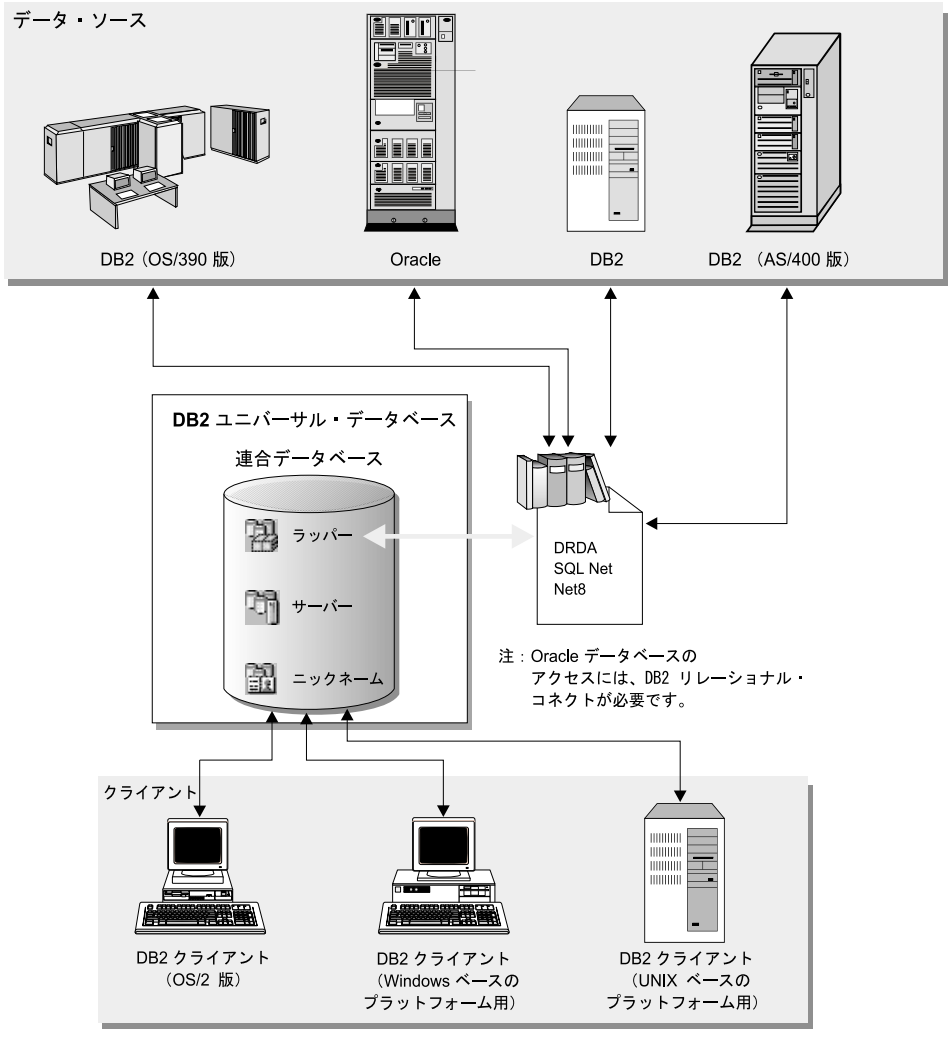


図 20. 連合データベース・システム

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識別子。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・オブジェクトは並行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 を並行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『設計の実装』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、多重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。

67ページの図21 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 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。

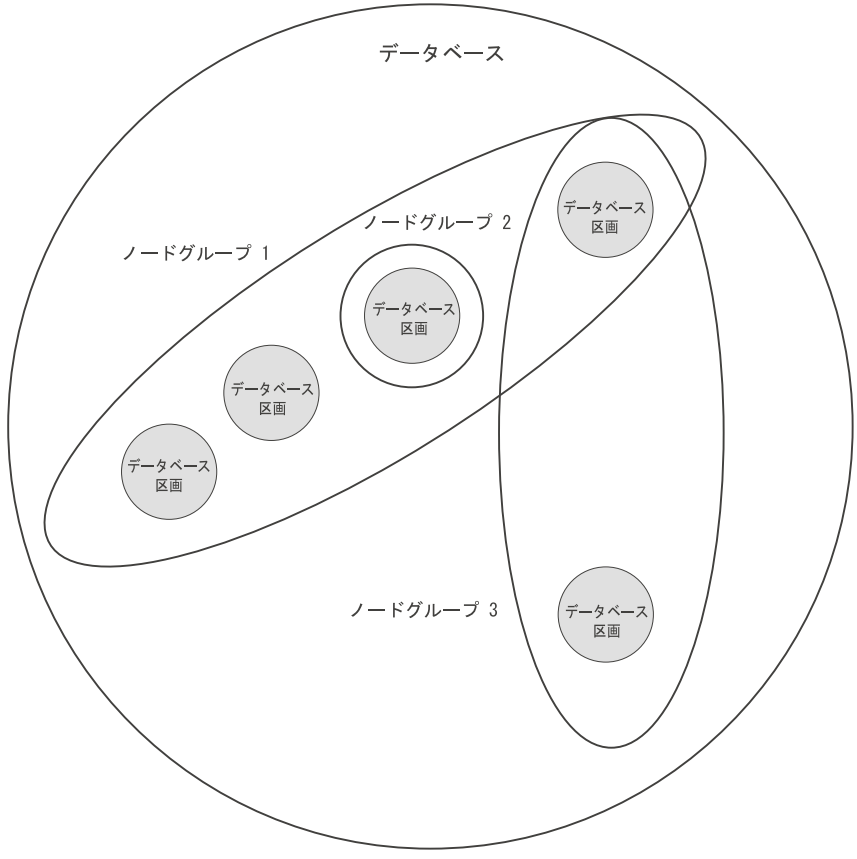


図 21. データベース内のノードグループ

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、136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構成要素は、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、72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構成要素です。 81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、*区画内並行化* または*区画間並行化* (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。

区画内並行化

区画内並行化 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。 71ページの『ユーティリティー並列化』を参照してください。)

区画内並行化では、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22 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。区画内並行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。

81ページの表3 は、区画内並行化に最適なハードウェア環境の一覧を示したものです。

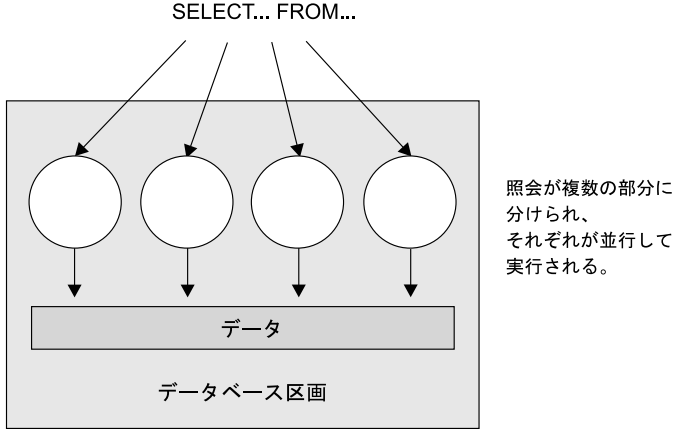


図 22. 区画内並行化

区画間並行化

区画間並行化 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。 71ページの『ユーティリティー並列化』を参照してください。)

区画間並行化では、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23 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。

81ページの表3 は、区画間並行化に最適なハードウェア環境の一覧を示したものです。

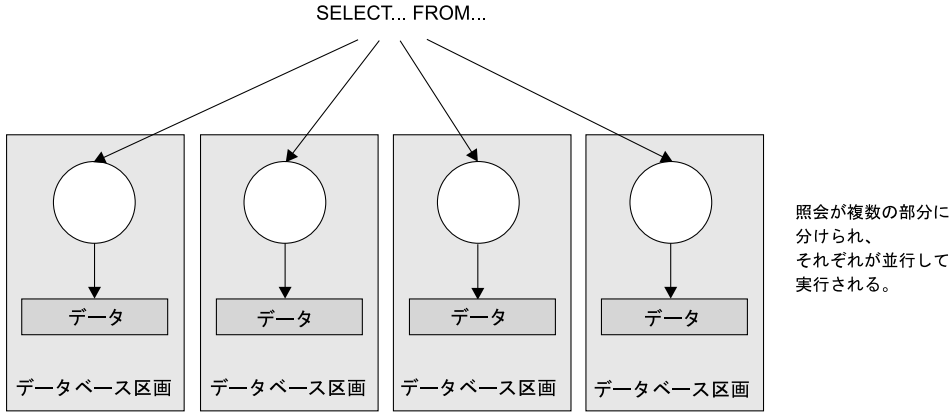


図 23. 区画間並行化

区画内並行化と区画間並行化の同時使用

区画内並行化と区画間並行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。

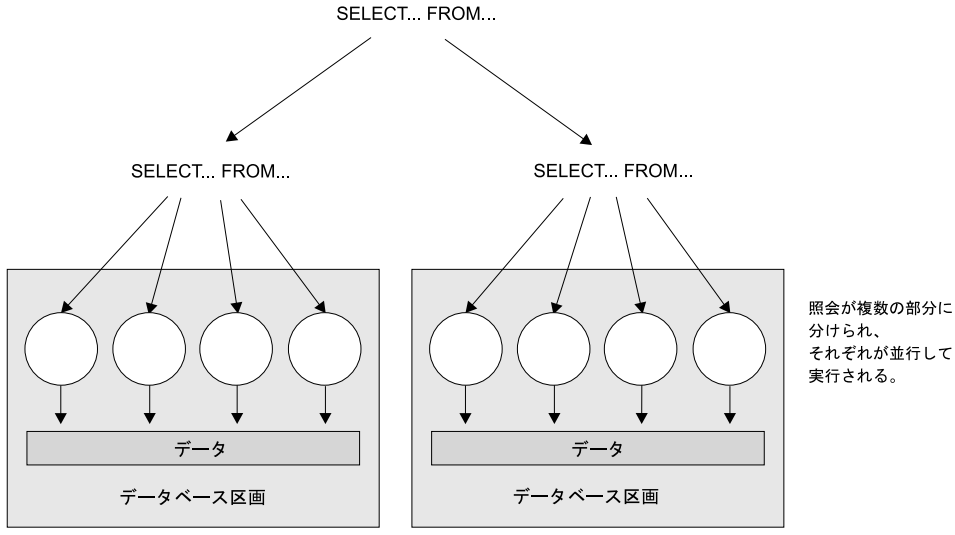


図 24. 区画内並行化と区画間並行化の同時使用

ユーティリティー並列化

DB2 のユーティリティーは、区画内並行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行化も利用しており、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、区画内並行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、および入出力並行処理を利用します。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行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行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媒体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、および記憶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 (73ページの図25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。

81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。

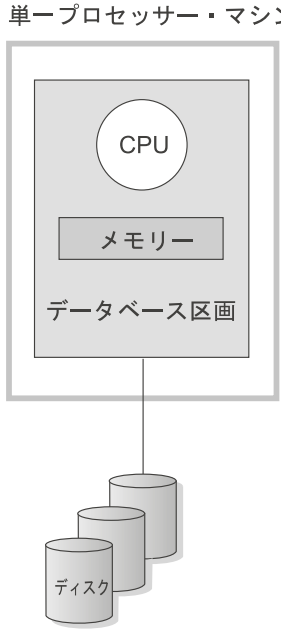


図 25. 単一プロセッサー上の単一区画

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。また、この環境に対してさらにハード・ディスク・スペースを追加することもできます。

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。しかし、作業負荷が増加すると、メモリーやディスクなどの構成要素にかかわりなく、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移行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 (75ページの図26 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。

81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。

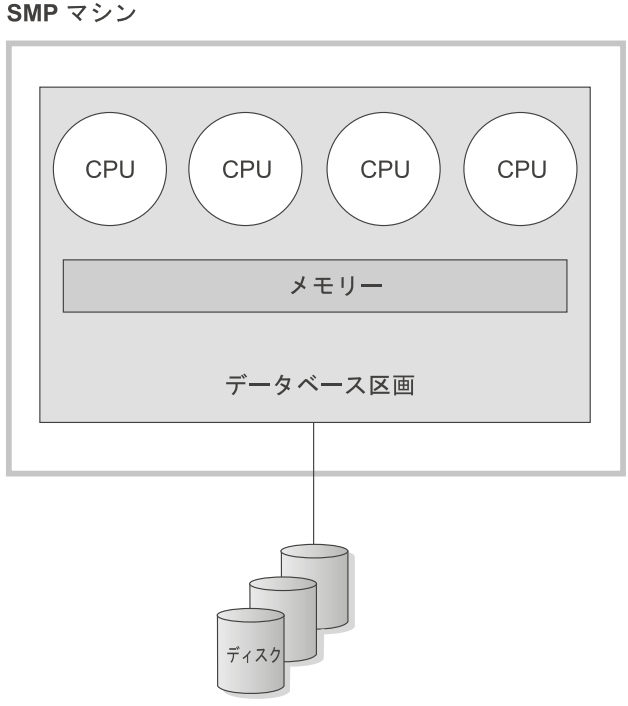


図 26. 単一区画データベースの対称マルチプロセッサー・システム

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移行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 (77ページの図27)。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。

81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。

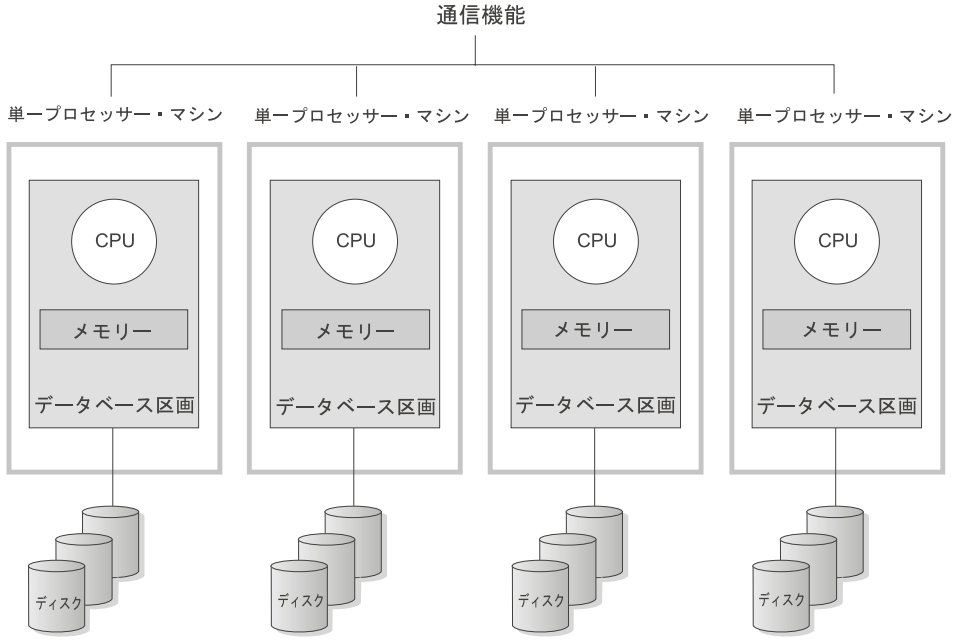


図 27. 大量並列処理システム

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移行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 (78ページの図28)。

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。

81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。

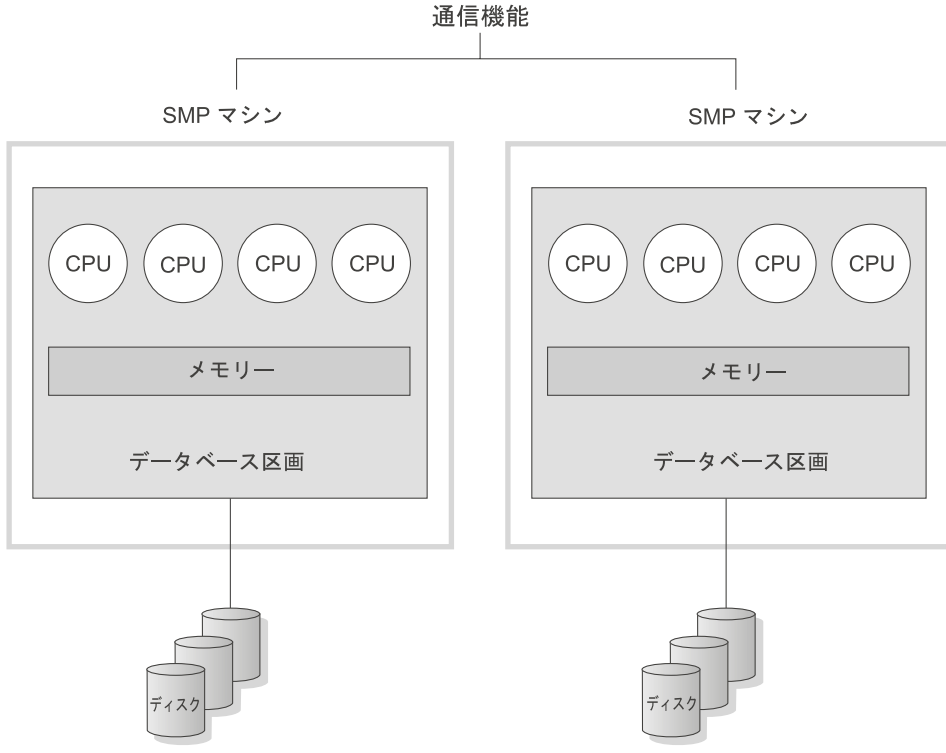


図 28. SMP のクラスター

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。 79ページの図29 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回復することができます。

大規模な **SMP** マシン

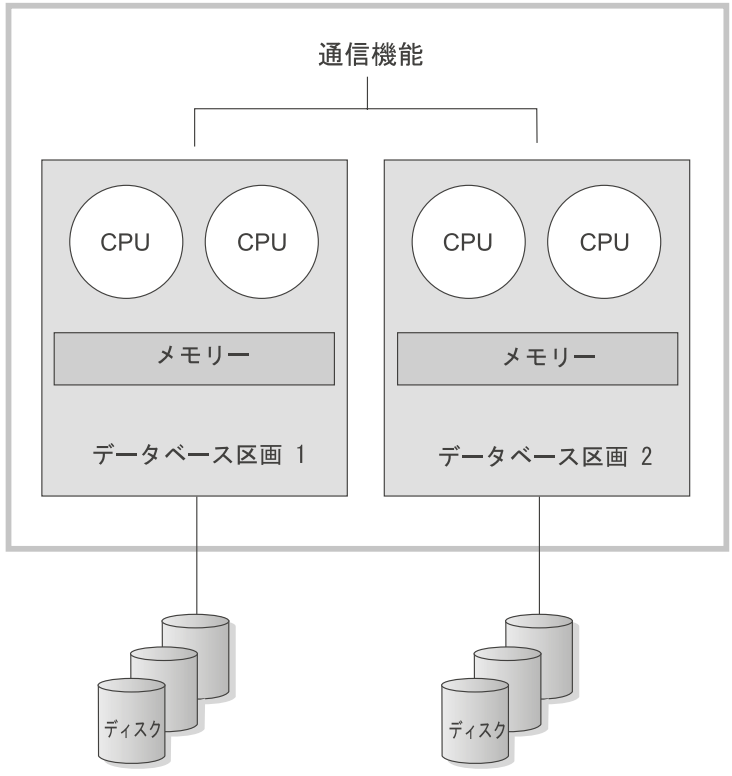


図 29. 区分データベース、対称マルチプロセッサー・システム

80ページの図30 は、処理能力を高めるために、 図29 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。

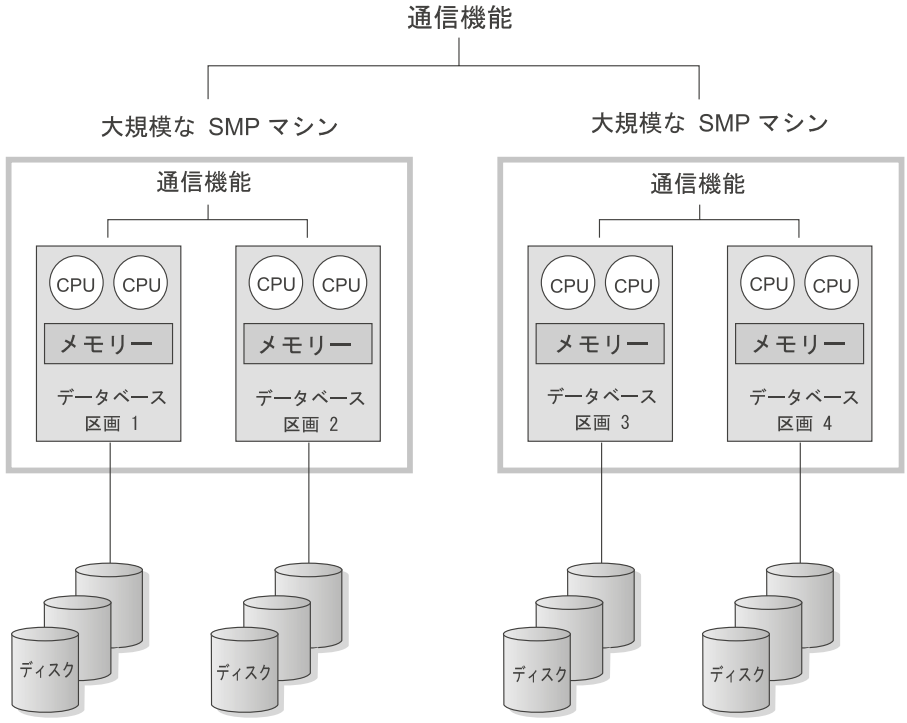


図 30. 区分データベース、一緒にクラスター化された対称マルチプロセッサー・システム

81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、227ページの『第11章 高可用性の設計』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	
		区画内並行化	区画間並行化
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構成要素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・コネクティビィティー (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。

必要に応じてに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。
- 機密保護を定義し、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空間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。

空間情報 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)。

空間情報を、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、ビジュアルに空間情報を生成して、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、空間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
- 空間データの特性
- 空間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、地勢は表または視点の行、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31 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。

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 31. 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。

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。

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、空間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。 図32 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 32. 空間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識別子を空間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空間情報のもとになるため、これらの引き出された値は*空間データ* と呼ばれます。次のセクションでは、空間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。

空間データの特性

多くの空間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、 Yellow Stone 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。

最も単純な空間データ項目は、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。さらに広範囲な空間データ項目は、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。

それぞれの空間データ項目は、空間データ・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、空間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。

空間データの取得方法

空間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
- 他の空間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

空間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。 91ページの図32 は、空間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33 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 33. ソース・データから導出された空間データを含む表. 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、結果の空間データを列に配置します。

他の空間データのソース・データとしての使用

空間データは属性データからだけでなく、他の空間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。 図34 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 34. 既存の空間データから導出された新しい空間データを含む表. 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。

空間データのインポート

空間データを取得する 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、生成した空間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』
- 99ページの『各タイプの関係の表を定義する』
- 102ページの『すべての表の列を定義する』
- 104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』
- 108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』
- 108ページの『表の正規化について』
- 113ページの『制約の強制について計画する』
- 121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、 123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『設計の実装』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』
- 100ページの『多対多の関係』
- 101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、両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、両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、391ページの『付録B. 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型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み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、オブジェクト識別子と共に、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回復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. どの列に省略時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省略時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。

   ヌル値および省略時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。 (106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。 24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行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行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、結果として並行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・タイプとみなされます。)
- 連続値は、指定した整数増分値によって異なることがあります。デフォルトの増分値は 1 です。
- 識別列のカウンター値は回復可能です。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 (『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 (『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 (111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 (112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
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 (112ページの表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、付加的な記憶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	English
000130	Database Design	English
000130	Application Design	English
000130	Data Modelling	Spanish
000130	Database Design	Spanish
000130	Application Design	Spanish

むしろ、これらの関係は 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
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	English
000130	Spanish

しかし、属性が相互依存の関係にある場合 (つまり従業員がある言語を特定の技術だけに使用するような場合)、表を分割す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
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逆トレース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』
- 118ページの『DELETE 規則』
- 119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を除き、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、値のいずれかの要素がヌル値であれば、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*、*活動記録データ*、*要約表*、*機密保護*、*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、*活動記録* 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、活動記録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。

機密保護 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・アクセスを機密保護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、最も厳重な機密保護制限があります。機密保護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『設計の実装』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 (97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』
- 126ページの『表スペース所要量の見積もり』
- 134ページの『さらに必要となるスペース量』
- 136ページの『ノードグループの設計』
- 145ページの『表スペースの設計と選択』
- 170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、省略時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
```

ここで、

- specified_path は、インスタンスをインストールするための任意選択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識別子です。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。

省略時には、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、データベースの活動記録ファイルです。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活動記録が保管されます。

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、直接変更しないようにしてください。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、ディスクの断片化によって発生するオーバーヘッド、空きスペース、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記憶域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)」ダイアログを使用すると、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。

- 『システム・カタログ表』
- 128ページの『ユーザー表データ』
- 130ページの『長形式フィールドのデータ』
- 130ページの『ラージ・オブジェクト (LOB) データ』
- 131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、145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、130ページの『長形式フィールドのデータ』および 130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 (空きスペース・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・ページの空き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。断片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 ページの行数は、最大で 255 であることを思い起こしてください。) 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 (128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 32,700 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、空きスペースを容易に再利用できるような方法で保管されます。割り振りと空き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 (128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、単にもっとも近い 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・オブジェクトでの空きスペース量は、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**

  割り振りと空き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、非葉ページや空き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 の場合、必要な最小スペースは 12 KB です。 DMS 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。

```
(.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 パーセント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。

  $$\frac{(.9 * (U - (M*2))) * (D + 1)}{K + 12 + (8 * D)}$$

  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 パーセントの空き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 パーセントを追加するのは、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』
- 135ページの『一時作業スペース』

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。

これら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* 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・マネージャーが実行時に必要とする 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、ときどき発生する、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回復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。

一時作業スペース

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分類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作業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**の中にすでに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することができます。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、139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、140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省略時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省略時ノードグループです。 IBMTEMPGROUP は、システム一時表スペース用の省略時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省略時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回復させるためには、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みなされます。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、併置されているとみなされます。

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』
- 139ページの『区分化マップ』
- 140ページの『区分化キー』
- 142ページの『表の併置』
- 143ページの『区画の互換性』
- 144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、区分化記憶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写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写します。詳しくは、144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。

140ページの図35 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。

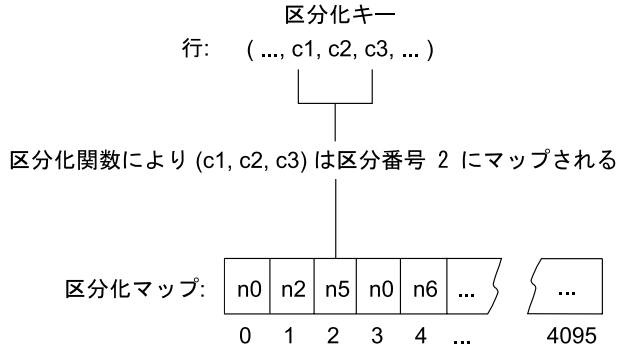


図 35. 区分化マップを使用したデータ配分

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、省略時解釈によって、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、省略時解釈の区分化キーは、その表に定義された最初のロング・フィールド列以外の列になります。 (*ロング* には、すべてのロング・データ・タイプとすべてのラージ・オブジェクト・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。省略時解釈では、区分化キーは作成されません。

省略時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することができます。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。

これによって、同じ区分化キー値を持つ併置された表の各行が、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、143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、142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。

- ロング・データ・タイプ (LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、または DBCLOB) のみを含む複数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。
- 表の 1 つの行について、区分化キー列値を更新することはできません。

- 区分化キー列値は、削除または挿入のみが可能です。
- 区分化キーには、もっと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。

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、139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、『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 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することができます。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。 REPLICATED キーワードは、 REFRESH DEFERRED オプションとともに定義された要約表に対してのみ指定できます。

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ディメンション表のある結合について特に役立ちます。必要とされる余分の記憶域を最小限にし、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記憶モデルです。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。 146ページの図36 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。

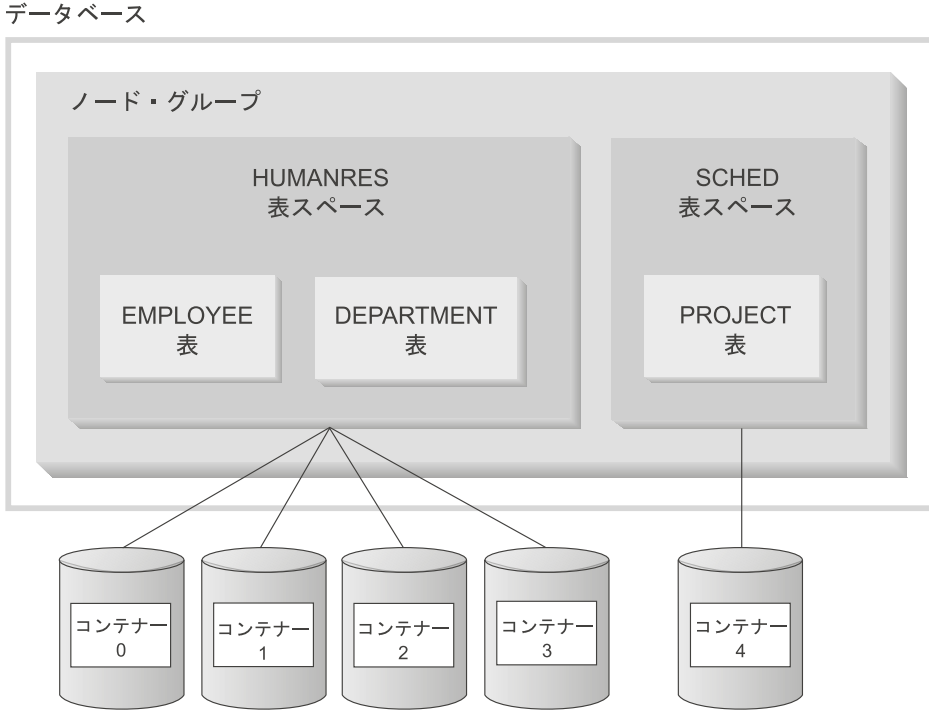


図 36. データベース内の表スペースと表

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。

147ページの図37 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。

HUMANRES 表スペース

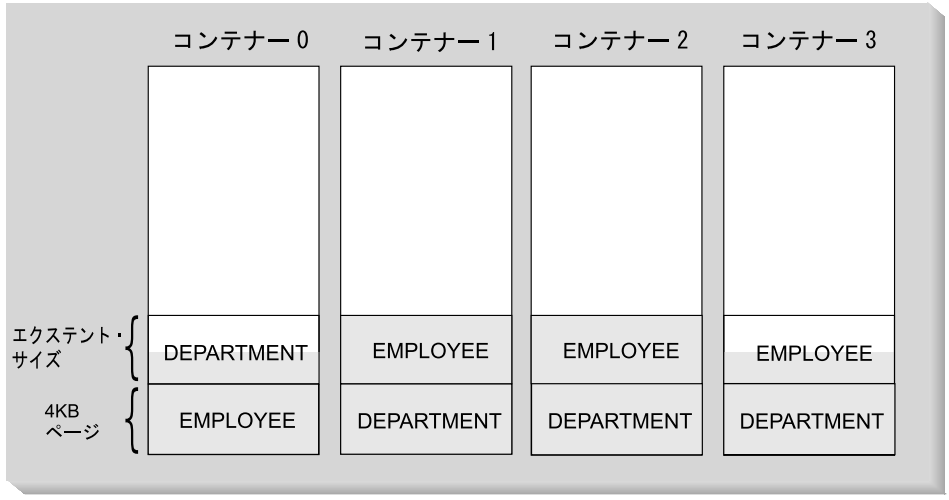


図 37. コンテナーとエクステント

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省略時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。省略時には、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省略時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 (省略時)、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。省略時には、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省略時ノードグループです。ユーザー一時表スペースは、省略時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する場合、一時オブジェクトはラウンドロビン方式で一時表スペースに割り振られます。

  省略時の 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と (たとえば 1012 列に対する ORDER BY)、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。 162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 つの省略時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。

- 149ページの『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・マネージャーが記憶スペースを制御します。
- 153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・マネージャーが記憶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。記憶モデルは、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、省略時の表スペースです。

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、151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、161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省略時エクステント・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、省略時エクステント・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、表の識別子 (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　　説明

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空き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・マネージャーが記憶スペースを制御します。記憶モデルは、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空き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することがあります。 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。

- 『表スペースの入出力 (I/O) についての考慮事項』
- 158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』
- 159ページの『ノードグループへの表スペースのマッピング』
- 160ページの『表スペースへの表のマッピング』
- 161ページの『エクステント・サイズの選択』
- 162ページの『一時表スペースについての推奨事項』
- 164ページの『カタログ表スペースについての推奨事項』
- 165ページの『作業負荷についての考慮事項』
- 166ページの『SMS 表スペースか DMS 表スペースかの選択』
- 168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、断片化が起こりやすい場合にもっともよく発生します。 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省略時値は、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行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省略時値は、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、制御装置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、実行時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。省略時のバッファー・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・プールに必要な記憶域が、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 が必要な記憶域を獲得できない場合、データベース・マネージャーは省略時バッファー・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、各表スペースは省略時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省略時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、 136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、161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履歴データの入った表がある場合、このデータに対する照会については、応答時間が長くてもよいとエンド・ユーザーは考えるかもしれません。その場合、履歴データ表に別の表スペースを使用して、その表スペースをアクセス速度が遅く、費用のかからない物理装置に割り当てるのも一案です。

  また、高可用性や短い応答時間が要求される重要な表を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、ロールフォワード回復が使用可能であれば、表のすべての部分をまとめてバックアップし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 パーセント以上が同じページ・サイズを使用する場合、一時表スペースの定義で同じページ・サイズを指定するのは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記憶要件がピーク時に非常に高く、普段の記憶要件はとても低いという場合があります。 DMS では、ピーク時の記憶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記憶域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。
- 省略時の 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生の装置を含む DMS 表スペースを使用することを強く推奨します。

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、SMS 表スペースは一時表スペースとしてのみ使用し、 DMS 表スペースを分割して、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 (後述) 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 (後述) 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。次に例を示します。

```
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、タグが独自の (フル) エクステント内に存在できるような仕方で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記憶域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、照会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・プロセスの中で回復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。

- 172ページの『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』

- 173ページの『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』
- 179ページの『構成についてのその他の考慮事項』
- 182ページの『2 フェーズ・コミット・プロセスについて』
- 185ページの『2 フェーズ・コミット時の問題を回復する』

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。

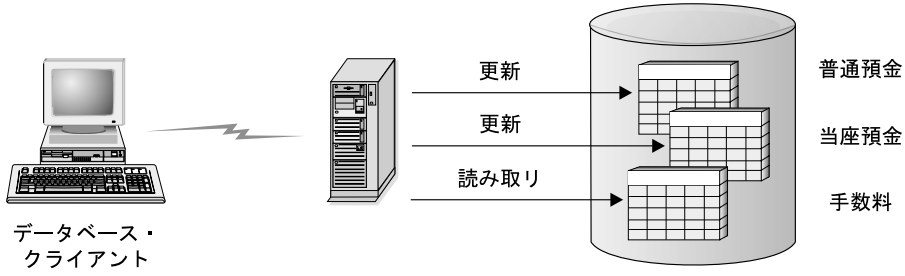


図 38. 1 つのトランザクションで単一のデータベースを使用する場合

図38 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『設計の実装』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をカタログ化する。
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を*複数サイト更新* または *2 フェーズ・コミット* といい、単一の作業単位内 (トランザクション) で実行できます。複数サイト更新の別の例については、175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。

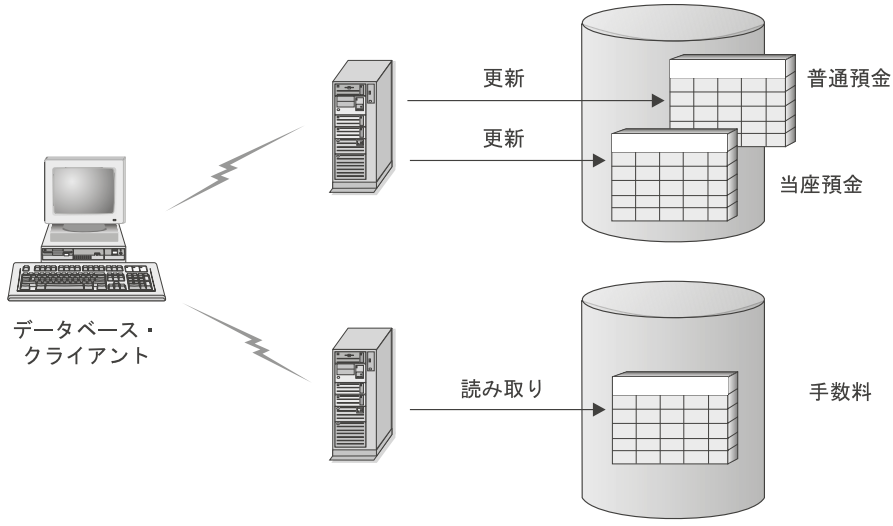


図 39. 単一のトランザクションで複数のデータベースを使用する場合

図39 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、172ページの図38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『設計の実装』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をカタログ化する。
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、*複数サイト更新* と呼ばれます。

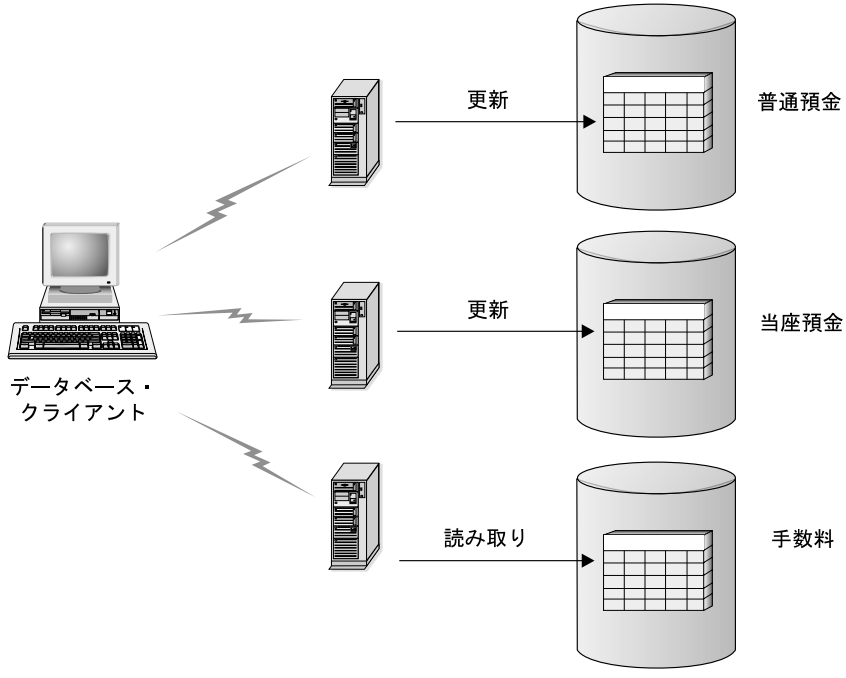


図 40. 単一のトランザクションで複数のデータベースを更新する

図40 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『設計の実装』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をカタログ化する。
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。
5. 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)。

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 189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、複数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。

  – 複数サイト更新に、 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、189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回復ログは、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を正しくカタログ化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化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、複数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を正しくカタログ化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化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。

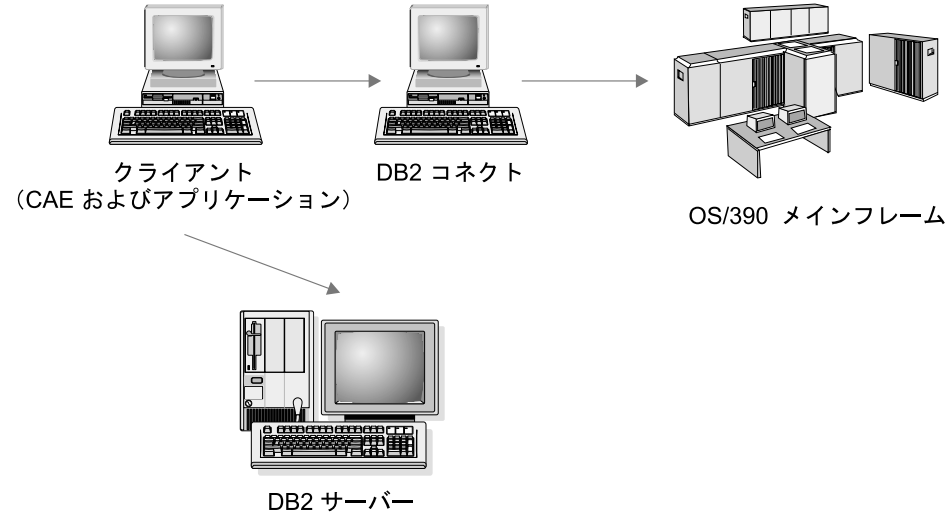


図 41. 構成についての考慮事項

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回復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、185ページの『2 フェーズ・コミット時の問題を回復する』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。省略時値は 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。

複数サイト更新で 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複数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。複数サイト更新には 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、機密保護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複数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複数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

183ページの図42 は複数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。

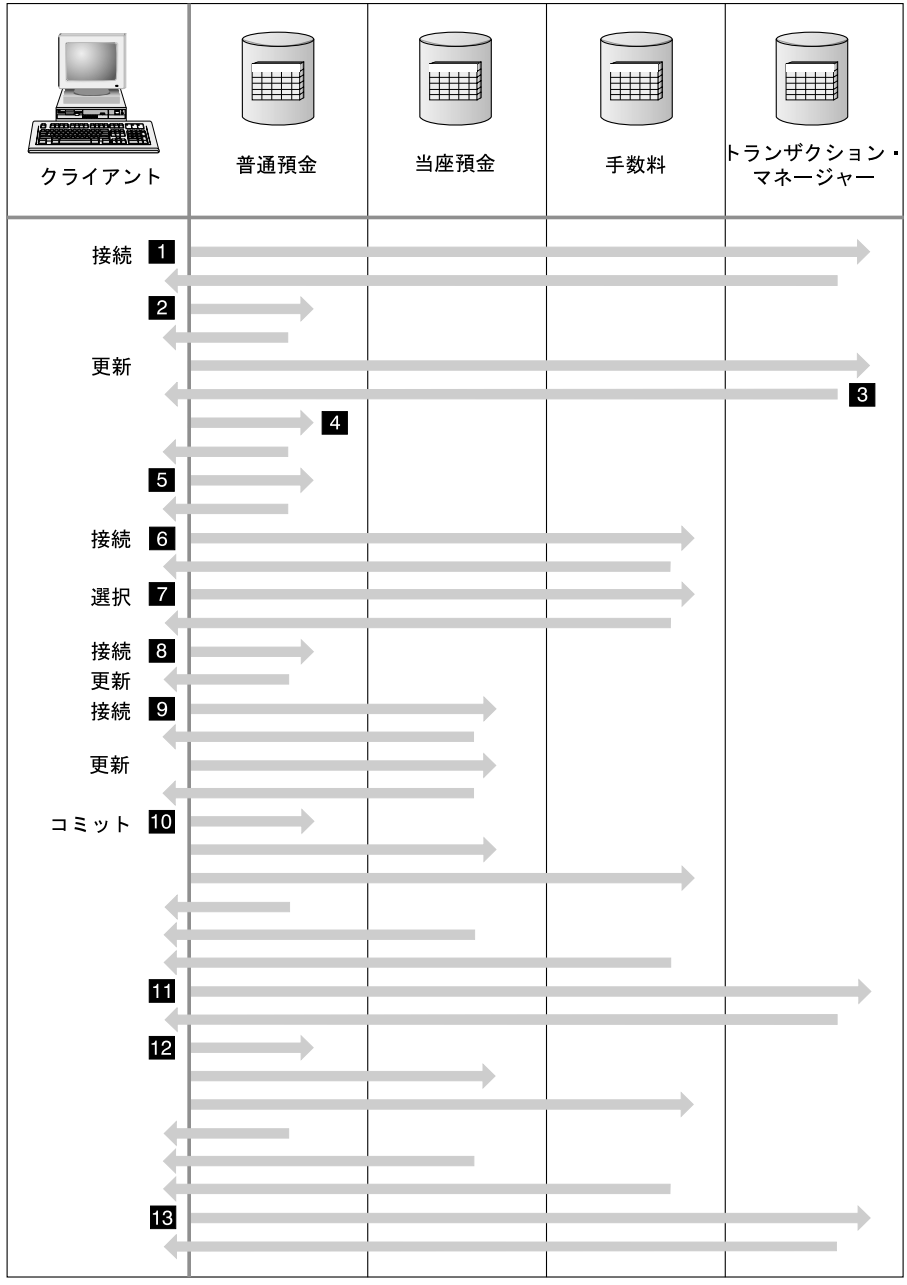


図 42. 複数のデータベースの更新

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。これは事前コンパイル・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回復する

エラー条件からの回復は、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、182ページの『2 フェーズ・コミット・プロセスについて』の **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みなします。)

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みなして、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回復について詳しくは、 203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、 179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、 175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 つの構成要素があります。

- 『アプリケーション・プログラム (AP)』
- 192ページの『トランザクション・マネージャー (TM)』
- 193ページの『リソース・マネージャー (RM)』

図43 は、このモデルと 3 つの構成要素の相互関係を示しています。

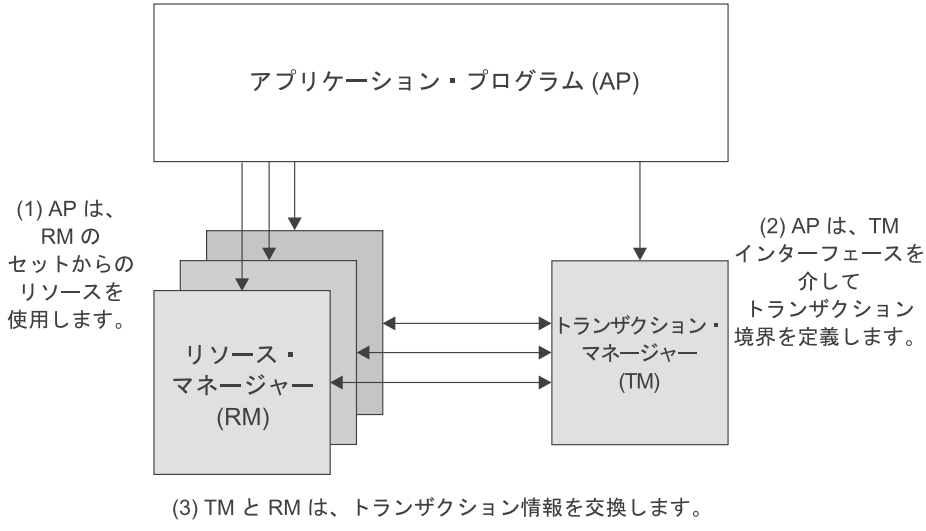


図 43. X/Open 分散トランザクション処理 (DTP) のモデル

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 一時データ待ち行列などのリソース・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、機密保護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、206ページの『機密保護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。

- 大域トランザクション

  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。大域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 との間の大域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 つの大域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、あたかも別々の大域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、疎結合の大域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大域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。 175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、密結合大域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 からは単一の実体とみなされます。 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識別子を割り当て、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識別子 (XID と呼ばれるもの) は TM によって割り当てられ、大域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さいに、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* を回復します。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大域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*大域トランザクション* 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、208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 が個々の大域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 ソフトウェア構成要素の間のシステム・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 構成要素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、 211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移行してください。 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、201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	はい	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	いいえ	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	いいえ	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	いいえ	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、198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。	いいえ	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。	いいえ	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。				

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
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。	いいえ	なし	F
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。	いいえ	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

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	

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。
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. 待ち行列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 (195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) に移行してください。

各データベースは、トランザクション・マネージャー (TM) に対して別個のリソース・マネージャー (RM) として定義されているので、次の構文の xa_open ストリングによってデータベースを識別する必要があります。

```
"database_alias<,userid,password>"
```

database_alias は必須であり、データベースの別名を指定するものです。データベース作成後に明示的に別名をカタログ化したのでない限り、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、 182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 (185ページの『2 フェーズ・コミット時の問題を回復する』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回復するには、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回復するか、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記憶域などが含まれます。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* は大域トランザクション 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。

機密保護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。

実行時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、185ページの『2 フェーズ・コミット時の問題を回復する』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。省略時値は 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、193ページの『リソース・マネージャー (RM)』 を参照してください。

- 関連の移行

  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移行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、 195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 は関連の移行を使用してはならないことを明示的に示します。非同期操作がサポートされないことを暗黙的に示します。
version	常に 0。

DB2 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)
  ```

  しかし、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。
  - データベースがカタログ化されなかった。
  - 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

このセクションに書かれている情報は、 *管理の手引き: パフォーマンス* にある類似のセクションよりも優先されます。

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』
- 215ページの『BEA Tuxedo の構成』
- 217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/ でオンライン表示することができます。

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/ でオンライン表示することができます。

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。次に例を示します。

   ```
   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。次に例を示します。

```
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、195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。次に例を示します。

   ```
   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。次に例を示します。

   ```
   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 が使用する内部識別子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 (この場合、 Tuxedo 構成ファイルの UDB グループの TMSCOUNT パラメーターによって指定された UDBCONFIG)。

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。

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。

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。

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. 複数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、複数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 サーバーが複数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。複数サイト更新で 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 1)

- 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)

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。

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複数サイト更新用に、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 5.2
- DB2 エンタープライズ拡張エディション バージョン 5.2
- 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 (VM および VSE 版)
- DB2 共通サーバー (SCO 版) バージョン 2
- DB2 ユニバーサル・データベース (AIX 版) PTF U453782 適用済み

- DB2 ユニバーサル・データベース (HP-UX 版) PTF U453784 適用済み
- 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 (OS/2 版) PTF WR09033 適用済み
- DB2 ユニバーサル・データベース (SOLARIS 版) PTF U453783 適用済み
- 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 (Windows NT 版) PTF WR09034 適用済み
- 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 (UNIX または Windows NT 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 (222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、 Fix Pack 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、それを平文テキスト・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・システムにこれら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、それを平文テキスト・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、各パラメーターは以下のとおりです。

   - *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。次に例を示します。

      ```
      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 高可用性の設計

DB2 ユニバーサル・データベースは、多くのプラットフォームで*高可用性フェールオーバー・サポート* を提供します。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、あるプロセッサーから別のプロセッサーにワークロードを自動的に移すことが可能になります。たとえば AIX 上では、IBM High Availability Cluster Multi-Processing (HACMP) の機能を用いることにより、 DB2 UDB がフェールオーバーをサポートします。このセクションでは、 AIX での例を使用して、高可用性と関連する概念を紹介しています。

HACMP は、ディスクやネットワーク・アクセスなどのリソースを共用するプロセッサーのクラスターを介して、高度の可用性を提供します。 1 つのプロセッサーに障害が起こると、クラスターにある別のプロセッサーがそれを置き換えます。

フェールオーバー・サポートには 3 つのモードがあります。

ホット・スタンドバイ
このモードでは、一方のプロセッサーが使用されて DB2 インスタンスを実行します。 2 番目のプロセッサーはスタンドバイ・モードになっており、最初のプロセッサーにかかわるオペレーティング・システムまたはハードウェアの故障があったときに、インスタンスを引き受ける準備をしています。

相互引き受け
このモードでは、以下のようになります。

- 両方のプロセッサーが別個の DB2 インスタンスを実行するために使用されています。
- 一方が DB2 インスタンスを実行し、他方が DB2 アプリケーションを実行しています。

一方のプロセッサーでオペレーティング・システムまたはハードウェアの故障が起こると、他方のプロセッサーが故障したプロセッサーのタスクを引き受け、結果的に両方のプロセッサーの作業をします。

同時アクセス
このモードでは、複数のプロセッサーを使用して、 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 (EEE) 製品の単

一のデータベース・インスタンスにスケールします。これは、非共用のモデルを使用し、クラスターにある各プロセッサーで 1 つまたはいくつかの区画が稼働するような形で、データを区分化することによって行われます。一方のプロセッサーでオペレーティング・システムまたはハードウェアの故障が起こると、他方のプロセッサーが故障したプロセッサーの区画を引き受けます。 DB2 UDB EEE では、冗長度を持つ目的のために同時リソース・マネージャーを使用する必要はありません。冗長度は、ホット・スタンドバイまたは相互引き受けモードを使用して管理されます。同時アクセス・モードの機能は、共用アーキテクチャーをもつデータベース・マネージャーでのみ必要とされます。

上記の構成はそれぞれ、区分データベースの 1 つまたはいくつかの区画をフェールオーバーするために使用できます。さらに、それぞれが、単一区画インストールの完全なインスタンスをフェールオーバーすることができます。

ホット・スタンドバイ

ホット・スタンドバイ 機能を使用すると、単一区画データベースのインスタンス全体または区分データベース構成の 1 つの区画をフェールオーバーすることができます。あるプロセッサーが故障すると、そのクラスターにある別のプロセッサーが、自動的にインスタンスを転送することによって、故障したプロセッサーと置き換わることができます。データベース・インスタンスと実際のデータベースが、 1 次およびフェールオーバーの両方のプロセッサーからアクセスできなければなりません。

- DB2 インストール・パスは、両方のシステムから共用されるパス、または非共用ファイル・システムのどちらかに置くことができます。非共用ファイル・システムを使用する場合は、インストールのレベルが同じでなければなりません。
- DB2 インスタンス・パスは、共用ファイル・システム、または手動でミラーリングされるファイル・システムのどちらかに置くことができます。
- データベースおよび関連するコンテナーは、両方のシステムからアクセス可能なファイル・システム (または装置) に置く必要があります。
- 区分データベース構成での区画のフェールオーバーの場合、区画は、 2 番目のプロセッサーで再始動されます。フェールオーバーのスクリプトは、新しいプロセッサー上のこの区画を指すよう `db2nodes.cfg` ファイルを変更し、そのプロセッサー上で区画を始動します。
- フェールオーバーが起こると、サポートされる通信プロトコルの外部通信アドレスがフェールオーバー・プロシージャーの一部として、透過的に転送されます。

実際のインストール要件およびインスタンスの作成に関する詳細については、*HACMP for AIX, Version 4.2: Installation Guide* (SC23-1940) を参照してください。

例

以下の例ではそれぞれ、サンプル・スクリプトが、(DB2 (AIX 版) のインストールの) `sqllib/samples/hacmp` の中に格納されています。

インスタンスのフェールオーバー

次に示すホット・スタンドバイのフェールオーバーのシナリオは、単一区画のデータベースのインスタンスを実行する、 2 つのプロセッサーからなる HACMP クラスターから構成されます (図44)。 HACMP クラスターの構成に関する詳細は、236ページの『リソース』を参照してください。

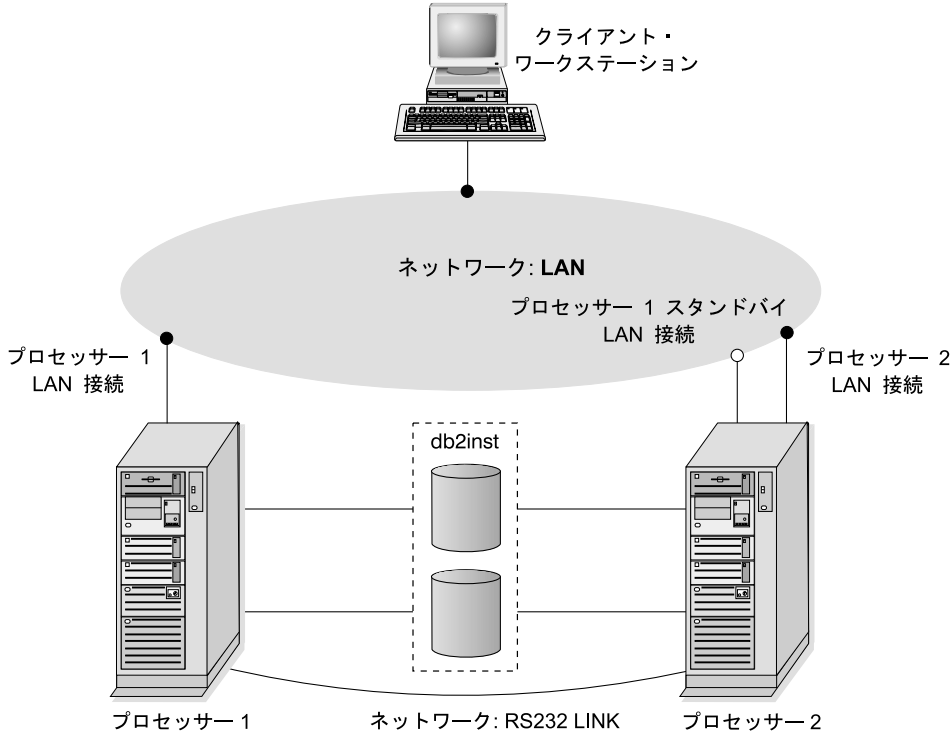


図 44. ホット・スタンドバイ・フェールオーバー構成の例

どちらのプロセッサーも、インストール・ディレクトリー、インスタンス・ディレクトリー、およびデータベース・ディレクトリーにアクセスできます。データベース・インスタンス ″db2inst″ は、プロセッサー 1 で活発に実行されて

います。プロセッサー 2 は活動状態でなく、ホット・スタンドバイとして使用されています。プロセッサー 1 で故障が起こり、インスタンスはプロセッサー 2 が引き受けます。フェールオーバーが完了すると、リモートおよびローカルの両方のアプリケーションが、インスタンス ″db2inst″ の中のデータベースにアクセスできるようになります。データベースは、手作業で再始動する必要があります。そうしない場合、AUTORESTART がオンであれば、データベースに最初に接続するときに再始動操作が開始されます。提供されているサンプル・スクリプトでは、AUTORESTART はオフで、フェールオーバー・スクリプトがデータベースの再始動を行うという想定になっています。
AUTORESTART 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* の回復の概要に関する項を参照してください。

サンプル・スクリプト:

```
hacmp-s1.sh
```

区画フェールオーバー

次に示すホット・スタンドバイ・フェールオーバー・シナリオでは、インスタンス全体の代わりにインスタンス区画を使用します。このシナリオでは、前の例と同様に、2 つのプロセッサーからなる HACMP クラスターを含みますが、マシンは、区分データベース・サーバーの 1 つの区画を表します。プロセッサー 1 は、構成全体のうちの 1 つの区画を実行し、プロセッサー 2 は、フェールオーバー・プロセッサーとして使用されます。プロセッサー 1 が故障すると、2 番目のプロセッサーで区画が再始動されます。フェールオーバーは db2nodes.cfg ファイルを更新し、プロセッサー 2 のホスト名とネット名を指し示すようにし、新しいプロセッサーで区画を再始動させます。

以下に、フェールオーバーの前と後の db2nodes.cfg ファイルの一部を示します。この例では、ノード番号 2 が HACMP マシンのプロセッサー 1 で実行されています。このマシンはホスト名とネット名の両方が ″node201″ です。フェールオーバーのあと、ノード番号 2 が HACMP マシンのプロセッサー 2 で実行されています。このマシンはホスト名とネット名の両方が ″node202″ です。

```
前:
        1 node101 0 node101
        2 node201 0 node201    <= HACMP
        3 node301 0 node301
   db2start nodenum 2 restart hostname node202 port 0 netname node202
後:
        1 node101 0 node101
        2 node202 0 node202    <= HACMP
        3 node301 0 node301
```

サンプル・スクリプト:

```
hacmp-s2.sh
```

複数の論理ノードのフェールオーバー

前述の例をさらに複雑にした例は、あるプロセッサーから別のプロセッサーへの、複数の論理ノードのフェールオーバーにかかわるものです。ここでも、前の例と同様、 2 つのプロセッサーからなる HACMP クラスター構成を使用しています。しかし、このシナリオでは、プロセッサー 1 は、3 つの論理区画を実行しています。このセットアップは、単純な区画フェールオーバーのシナリオの場合と同じですが、このケースでは、プロセッサー 1 が故障すると、論理区画はそれぞれプロセッサー 2 で開始する必要があります。各論理区画は、db2nodes.cfg ファイルで定義されている順序で開始する必要があります。ポート番号 0 の論理区画を常に最初に開始する必要があります。

以下に、フェールオーバーの前と後の db2nodes.cfg ファイルの一部を示します。この例では、 2 つのプロセッサーからなる HACMP クラスターのプロセッサー 1 で定義されている論理区画が 3 つあります。

```
前:
        1 node101 0 node101
        2 node201 0 node201    <= HACMP
        3 node201 1 node201    <= HACMP
        4 node201 2 node201    <= HACMP
        5 node301 0 node301
   db2start nodenum 2 restart hostname node202 port 0 netname node202
   db2start nodenum 3 restart hostname node202 port 1 netname node202
   db2start nodenum 4 restart hostname node202 port 2 netname node202
後:
        1 node101 0 node101
        2 node202 0 node202    <= HACMP
        3 node202 1 node202    <= HACMP
        4 node202 2 node202    <= HACMP
        5 node301 0 node301
```

サンプル・スクリプト:

```
hacmp-s3.sh
```

相互引き受け

相互引き受け モードでは、ある 1 つのプロセッサーは、区分データベース構成のもう 1 つのインスタンスまたは他の区画を実行する一方で、単一区画データベース・インスタンスまたは区分データベースの区画をフェールオーバーします。ホット・スタンドバイ構成の場合と同様、インストール・パス、インスタンス・ディレクトリー、およびデータベースは、フェールオーバー処理に関係する可能性のある各プロセッサーからアクセスできるものでなければなりま

せん。インストール・パスとインスタンス・パスは、共用ファイル・システムに置くか、別個のファイル・システムにミラーリングすることができます。

インスタンスのフェールオーバーについて相互引き受け戦略を使用するときは、両方のインスタンスを同じプロセッサーで同時に実行できるような方式で、インスタンスを定義する必要があります。実際のインストール要件およびインスタンスの作成に関する詳細については、 *HACMP for AIX, Version 4.2: Installation Guide* (SC23-1940) を参照してください。

例

以下の例ではそれぞれ、サンプル・スクリプトが、(DB2 (AIX 版) のインストールの) `sqllib/samples/hacmp` の中に格納されています。

相互 DB2 インスタンス・フェールオーバー

次に示す相互インスタンス・フェールオーバー・シナリオは、 ″node10″ および ″node20″ と呼ばれる 2 つのプロセッサーからなる HACMP システムによって構成されています。

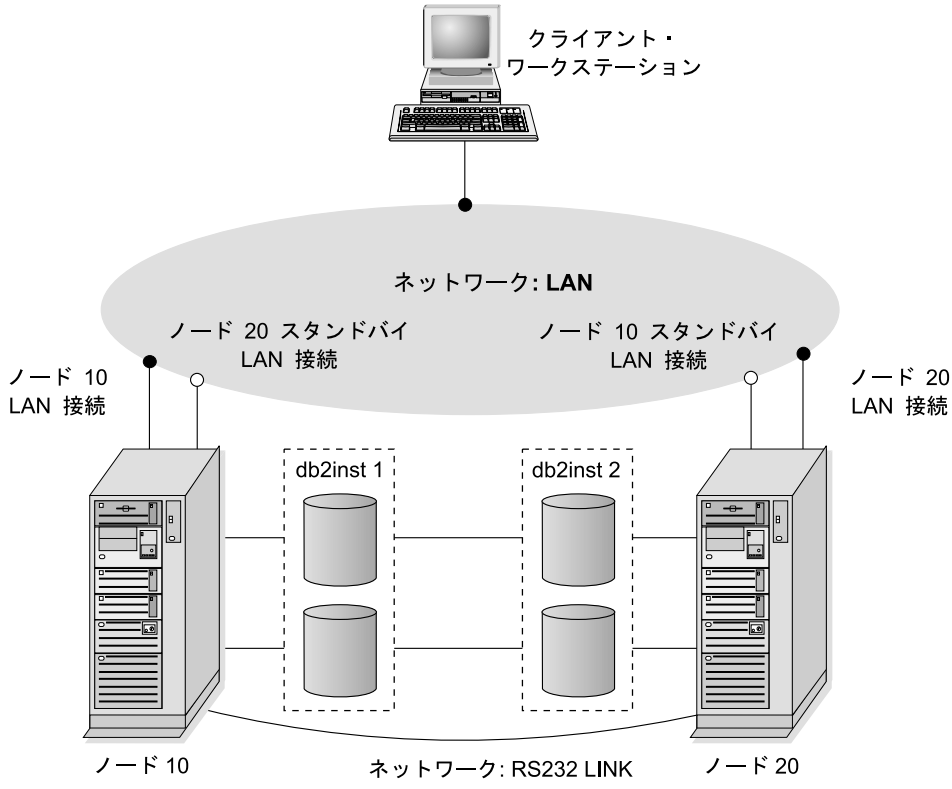


図 45. 相互インスタンス・フェールオーバー構成の例

2 つのインスタンス ″db2inst1″ と ″db2inst2″ は、共用ファイル・システムの単一のインストール・パスから作成されます。インスタンス ″db2inst1″ は/u/db2inst1 で作成され、インスタンス ″db2inst2″ は /u/db2inst2 で作成されます。これらのパスはどちらも、両方のプロセッサーでアクセスできる共用ファイル・システム上にあります。インスタンスはそれぞれ、固有のパスをもつ単一データベースをもっており、やはり、両方のプロセッサーでアクセスできる共用リソース上に置かれています。

どちらのインスタンスも、リモート・クライアントから TCP/IP プロトコルでアクセスされます。 ″db2inst1″ は、サービス名 ″db2inst1_port″ (ポート番号 5500) を使用し、 ″db2inst2″ は、サービス名 ″db2inst2_port″ (ポート番号 5550) を使用します。 ″db2inst1″ インスタンスにアクセスするリモート・クライアントは、 ″node10″ をホスト名として使用して、それらのノード・ディレクトリーにこのインスタンスをカタログしてもっています。 ″db2inst2″ インスタンスにアクセスするリモート・クライアントは、 ″node20″ をホスト名として使用して、それらのノード・ディレクトリーにこのインスタンスをカタログ

してもっています。通常の操作条件では、″db2inst1″ は ″node10″ 上で実行され、 ″db2inst2″ は ″node20″ 上で実行されます。 ″node10″ が故障した場合には、フェールオーバー・スクリプトが ″db2inst1″ を ″node20″ で開始し、 ″node10″ と関連する外部 IP アドレスが ″node20″ に切り換えられます。フェールオーバー・スクリプトによってインスタンスが開始され、データベースが再始動されると、リモート・クライアントは、あたかも ″node10″ で実行されているように、このインスタンスの内部のデータベースに接続することができます。

サンプル・スクリプト:

```
hacmp-s4.sh
```

相互 DB2 区画フェールオーバー

区分データベース・サーバー環境での区画の相互フェールオーバーでは、フェールオーバー・プロセッサー上の論理ノードとして、区画のフェールオーバーが起こるようにする必要があります。たとえば、相互引き受け用に構成された 2 つのプロセッサーからなる HACMP クラスターの別個のプロセッサーで実行されている区分データベース・サーバーに 2 つの区画がある場合は、それらの区画は論理ノードとしてフェールオーバーする必要があります。それぞれのノードの省略時の区画は、論理ノード 0 として定義されている必要があります。このことは、あるプロセッサーから別のプロセッサーに区画がフェールオーバーするとき、その区画は直接のリモート通信プロトコルの listener をもたない論理ノードとして開始されることを意味しています。そのような区画を調整プログラム・ノードとして使用することはできません。

相互区画引き受けとしてシステムを構成するときに、ほかに考慮すべき 1 つの重要な事項は、ローカル区画データベース・パスに関することです。区分データベース環境でデータベースが作成されるときは、複数の区分データベース・サーバーによって共用されないルート・パス上に作成されます。たとえば、次のコマンドを考えてみます。

```
CREATE DATABASE db_a1 ON /dbpath
```

このコマンドはインスタンス ″db2inst″ で実行され、データベース db_a1 を /dbpath で作成します。それぞれのデータベース区画は、/dbpath/db2inst/node*xxxx* の下にあるローカルのファイル・システム上に作成されます。ここで、*xxxx* は、ノード番号を表します。 HACMP フェールオーバーでは、他のプロセッサーですでに使用されている /dbpath ファイル・システムをマウントしようとします。したがって、フェールオーバー・スクリプトは、異なる論理ポイントの下にファイル・システムをマウントして、そのファ

イル・システムから該当の /dpath/db2inst/node*xxxx* パスへのシンボリック・リンクをセットアップする必要があります。

以下に、フェールオーバーの前と後の db2nodes.cfg ファイルの一部を示します。この例では、ノード番号 2 が HACMP マシンのプロセッサー 1 で実行されています。このマシンはホスト名とネット名の両方が "node201" です。ノード番号 3 が HACMP マシンのプロセッサー 2 で実行されています。このマシンはホスト名とネット名の両方が "node202" です。

```
前:
        1 node101 0 node101
        2 node201 0 node201    <= HACMP
        3 node202 0 node202    <= HACMP
        4 node301 0 node301
   db2start nodenum 2 restart hostname node202 port 1 netname node202
後:
        1 node101 0 node101
        2 node202 1 node202    <= HACMP
        3 node202 0 node202    <= HACMP
        4 node301 0 node301
```

フェールオーバーのあとで、調整プログラムとしてのノード番号 2 に直接アクセスしようとしているリモート・クライアントは、データベースのノード・エントリーを再カタログして、フェールオーバー・ノードを指し示すようにする必要があります。調整プログラム・ノードで相互フェールオーバー・シナリオを使用することはお勧めしません。ご使用の調整プログラム・ノードで冗長度が必要な場合は、ホット・スタンドバイ構成を使用してください。

サンプル・スクリプト:

```
hacmp-s5.sh
```

フェールオーバーの後の再接続

クライアントが SET CLIENT ステートメントを使用して特定のノードに接続した場合、そのノードがフェールオーバー中に異なるホストに移動すると、そのクライアントからの次の接続要求は失敗します。 SET CLIENT ステートメントが実行されているノードで、 **db2stop** と **db2start** *nodenum* を続けて出してから、 SET CLIENT ステートメントを再び発行し、クライアントとサーバーの両方がターゲット・ノードの物理的な位置を検出するようにします。

リソース

HACMP の概念、インストール、および構成の詳細については、以下の資料を参照してください。

- *HACMP for AIX V4.2 概念および諸機能* (SC88-6912)
- *HACMP for AIX, Version 4.2: Installation Guide* (SC23-1940)
- *HACMP for AIX V4.2 プランニング・ガイド* (SC88-6913)

第4部 高可用性

第12章 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)

拡張スケーラビリティー (ES) は、 High Availability Cluster Multi-processing (HACMP) for AIX Version 4.2.2 の機能で、現在 RS/6000 SP ノードでのみ稼動します。

この機能は、HACMP と同じフェールオーバー回復を提供し、以前の HACMP バージョンと同じ事象構造を持ちます (*HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide* を参照してください)。また、拡張スケーラビリティーは以下のものを提供します。

- より大きな HACMP クラスター (クラスターごとに最大 16 ノードのスケーラビリティーを持つ)。
- ユーザー定義事象 による追加エラー保守。モニター・エリアでユーザー定義事象が生成される可能性があり、その場合には処理が停止したり、ページ・スペースが容量の限界に近づいたりすることがあります。そのような事象には事象前と事象後があり、必要に応じて、フェールオーバー回復処理に追加できます。 HACMP の事象前ストリームと事象後ストリームの中で、異なる実装に特有の追加機能を加えることができます。

  規則ファイル (`/usr/sbin/cluster/events/rules.hacmprd`) には、 HACMP 事象が含まれています。ユーザー定義事象はこのファイルに追加されます。その定義の一部として、事象発生時に実行されるスクリプト・ファイルがあります。ユーザー定義事象および規則ファイルについて詳しくは、 261ページの『HACMP ES イベント・モニターおよびユーザー定義事象』を参照してください。
- HACMP クライアント・ユーティリティー。 HACMP クラスター外の AIX 物理ノードから (1 つまたは複数のクラスターにおける) 状況変更をモニターし、検出します。

HACMP ES クラスター内のノードは、ハートビート またはキープアライブ・パケット というメッセージを交換して、自身の可用性についての情報を他のノードに通知します。応答しなくなったノードがあると、クラスター内の残りのノードは回復を呼び出します。回復処理は *node_down 事象* と呼ばれ、フェールオーバー ということもあります。回復処理が完了すると、ノードをクラスターに再統合されます。この処理を *node_up 事象* といいます。

事象には 2 つのタイプ、つまり、HACMP ES の操作中に予期される標準事象と、ハードウェアおよびソフトウェア構成要素のモニターに関連したユーザー定義事象があります。

標準事象の 1 つに、node_down 事象があります。回復処理の一部として実行する処理を計画する場合、 HACMP では 2 つのフェールオーバー・オプション、つまり「ホット (またはアイドル) スタンドバイ」と「相互引き受け」を指定できます。

クラスター構成

ホット・スタンドバイ 構成では、引き受けノードではない AIX プロセッサー・ノードは他の作業負荷を一切実行していません。*相互引き受け* 構成では、引き受けノードである AIX プロセッサー・ノードは他の作業負荷を実行しています。

一般に、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 (UDB EEE) は、各ノード上の区画で相互引き受けモードで実行されます。1 つの例外として、カタログ・ノード部品がホット・スタンドバイ構成の一部となるシナリオがあります。

HACMP ES を使用する RS/6000 SP で大規模な DB2 インストールを計画する場合は、 RS/6000 SP フレームの内部またはフレーム間でクラスターのノードを分割する方法を考慮する必要があります。異なる SP フレームにノードおよびそのバックアップを指定すると、 1 つのフレームがダウン (つまり、フレームの電源 / スイッチ・ボードが故障) するという事象でも引き受けが可能です。しかし、こうした故障は非常にまれだと考えられます。各 SP フレームには *N*+1 の電源機構があり、各 SP スイッチには *N*+1 のファンと電源を含む冗長パスがあるからです。フレームが故障すると、手作業で介入して、残りのフレームを回復しなければならない場合があります。この回復手順については、SP 管理の手引きで説明されています。 HACMP ES では、SP ノード障害を回復できます。その場合、フレーム障害の回復は、 1 つまたは複数の SP フレーム内のクラスターの適正なレイアウトに依存しています。

計画時に考慮すべき別の点として、大きなクラスターの管理方法があります。大きなクラスターよりも小さなクラスターのほうが管理は容易ですが、多くの小さなクラスターよりも 1 つの大きなクラスターのほうが管理はやはり容易です。計画時には、実際のアプリケーションがクラスター環境で使用される方法を考慮してください。たとえば、16 ノードで単一の大きな同類のアプリケーションが実行されていれば、構成を 8 個の 2 ノード・クラスターではなく、単一のクラスターとして管理するほうが容易でしょう。同じ 16 ノードでも、異

なるネットワーク、ディスク、およびノード関係を持つ多数の異なるアプリケーションが含まれているのであれば、ノードを小さめのクラスターにグループ化したほうが得策と思われます。各ノードは一度に 1 つずつ HACMP クラスターに統合される点に留意してください。つまり、1 つの大きなクラスターよりも複数のクラスター構成のほうが始動速度は向上します。 HACMP ES は、ノードおよびそのバックアップが同じクラスターにある限り、単一のクラスターも複数のクラスターもサポートします。

HACMP ES フェールオーバー回復では、物理ノードにリソース・グループを事前定義 (*カスケード* ともいう) によって割り当てることができます。フェールオーバーの回復手順では、物理ノードにリソース・グループを浮動 (*回転* ともいう) によって割り当てることもできます。 IP アドレス、外部ディスク・ボリューム・グループ、ファイル・システム、 NFS ファイル・システム、および各リソース・グループ内のアプリケーション・サーバーは、アプリケーションまたはアプリケーション構成要素のいずれかを指定します。これは、フェールオーバーまたは再統合によって、物理ノード間で HACMP ES が操作するものです。フェールオーバーおよび再統合の操作は、作成されるリソース・グループのタイプ、およびリソース・グループに配置されるノード数によって指定されます。

たとえば、DB2 データベース区画 (論理ノード) を考慮してみます。そのログと表スペースのコンテナーが外部ディスクに置かれ、他のノードがそのディスクにリンクされていた場合、それら他のノードは外部ディスクにアクセスし、(引き受けノード上にある) データベース区画を再始動することができます。HACMP では、このような操作が自動化されます。 HACMP ES は、 DB2 インスタンスの主要なユーザー・ディレクトリーが使用する NFS ファイル・システムを回復する場合にも使用できます。

DB2 UDB EEE の回復を計画する場合には、その一環として HACMP ES 資料を精読してください。概念、計画、インストール、管理の手引きに目を通した後、環境に合った回復アーキテクチャーを構築してください。知られている障害点に基づいて識別した、回復を要する各サブシステムについては、必要な HACMP クラスターと回復ノード (ホット・スタンドバイまたは相互引き受け) を識別してください。これは、資料にある HACMP ワークシートを完成させる上で開始点となります。

ディスクとアダプターはどちらも、外部ディスク構成でミラーリングすることをぜひお勧めします。 HACMP 用に構成する DB2 物理ノードの場合、ボリューム・グループ上のノードを共用外部ディスクから構成変更できるように配慮する必要があります。相互引き受け構成でこのような設定を行う場合、対になったノードが競合することなく互いのボリューム・グループにアクセスできる

よう、いくらか余分の計画を立てることが必要です。 DB2 UDB EEE 内では、すべてのデータベース間でコンテナー名をすべて固有のものにしておく必要があります。

固有のものにする 1 つの方法は、名前の一部に区分番号を含めることです。SMS または DMS コンテナーを作成するときに、コンテナーのストリング構文にノード式を指定できます。式を指定するときは、ノード番号をコンテナー名の一部とすることができます。また、追加の引き数を指定する場合は、これらの引き数の結果をコンテナー名の一部とすることができます。ノード式を指定するには、引き数 「 $N」([ブランク]$N) を使用します。引き数は必ずコンテナー・ストリングの最後に指定するようにし、以下のいずれかの形式だけを使用できます。

表 24. コンテナーを作成するための引き数. ノード番号を 5 と仮定します。

構文	例	値
[ブランク]$N	「 $N」	5
[ブランク]$N+[番号]	「 $N+1011」	1016
[ブランク]$N%[番号]	「 $N%3」	2
[ブランク]$N+[番号]%[番号]	「 $N+12%13」	4
[ブランク]$N%[番号]+[番号]	「 $N%3+20」	22

注:

1. % はモジュラスです。
2. どの場合でも、演算子は左から右に向かって評価されます。

次に、この特殊な引き数を使用してコンテナーを作成する方法の例をいくつか示します。

- 2 ノード・システムで使用するためのコンテナーの作成。

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
   (device '/dev/rcont $N' 20000)
```

次のコンテナーが使用されます。

```
/dev/rcont0   - on Node 0
/dev/rcont1   - on Node 1
```

- 4 ノード・システムで使用するためのコンテナーの作成。

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
   (file '/DB2/containers/TS2/container $N+100' 10000)
```

次のコンテナーが使用されます。

```
/DB2/containers/TS2/container100   - on Node 0
/DB2/containers/TS2/container101   - on Node 1
/DB2/containers/TS2/container102   - on Node 2
/DB2/containers/TS2/container103   - on Node 3
```

- 2 ノード・システムで使用するためのコンテナーの作成。

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
   ('/TS3/cont $N%2, '/TS3/cont $N%2+2')
```

次のコンテナーが使用されます。

```
/TS3/cont0   - on Node 0
/TS3/cont2   - on Node 0
/TS3/cont1   - on Node 1
/TS3/cont3   - on Node 1
```

244ページの図46 および 245ページの図47 は、 DB2 SSA I/O サブシステム構成の例を示しています。また、高可用性外部ディスク構成、および競合を起こさずにすべてのボリューム・グループにアクセスする機能の両方を実現するために必要な計画の一部を示しています。

DB2 SSA I/O サブシステム構成 – 単一障害点がない

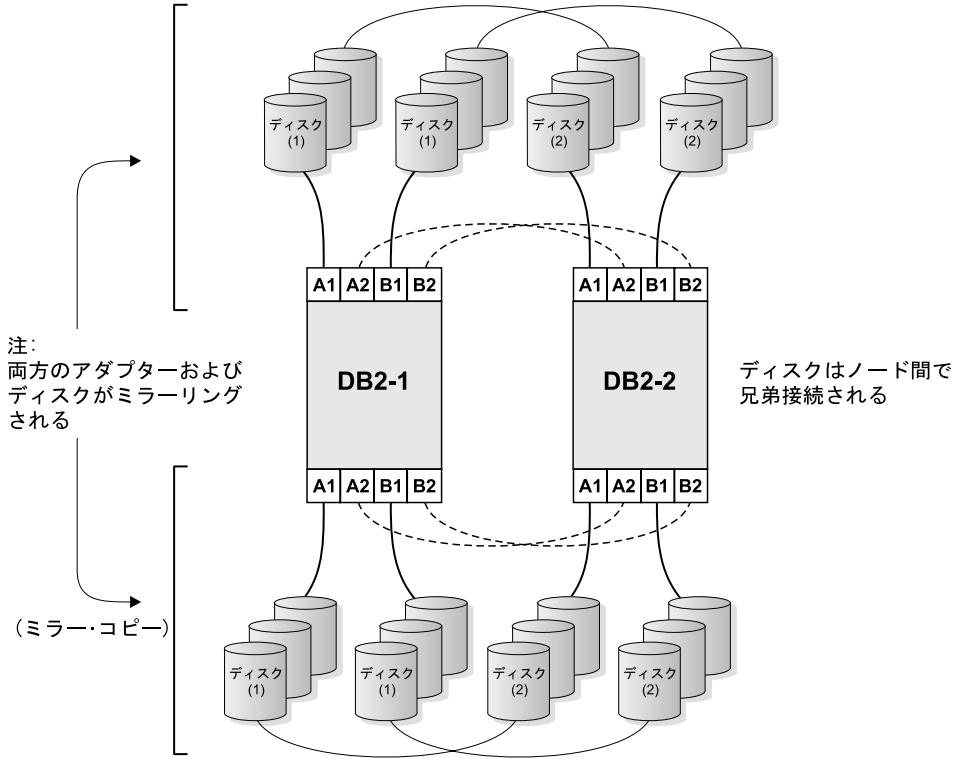


図 46. 単一の障害点がない

DB2 SSA I/O サブシステム構成 -
ボリューム・グループおよび論理ボリュームのセットアップ

ファイル・システム / データベースのインスタンス名 powertp における DB2 データベース・テストデータ

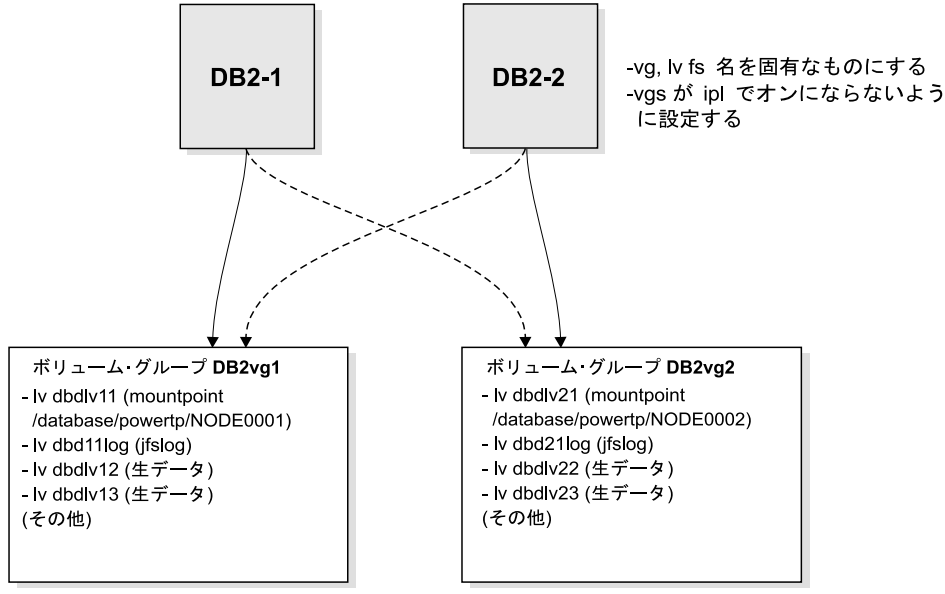


図 47. ボリューム・グループおよび論理ボリュームのセットアップ

DB2 データベース区画の構成

構成が済むと、インスタンス中の各データベース区画は HACMP ES によって物理ノードごとに始動されます。 4 ノードよりも大きい並列 DB2 構成を始動する場合は、複数のクラスターをお勧めします。 64 ノードの並列 DB2 構成では、 4 個の 16 ノード・クラスターよりも、 32 個の 2 ノード HACMP クラスターのほうが始動速度が向上することに注意してください。

スクリプト・ファイル `rc.db2pe` は、ホット・スタンドバイまたは相互引き受けノードのいずれかで HACMP ES のフェールオーバーまたは回復を構成するための補助機構として、 DB2 UDB EEE にパッケージされています (`/usr/bin` の各ノードにインストールされます)。また、`rc.db2pe` の内部から、フェールオーバーまたは相互引き受け構成で DB2 バッファー・プールのサイズをカスタマイズできます。 (バッファー・プール・サイズは、 1 つの物

理ノードで 2 つのデータベース区画を稼働する場合に適切なパフォーマンスを確保できるように構成する必要があります。)

DB2 データベース区画の HACMP 構成でアプリケーション・サーバーを作成する場合、次のようにして rc.db2pe を開始および停止スクリプトとして指定します。

```
/usr/bin/rc.db2pe <instance> <dpn> <secondary dpn> start <use switch>
/usr/bin/rc.db2pe <instance> <dpn> <secondary dpn> stop <use switch>

ここで、各パラメーターは以下のとおりです。

<instance> はインスタンス名。
<dpn> はデータベース区画番号。
<secondary dpn> は、相互引き受け構成のみにおける
    'コンパニオン' データベース区画番号。
    ホット・スタンバイ構成では、<dpn> と同じ。
<use switch> は通常はブランク。ブランクの場合は、
    SP switch ネットワークが db2nodes.cfg ファイルのホスト名 フィールドに使用される。
    (DB2 のためのすべてのトラフィックが SP スイッチを介して経路指定される。)
    ブランクでない場合、ここで使用する名前は、使用される SP ノードのホスト名。
```

このデータベース区画用に構成したすべてのデータベースを検索するには、rc.db2pe 内から DB2 コマンド LIST DATABASE DIRECTORY を使用します。すると、スクリプト・ファイルは、 /usr/bin/reg.parms.*DATABASE* ファイル、および /usr/bin/failover.parms.*DATABASE* ファイルを探します。ただし、*DATABASE* はこのデータベース区画用に構成された個々のデータベースを表します。相互引き受け構成では、パラメーター・ファイル reg.parms.*xxx* および failover.parms.*xxx* を作成することをお勧めします。failover.parms.*xxx* ファイルでは、複数のバッファー・プールがある可能性を考慮して、 BUFFPAGE、DBHEAP、およびバッファー・プールに影響を与える他のパラメーターの設定を調整します。サンプル・ファイル reg.parms.SAMPLE および failover.parms.SAMPLE は、ユーザーが使用するために用意されました。

この環境で重要なパラメーターの 1 つは、 *start_stop_time* データベース・マネージャー構成パラメーターです。このパラメーターの省略時値は 10 分です。ただし、rc.db2pe を指定すると、このパラメーターは 2 分に設定されます。このパラメーターは rc.db2pe で調整して、 10 分か、もう少し大きめの値に設定するとよいでしょう。このコンテキストでは、指定した時間は、区画の障害発生からその区画の回復までの時間間隔です。区画上で実行されているアプリケーションが頻繁に COMMIT を出す場合、データベース区画上の障害後、10 分もあれば、コミットされていないトランザクションをロールバックし、その区画上のデータベースの一貫性ポイントに到達することが可能です。

作業負荷が重いか、または多くの区画がある場合、ロールバック操作が完了する前にタイムアウトになってしまう可能性を少なくするために、時間を増やす必要があるかもしれません。

以下は、ホット・スタンドバイ構成および相互引き受け構成の例です。どちらの例でも、リソース・グループにはサービス IP スイッチ別名アドレスが入っています。このスイッチ別名アドレスは、以下の目的で使用されます。

- DB2 インスタンス所有者ファイル・システムを使用するために行うファイル・サーバーへの NFS アクセス。
- フェールオーバー、TSM (Tivoli Storage Manager、以前の ADSM) 接続、または他の同様の操作で管理が必要となる他のクライアント・アクセス。

実際の設定でこれらの別名が必要でなければ、削除しても構いません。削除した場合は、 `rc.db2pe` スクリプト・ファイルで *MOUNT_NFS* パラメーターを `NO` に設定してください。

ホット・スタンドバイ構成の例

この例では、ホット・スタンバイ構成がノード 1 とノード 2 の間にあり、DB2 インスタンス名が POWERTP であると仮定します。データベース区画は 1 で、データベースはファイル・システム `/database` 上の TESTDATA です。

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1     (<<< this is the switch alias address)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 1 stop
```

相互引き受け構成の例

この例では、相互引き受け構成がノード 1 とノード 2 の間にあり、 DB2 インスタンス名が POWERTP であると仮定します。データベース区画は 1 および 2 で、データベースはファイル・システム `/database` 上の TESTDATA です。

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1     (<<< this is the switch alias address)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 2 start
```

```
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 2 stop

Resource group name: db2_pd_2
Node Relationship: cascading
Participating nodenames: node2_eth, node1_eth
Service_IP_label: nfs_switch_2     (<<< this is the switch alias address)
Filesystems: /database/powertp/NODE0002
Volume Groups: DB2vg2
Application Servers: db2_dp2_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 2 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 2 1 stop
```

NFS サーバー・ノードの構成

また、rc.db2pe スクリプトを使用して、 DB2 並列インスタンス・ユーザー・ディレクトリーの NFS マウント・ディレクトリーが使用可能になります。これは、 rc.db2pe スクリプト・ファイルで *MOUNT_NFS* パラメーターを YES に設定し、 NFS フェールオーバー・サーバーのペアを以下のように構成することによって実行できます。

- ホーム・ディレクトリーを構成し、「ルート」として、 /etc/exports および **exportfs** コマンドを使用して、このディレクトリーをエクスポートします。エクスポート先は、 NFS サーバーの IP アドレスと同じサブネット内のノードで使用する IP アドレスとします。 HACMP ブートおよびサービス・アドレスの両方を指定します。 NFS サーバーの IP アドレスは、バックアップで引き受け可能な HACMP 内のサービス・アドレスと同じアドレスです。 DB2 インスタンス所有者のホーム・ディレクトリーには、自動マウントではなく、 NFS マウント・ディレクトリーを指定してください。 (スクリプトは、DB2 インスタンス所有者ホーム・ディレクトリーとして、自動マウント・プログラムの使用をサポートしていません。)
- このファイル・システムでは、SMIT または実用的な構成を使用して別個の /etc/filesystems 項目を作成し、 DB2 並列グループ化ですべてのノード (ファイル・サーバーを含む) が NFS ファイル・システム・コマンドでマウントできるようにします。

  たとえば、/nfshome JFS ファイル・システムは、 /dbhome としてすべてのノードにエクスポートできます。各ノードは、 nfs_server:/nfshome である NFS ファイル・システム /dbname を作成します。したがって、インスタンス名が「powertp」の場合、 DB2 インスタンス所有者のホーム・ディレクトリーは /dbhome/powertp となります。

  /etc/filesystems のマウント用 NFS パラメーターが、「hard」、「bg」、「intr」、「rw」であることを確かめてください。

- /etc/passwd にあるホーム・ディレクトリー /dbhome/powertp と関連付けられた DB2 インスタンス所有者定義が、すべてのノードで同じものであることを確認します。

  SP 環境内のユーザー定義は普通、コントロール・ワークステーションで作成され、「supper」または「pcp」は、/etc/passwd、/etc/security/passwd、 /etc/security/user、および /etc/security/group をすべてのノードに配布するために使用します。
- HACMP リソース・グループで、ボリューム・グループおよびエクスポートされるファイル・システムに対して「エクスポートする nfs_filesystems」を構成しないでください。代わりに、NFS に対して通常どおりに構成します。NFS サーバーのスクリプトは、ファイル・システムのエクスポートを制御します。
- ファイル・システムがあるボリューム・グループの主要な番号が、 1 次ノードでも引き受けノードでも同じ番号であることを確かめてください。この確認は、 **importvg** に -V オプションを指定して行います。
- rc.db2pe で *MOUNT_NFS* パラメーターが YES に設定され、/etc/filesystems にマウントする NFS ファイル・システムが各ノードにあることを検査します。もしなければ、 rc.db2pe がファイル・システムをマウントして DB2 を始動することはできません。
- DB2 インスタンス所有者がすでに作成されており、作成するファイル・システムにユーザーのディレクトリー構造をコピーするのであれば、必ずディレクトリーを **tar** (-cvf) してください。これにより、シンボリック・リンクが確実に保存されます。
- 作成するファイル・システムの論理ボリュームとファイル・システム・ログについて、アダプターとディスクの両方を必ずミラーリングしてください。

NFS サーバー引き受け構成の例

この例では、 IP アドレス「nfs_server」を介してボリューム・グループ nfsvg に NFS サーバー・ファイル・システム /nfshome があることを想定しています。 DB2 インスタンス名は POWERTP で、ホーム・ディレクトリーは /dbhome/powertp です。

```
Resource group name: nfs_server
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_server     (<<< this is the switch alias address)
Filesystems: /nfshome
Volume Groups: nfsvg
Application Servers: nfs_server_app
Application Server Start Script: /usr/bin/rc.db2pe powertp NFS SERVER start
Application Server Stop Script: /usr/bin/rc.db2pe powertp NFS SERVER stop
```

この例では、以下のようになります。

- 全ノード上の /etc/filesystems には、マウントする nfs_server:/nfshome として /dbhome の項目が入っています。 nfs_server は サービス IP スイッチ別名アドレスです。
- nfs_server ノードおよびバックアップ・ノード上の /etc/exports には、ブートおよびサービス・アドレスが入っており、 /nfsfs -root=nfs_switch_1、nfs_switch_2、 ... の項目が含まれています。

SP スイッチ構成時の考慮事項

HACMP ES に SP スイッチを実装する場合は、以下の事柄を考慮します。

- SP スイッチには、「基底」アドレスおよび「別名」アドレスがあります。基底アドレスは SP システム・データ・リポジトリー (SDR) で定義されたアドレスであり、システムが「ブートされる」ときに rc.switch で定義されます。別名アドレスは、別名属性を指定した ifconfig コマンドにより、基底アドレスに加えて **css0** インターフェースへ構成された IP アドレスです。たとえば、次のようなものがあります。

  ```
  ifconfig css0 inet alias sw_alias_1 up
  ```

- DB2 db2nodes.cfg ファイルを構成するときは、「hostname」フィールドでも「netname」フィールドでも、 SP スイッチ「基底」IP アドレスを使用しなければなりません。スイッチ IP アドレスの別名は、 NFS 接続性を維持することだけ を目的として使用します。 DB2 フェールオーバーは、 **db2start** (RESTART) コマンド (db2nodes.cfg を更新する) を使用して DB2 を再始動することにより実行します。
- スイッチ・アドレスと etc/hosts 別名を混同しないでください。 SP スイッチ・アドレスも SP スイッチ別名アドレスも、 etc/hosts または DNS のいずれかに実在します。スイッチ別名アドレスは、SP スイッチ基底アドレスの別名ではありません。どちらのアドレスにも、固有の別個のアドレスがあります。
- 活動時のノードには常に、SP スイッチ基底アドレスがあります。 HACMP ES は、ノード間でこれらのアドレスを構成したり移動したりしません。
- SP スイッチ別名アドレスを使用する場合は、「ハートビート」用のブート・アドレスとサービス・アドレスおよび IP アドレス引き受けとして、HACMP に対して別名アドレスを構成します。 SP スイッチ別名アドレスを使用しない場合は、「ハートビート」*専用* のサービス・アドレス (IP アドレス引き受けではない) として、 HACMP に対して基底アドレスを構成しま

す。どのような構成を行う場合でも、別名アドレスとスイッチ基底アドレスの*両方* は構成しないでください。そのような構成は、HACMP ES ではサポートされません。

- SP スイッチ基底アドレスではなく、 SP スイッチ別名アドレスだけが、 IP 引き受け構成のノード間を移動されます。
- ノードあたりの SP スイッチ・アダプターが 1 つだけの場合もあるので、SP スイッチ別名の必要性が生じます。別名アドレスを使用すると、ノードは、別のスイッチ・アダプターを追加しなくても、別のノードのスイッチ別名 IP アドレス を引き受けることができます。この方法は、「スロットに制約がある」ノードで役立ちます。 SP スイッチ・アダプターの障害からの回復方法の詳細については、 265ページの『HACMP ES スクリプト・ファイル』のネットワーク障害のセクションを参照してください。
- IP アドレス引き受け用の SPスイッチを構成する場合、ノードごとに 2 つの余分の IP アドレスが必要になります。 1 つはブート・アドレス、もう 1 つはサービス・アドレスになります。
- SP スイッチ IP アドレスまたは SP スイッチ別名 IP アドレスの HACMP ES ネットワーク名定義では、必ず「HPS」を使用してください。
- HACMP の `rc.cluster` は、 HACMP の始動時に SP スイッチ・ブート・アドレスで自動的に **ifconfig** を実行します。 IP アドレスと名前を作成し、それらを HACMP に定義する以外に、余分の構成は必要とされません。
- SP スイッチの Eprimary ノードは、 Estart、Efence、および Eunfence コマンドを実装するサーバーです。スイッチがネットワークの 1 つとして定義されると、 HACMP スクリプトは、HACMP の始動時にノードに対して Eunfence または Estart を実行し、スイッチを使用可能にしようとします。それで、HACMP を始動するときは Eprimary が使用可能であることを確認してください。 HACMP コードは、Eprimary フェールオーバーがエラー終了する前に完了できるよう、最大で 12 分間待機します。
- ノード間では、HACMP ではなく、 SP AIX 並列システム・サポート・プログラム (PSSP) が SP スイッチの Eprimary ノードを移動します。 Eprimary ノードがオフラインになると、 PSSP は自動的にバックアップ・ノードで Eprimary ノードとしての責任を果たします。スイッチ・ネットワークはこの変更による影響を受けず、活動状態を保ちます。

DB2 HACMP 構成の例

次の例では、異なるフェールオーバー・サポート構成と、障害発生時に生じる事柄を示します。

DB2 HACMP 相互引き受け構成 (253ページの図48、254ページの図49、および255ページの図50) については次のとおりです。

- HACMP アダプターはイーサネット、および SP スイッチ別名のブートおよびサービス別名用に定義され、基底アドレスは未定義のままです。 HACMP ネットワーク名では必ず、「HPS」ストリングを使用してください。
- NFS_server/nfshome は、スイッチ別名を使用して全ノードで /dbhome としてマウントされます。
- db2nodes.cfg ファイルには、 SP スイッチ基底アドレスが入っています。 DB2 データベース区画 (論理ノード) のフェールオーバー後、 db2nodes.cfg ファイルは **DB2START** (RESTART) コマンドで変更されます。
- SP スイッチ別名ブート・アドレスは表示されません。
- ノードは、異なる SP フレームに置くことができます。

NFS フェールオーバーを使用した DB2 HACMP 相互引き受け - 正常

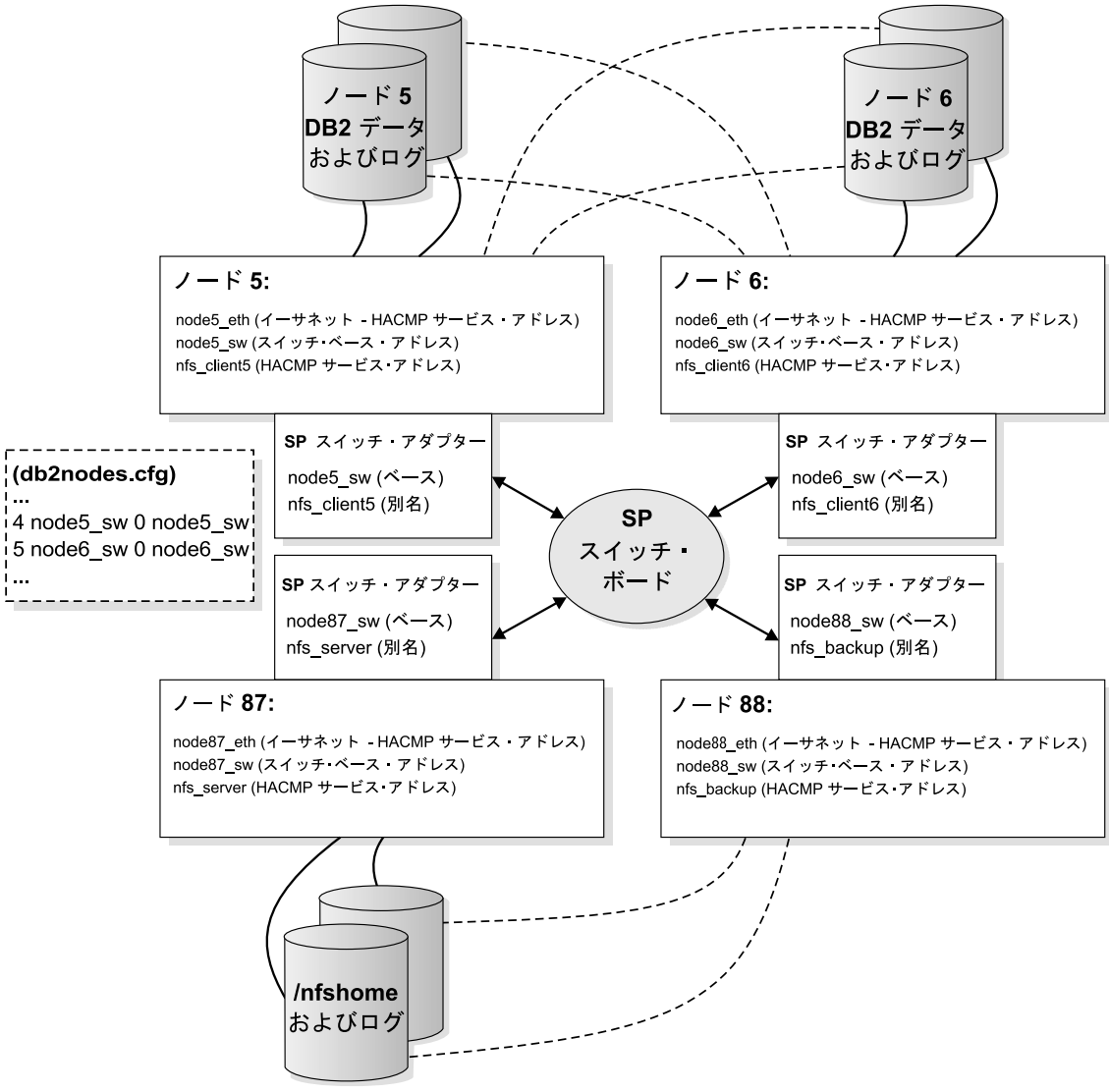


図 48. NFS フェールオーバーを使用した相互引き受け - 正常

NFS フェールオーバーを使用した DB2 HACMP 相互引き受け- NFS フェールオーバー

- nfs_server SP スイッチ別名 IP アドレスと、nfs マウントされた /nfshome がノード 87 から 88 に移動されます。
- SP スイッチ arp コードは、この移動により、すべてのスイッチ arp キャッシュを更新する機能があります。

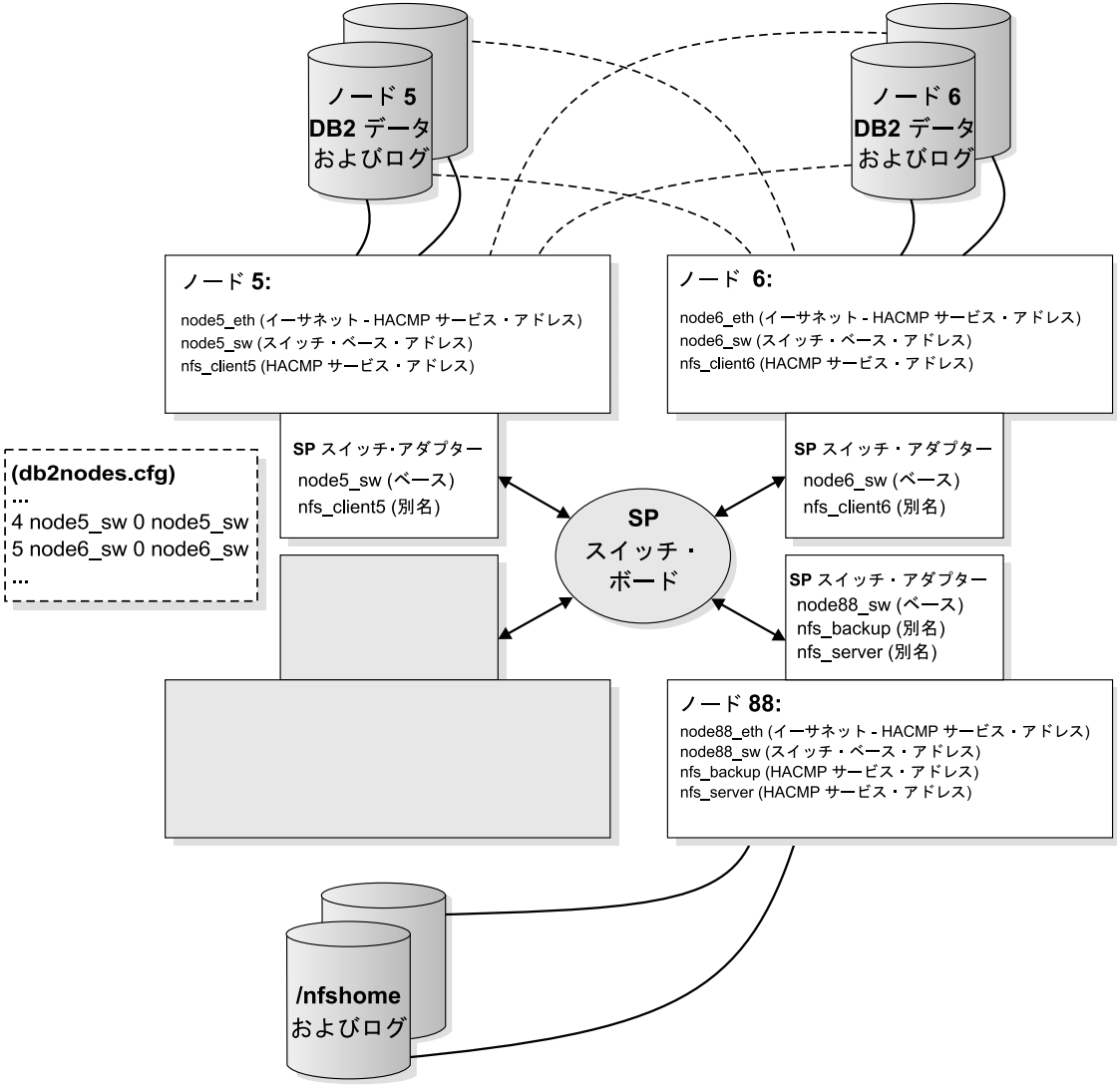


図 49. NFS フェールオーバーを使用した相互引き受け - NFS フェールオーバー

NFS フェールオーバーを使用した DB2 HACMP 相互引き受け - DB2 フェールオーバー

- スイッチ IP アドレス引き受けによって、他のサーバー (ADSM など) が接続を保持できるようになります。
- ノード 5 は、DB2 の 2 つの論理ノードを実行します。

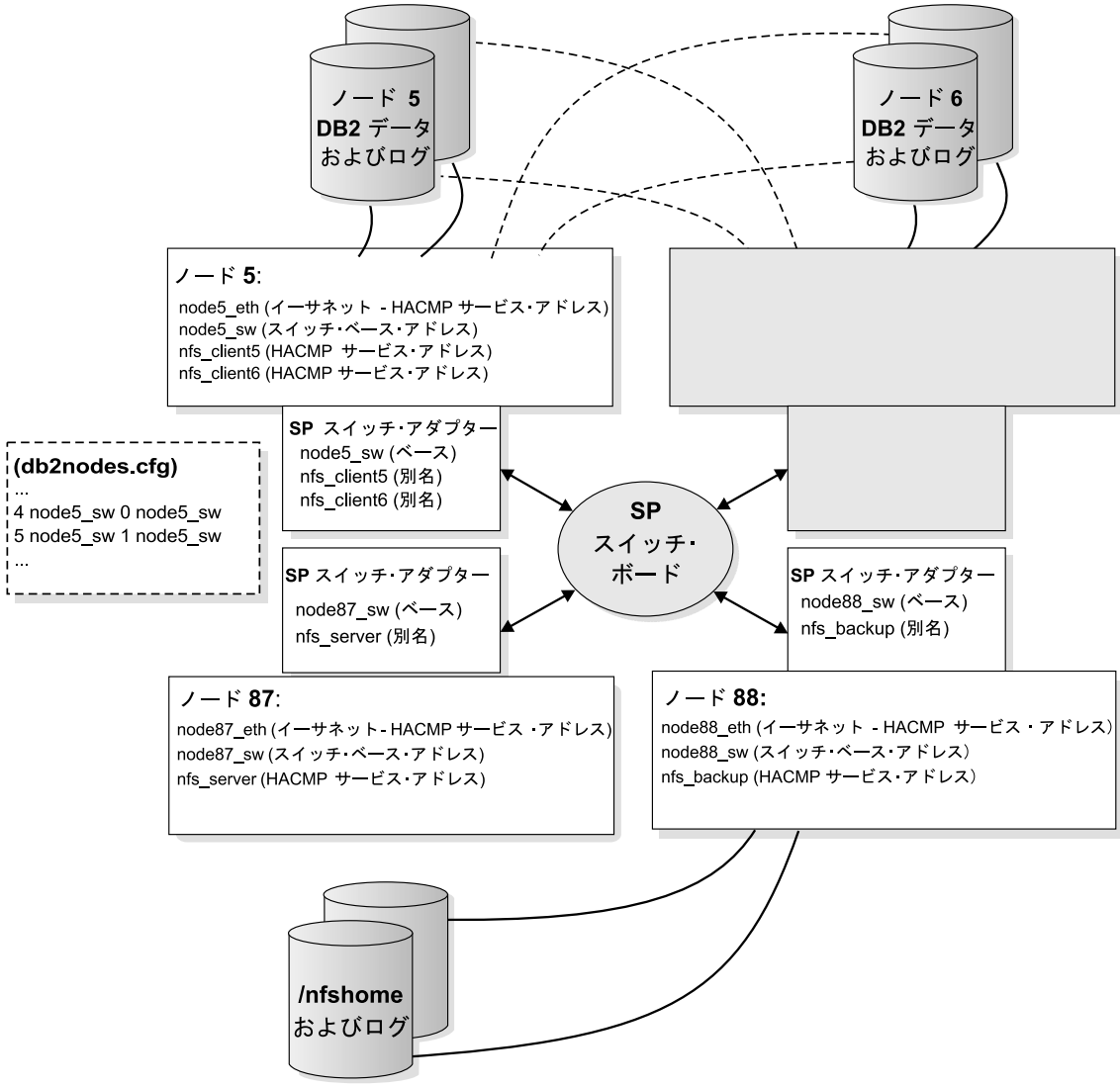


図 50. NFS フェールオーバーを使用した相互引き受け - DB2 フェールオーバー

DB2 HACMP ホット・スタンバイ構成 (257ページの図51および 258ページの図52) については次のとおりです。

- HACMP アダプターはイーサネット、および SP スイッチ別名のブートおよびサービス別名用に定義され、基底アドレスは未定義のままです。 HACMP ネットワーク名では必ず、「HPS」ストリングを使用してください。
- NFS_server/nfshome は、スイッチ別名を使用して全ノードで /dbhome としてマウントされます。
- db2nodes.cfg ファイルには、 SP スイッチ基底アドレスが入っています。 DB2 データベース区画 (論理ノード) のフェールオーバー後、 db2nodes.cfg ファイルは **DB2START** (RESTART) コマンドで変更されます。
- SP スイッチ別名ブート・アドレスは表示されません。

NFS フェールオーバーを使用した DB2 HACMP ホット・スタンドバイ - 正常

注:ホット・スタンドバイ・ノードは、ディスク配線に応じて複数のノードをバックアップできます。

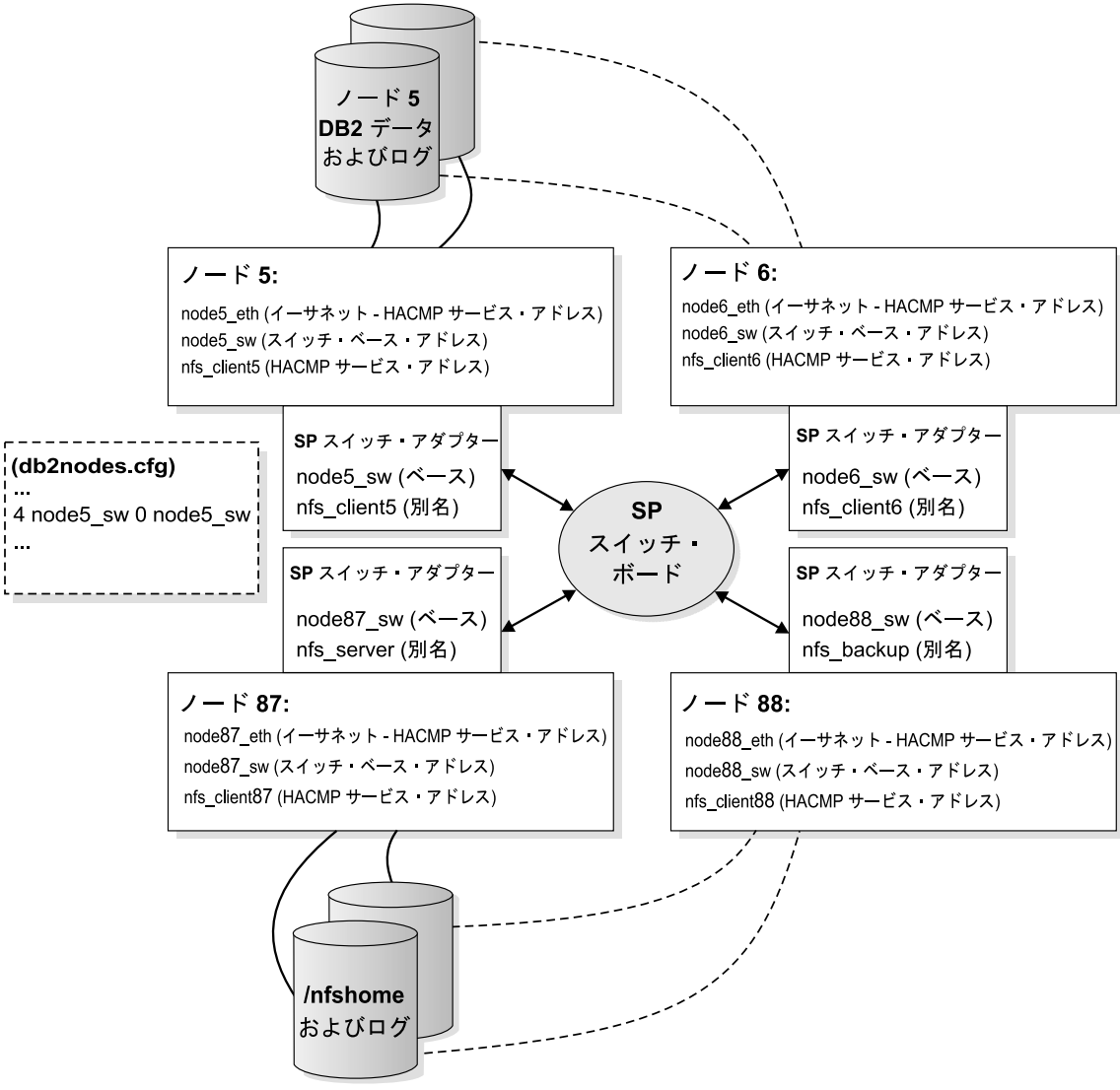


図 51. NFS フェールオーバーを使用したホット・スタンドバイ - 正常

NFS フェルオーバーを使用した DB2 HACMP ホット・スタンバイ - DB2 フェールオーバー

注: ホット・スタンバイ・ノードは、ディスク配線に応じて複数のノードをバックアップできます。

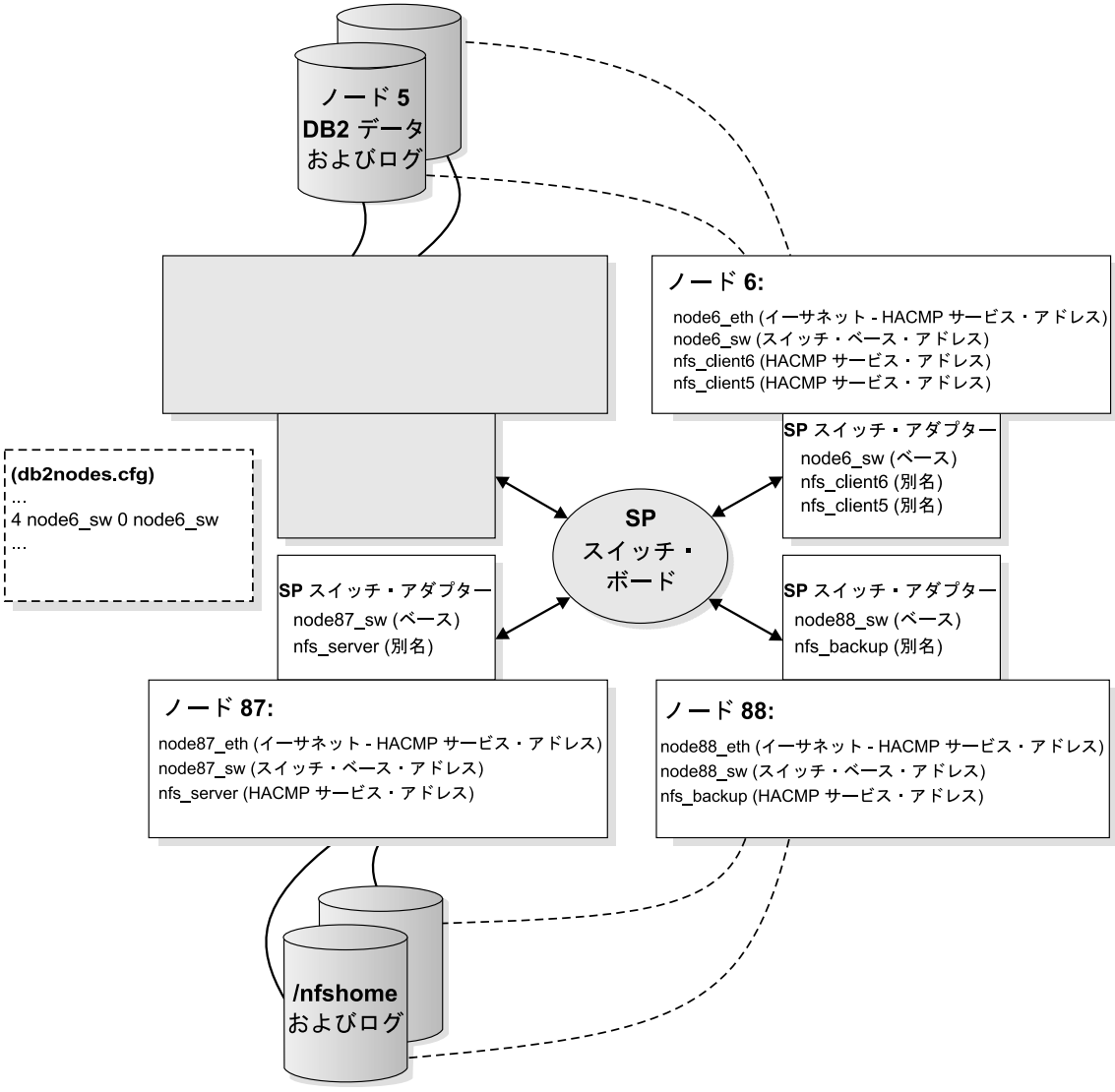


図 52. NFS フェールオーバーを使用したホット・スタンドバイ - DB2 フェールオーバー

NFS フェールオーバー構成を使用しない DB2 HACMP 相互引き受け (259ページの図53 および 260ページの図54) については次のとおりです。

- HACMP アダプターはイーサネット、および SP スイッチ基底アドレス用に定義されています。基底アドレスがサービス・アドレスとして HACMP に構成される場合、ブート・アドレスはない (「ハートビート」のみ) ということを忘れないでください。 SP スイッチには必ず、 HACMP ネットワーク名で「HPS」ストリングを使用してください。
- `db2nodes.cfg` ファイルには、 SP スイッチ基底アドレスが入っています。DB2 データベース区画 (論理ノード) のフェールオーバー後、`db2nodes.cfg` ファイルは **DB2START** (RESTART) コマンドで変更されます。
- NFS フェールオーバーの機能は表示されません。
- ノードは、異なる SP フレームに置くことができます。

NFS フェールオーバーを使用しない DB2 HACMP 相互引き受け - 正常

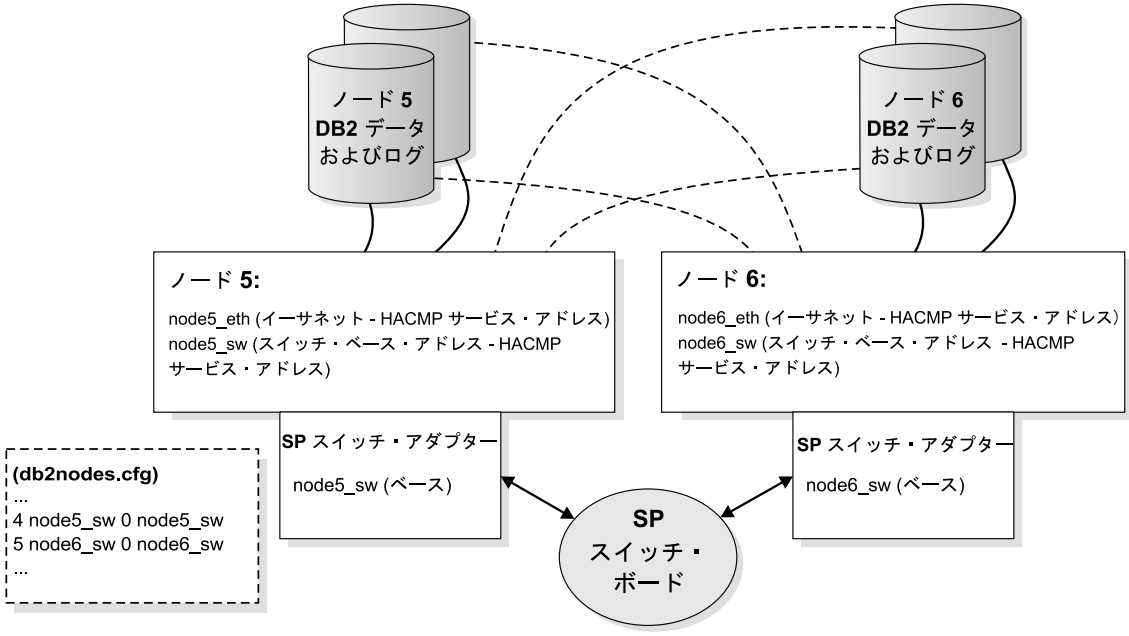


図 53. NFS フェールオーバーを使用しない相互引き受け - 正常

NFS フェールオーバーを使用し DB2 HACMP 相互引き受け - DB2 フェールオーバー

- ノード 5 は DB2 の 2 つの論理ノードを実行します。

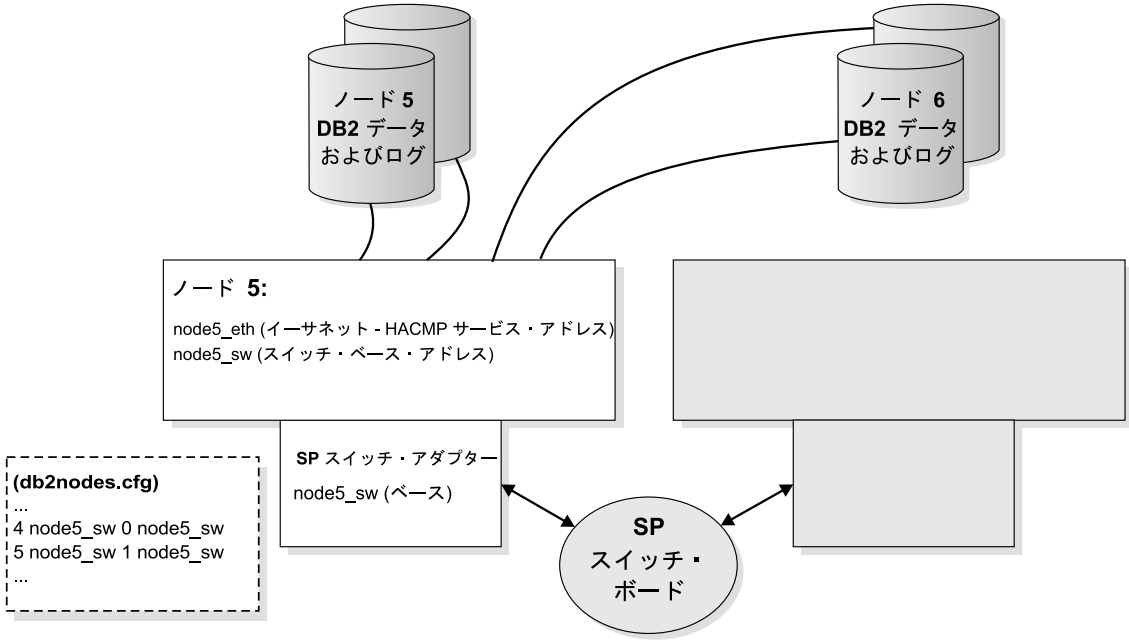


図 54. NFS フェールオーバーを使用しない相互引き受け - DB2 フェールオーバー

DB2 HACMP 始動に関する推奨事項

/etc/inittab でブート時での HACMP の始動を指定しないことをお勧めします。 HACMP は、ノードのブート後に手作業で始動してください。この方法により、障害ノードの破壊的な保守を避けることができます。

「破壊的な保守」の例として、ハードウェアの障害と破損が生じたノードについて考慮してみます。フェールオーバーは HACMP により自動的に開始され、回復は正常に完了します。しかし、障害ノードは、修復する必要があります。HACMP がリブート時に始動するように /etc/inittab で構成された場合、このノードはブート完了後に再統合を試みますが、それはこのような状況では望ましいことではありません。

「非破壊的な保守」として、各ノードで HACMP を手作業で始動する場合を考慮してみましょう。手作業での始動により、他のノードに影響を与えないで障

害ノードを修正し、再統合することができます。コントロール・ワークステーションから HACMP の始動および停止コマンドを制御するには、 ha_cmd スクリプトを使用できます。

注: はじめて DB2 インスタンスを作成するときは、次の項目が /etc/inittab ファイルに付加されます。

```
rcdb2:2:once:/etc/rc.db2 > /dev/console 2>&1 # Autostart DB2 Services
```

HACMP または HACMP ES が使用可能になっている場合、 HACMP 項目の前に上記の行を置き、 /etc/inittab ファイルを更新してください。以下は、/etc/inittab ファイル内のサンプル HACMP 項目です。

```
clinit:a:wait:touch /usr/sbin/cluster/.telinit # HACMP for AIX
```

この項目は、 /etc/inittab ファイル内の最後の項目でなければなりません。

HACMP ES イベント・モニターおよびユーザー定義事象

ユーザー定義事象の例には、ページング・スペースが一定の満了状態に達するときに、 AIX 物理ノードで DB2 データベース区画をシャットダウンする事象や、特定のノードで処理が停止する場合に、 DB2 データベース区画を再始動したりフェールオーバー操作を開始したりする事象が挙げられます。データベース区画のシャットダウン、およびページング・スペースを解放するためのトランザクションの打ち切りといったユーザー定義事象を説明する例が samples サブディレクトリーにあります。

/user/sbin/cluster/events/rules.hacmprd 規則ファイルには HACMP 事象が含まれています。このファイルの各事象記述には、以下の 9 つの構成要素があります。

- 事象名。この名前は固有でなければなりません。
- 状態。これは、事象の修飾子です。事象名および状態は、規則トリガーです。 HACMP ES クラスター管理プログラムは、規則トリガーが事象名および状態に対応する規則を検出した場合にのみ、回復処理を開始します。
- リソース・プログラム・パス。回復プログラムが入った *xxx*.rp ファイルを指定する全パスです。
- 回復タイプ。これは、将来の利用のために予約されています。
- 回復レベル。これは、将来の利用のために予約されています。
- リソース変数名。事象管理プログラムの事象で使用します。

- インスタンス・ベクトル。事象管理プログラムの事象で使用します。これは、形式「name=value」の要素セットです。各値は、システム内のリソースのコピーと、拡張子を使用してリソース変数のコピーを一意的に識別します。
- 述部。事象管理プログラムの事象で使用します。これは、リソース変数と他の要素との関係式です。この式が真の場合、事象管理サブシステムは事象を生成して、クラスター管理や該当するアプリケーションに通知します。
- リアーム述部。事象管理プログラムの事象で使用します。この述部を使用して、基本述部の状況を変更する事象を生成します。この述部は一般に、基本述部の反対です。また、関心のある条件の上限および下限の境界を設定するための事象述部でも使用できます。

たとえ行を使用しない場合でも、事象定義では各オブジェクトに 1 行が必要となります。これらの行が削除されると、 HACMP ES クラスター管理プログラムは事象定義を適切に解析できなくなり、これが原因でシステムがハングする可能性があります。「#」で始まる行は注釈行として扱われます。

注: 規則ファイルは、注釈行を数えないで、事象定義ごとに 9 行だけを必要とします。規則ファイルの最後にユーザー定義の事象を追加する場合、ファイルの末尾の不要な空白行を削除することが重要です。削除しなければ、ノードがハングします。

以下は node_up の事象定義の例です。

```
##### Beginning of the Event Definition: node_up
#
TE_JOIN_NODE
0
/usr/sbin/cluster/events/node_up.rp
2
0
# 6) Resource variable - only used for event management events
# 7) Instance vector - only used for event management events
# 8) Predicate  - only used for event management events
# 9) Rearm predicate - only used for event management events
###### End of the Event Definition: node_up
```

この例は、 rules.hacmprd ファイルにある事象定義の 1 つです。この例では、 node_up 事象が発生すると、回復プログラム /usr/sbin/cluster/events/node_up.rp が呼び出されます。値は、状態、回復タイプ、および回復レベルについて指定されます。ここでは、リソース変数、インスタンス変数、述部、およびリアーム述部用に 4 つの空白行があります。

非標準 HACMP ES 事象に対応するために、他の事象を定義することができます。たとえば、/tmp ファイルが 90 パーセント以上いっぱいになった事象を定義するには、 rules.hacmprd ファイルを修正する必要があります。

IBM AIX 並列システム・サポート・プログラム (PSSP) では、多くの事象が事前定義されています。これらの事象は、(ユーザー定義事象内で使用されるときに) 次のように活用できます。

1. クラスターを停止する。
2. rules.hacmprd ファイルを編集する。ファイルを修正する前にバックアップしてください。事前定義した PSSP 事象を追加します。クラスター内の全ノード間で同期点を必要とする場合は、回復プログラムで **barrier** コマンドを使用します。 (バリア・コマンドおよび回復プログラムの同期については、 HACMP 概念、インストール、管理の手引きで詳細をお読みください。)
3. クラスターを再始動する。クラスター管理プログラムの始動時に rules.hacmprd ファイルがメモリーに保管されます。変更を正確に実装するには、すべてのクラスターを再始動します。クラスター内に矛盾する規則があってはなりません。
4. クラスター管理プログラムは、 rules.hacmprd ファイルにあるすべての事象を使用します。

HACMP ES は PSSP 事象検出を使用して、ユーザー定義事象を取り扱います。 PSSP 事象管理サブシステムは、さまざまなハードウェアおよびソフトウェア・リソースをモニターして、包括的な事象検出を提供します。

リソース状態は、リソース変数によって表されます。リソース条件は、述部と呼ばれる式によって表されます。

事象管理は、リソース・モニターからリソース変数を受け取ります。リソース・モニターは、特定のリソースの状態を観察し、その状態をいくつかのリソース変数に変換します。これらの変数は周期的に事象管理に渡されます。事象管理は、 rules.hacmprd で HACMP ES クラスター管理プログラムが指定した述部を各リソース変数に適用します。述部が真として評価されると、事象が生成されて、クラスター管理プログラムに送信されます。クラスター管理プログラムは票決プロトコルを開始し、回復プログラム内の「ノード・セット」で指定されたノード・セットで、 (事象優先順位にしたがって) 回復プログラム・ファイル (xxx.rp) が実行されます。

回復プログラム・ファイル (*xxx*.rp) は、 1 つまたは複数のプログラム行でできています。各行は、次の書式で宣言されています。

```
relationship     command_to_run     expected_status     NULL
```

行の各値の間には少なくとも 1 つのスペースがなければなりません。
「Relationship」は、どの種類のノードでどのプログラムを実行するかを決定する値です。次の 3 つの関係タイプがサポートされています。

- すべて (All)。現行 HACMP クラスターの全ノードで指定コマンドまたはプログラムが実行されます。
- 事象 (Event)。事象が発生したノードでのみ指定コマンドまたはプログラムが実行されます。
- その他 (Other)。事象が発生しなかった 全ノードで、指定コマンドまたはプログラムが実行されます。

「Command_to_run」は引用区切りストリングで、実行可能プログラムに対する全パスが指定される場合もあれば、指定されない場合もあります。 HACMP で送達される事象スクリプトでは、相対パス定義を使用できます。他のスクリプトまたはプログラムでは、 HACMP 事象スクリプトと同じディレクトリーにある場合でも、全パスの指定をしなければなりません。

「Expected_states」は、指定コマンドまたはプログラムの戻りコードです。このコードは整数値または「x」です。「x」を使用すると、クラスター管理プログラムは戻りコードを無視します。他のすべてのコードでは、予期された戻りコードと等しくなければなりません。等しくならない場合、クラスター管理プログラムは事象障害を検出します。この事象を処理すると、 (手作業による介入で) 回復が行われるまで、処理が「停止」します。手作業による介入がなければ、ノードは他のノードとの同期を図れません。クラスター管理プログラムが全ノードを制御するには、全ノード間の同期が必須です。

「NULL」は、将来の利用のために予約されたフィールドです。「NULL」という語は、バリア行を除いて、個々の行の末尾に表示されます。 2 つのバリア・コマンドの間、または最初のコマンドの前で複数の回復コマンドを指定すると、ノード自身およびノード間で回復コマンドが並列処理されます。

バリア・コマンドは、全クラスター・ノード間の全コマンドを同期させるために使用します。ノードが回復プログラムのバリア・ステートメントをヒットすると、クラスター管理プログラムはそのノードでバリア・プロトコルを開始します。バリア・プロトコルは 2 フェーズ・プロトコルなので、すべてのノードが回復プログラム内のバリアに遭遇し、プロトコルの承認を「票決した」ときに、両方のフェーズが完了したことがすべてのノードに通知されます。

次のようにプロセスを要約することができます。

1. (事前定義事象の) グループ・サービス /ES または (ユーザー定義事象の) 事象管理のいずれかが、事象について HACMP ES クラスター管理プログラムに通知します。
2. クラスター管理プログラムは rules.hacmprd ファイルを読み取り、事象にマップされた回復プログラムを判別します。
3. クラスター管理プログラムは、一連の回復コマンドから成る回復プログラムを実行します。
4. 回復プログラムは、シェル・スクリプトまたは 2 進コマンドである回復コマンドを実行します。 (HACMP for AIX では、回復コマンドは、HACMP 事象スクリプトと同じです。)
5. クラスター管理プログラムは、回復コマンドから戻り状況を受け取ります。予期しない状況により、 (smit cm_rec_aids または /usr/sbin/cluster/utilities/clruncmd コマンドによって) 手作業の介入が実行されるまで、クラスターは「停止」されます。

HACMP ES スクリプト・ファイル

DB2 UDB EEE には、フェールオーバー回復およびユーザー定義事象の以下のサンプル・スクリプトが組み込まれています。スクリプト・ファイルは、$INSTNAME/sqllib/samples/hacmp/es ディレクトリーにあります。このスクリプトは、「現状のまま」でも動作し、また回復アクションをカスタマイズすることもできます。

- DB2 データベース区画回復スクリプト rc.db2pe。これは、データベース区画上で HACMP 構成を開始したり、停止したりするのに使用されるスクリプト・ファイルです。また、DB2 インスタンス所有者の NFS サーバーでは、 HACMP 開始および停止スクリプトとしても動作します。
- HACMP ES 用の DB2 固有のユーザー定義事象。 6 つの省略時事象が含まれています。 1 つは処理回復用、2 つはページング・スペース用、 3 つは NFS および自動マウント回復用です。
- DB2 インスタンス NFS ファイル・サーバーのフェールオーバー。このスクリプトはフェールオーバーによって、 DB2 インスタンス用のファイル・システム・サーバーを回復してバックアップします。
- ネットワークのフェールオーバー。スクリプト network_up_complete、network_back、 network_down_complete、および network_down は、 SP スイッチ・アダプターが失敗した場合に SP DB2 データベース区画のフェールオーバーを可能にします。

- SP GUI 透視図のモニター事象を定義するためのスクリプト。事象およびハードウェア透視図を使用すれば、フェールオーバーおよびユーザー定義回復のモニターをすることができます。透視図の詳細については、PSSP 管理に関する資料を参照してください。
- HACMP ES ノードでコア・スクリプトと事象をインストールおよび削除するためのインストール・スクリプト。
- HACMP および DB2 構成をモニターするための SP 透視図問題管理 (pman) リソースを作成および削除するためのスクリプト・ファイル。

回復スクリプトは、回復操作を実行する各ノードにインストールする必要があります。スクリプト・ファイルは、 SP コントロール・ワークステーションまたは他の指定 SP ノードから集中方式でインストールすることができます。

1. スクリプトを `$INSTNAME/sqllib/samples/hacmp/es` ディレクトリーから、SP コントロール・ワークステーションか、 **pcp** および **pexec** コマンドを実行できる別の SPノードにコピーする。これらのコマンドはインストール操作の際に必要です。
2. フェールオーバー構成で (BUFFPAGE などの) キー・パラメーターを設定することにより、実際の環境に合わせて `reg.parms.SAMPLE` ファイルおよび `failover.parms.SAMPLE` ファイルをカスタマイズする。相互引き受け構成の場合、一般にフェールオーバーの設定は、通常の設定サイズの半分またはそれ以下に調整されます。また、独自の名前 (「SAMPLE」ではない) で名前変更したこれらのファイルのコピーも使用します。
3. 必要に応じて、`rc.db2pe` ファイルで 5 つのパラメーター、NFS_RETRIES、START_RETRIES、MOUNT_NFS、STOP_RETRIES、および FAILOVER をカスタマイズする。大半の実装では、再試行およびフェールオーバーの設定で十分でしょう。 MOUNT_NFS 設定は、NFS サーバー可用性のパッケージを使用するかどうかに応じて構成します。この設定は、`rc.db2pe` をマウントし、 DB2 インスタンス所有者の NFS ホーム・ディレクトリーを自分で検証する場合に設定してください。 FAILOVER パラメーターを「YES」に設定すると、 `db2_proc_restart` が呼び出され、 DB2 データベース区画の再始動が試行されます。再始動操作が失敗すると、HACMP はフェールオーバーによりシャットダウンします。
4. 事象ファイルで `db2_paging_action`、 `db2_proc_recovery`、`nfs_auto_recovery` をカスタマイズする。 `pwq` を編集して、 DB2 インスタンス所有者に変更します。 `db2_paging_action` をカスタマイズして、ページング・スペースの使用量が 90 パーセントを超えた場合に実行するアクションを指定します。 (この処置が実行されない場合、DB2 データベース区画は停止します。) 回復アクションがさらに必要であれば、スクリプトを修正します。

5. db2_inst_ha を使用して、指定したノードにスクリプトと事象をインストールする。 (これらのノードには、インストール前に HACMP ES をインストールしておく必要があります。) db2_inst_ha の構文は、次のとおりです。

   ```
   db2_inst_ha $INSTNAME/sqllib/samples/hacmp/es <nodelist> <DATABASENAME>
   ここで、
      $INSTNAME/sqllib/samples/hacmp/es は、スクリプトおよび事象がある
         ディレクトリーを表します。
      <nodelist> は、ノードの pcp または pexec スタイルを表します。
         たとえば、1-16 または 1、2、3、4 です。
      <DATABASENAME> は、通常およびフェールオーバーのパラメーター・
         ファイルに使用するデータベースの名前です。
   ```

   reg.parms.SAMPLE および failover.parms.SAMPLE は各ノードにコピーされ、 reg.parms.DATABASENAME に名前変更されます。 db2_inst_ha は /usr/bin の各ノードにファイルをコピーし、次の HACMP 事象ファイルを更新します。

   ```
   /usr/sbin/cluster/events/rules.hacmprd
   /usr/sbin/cluster/events/network_up_complete
   /usr/sbin/cluster/events/network_down_complete
   ```

6. HACMP を使用してシステムとスクリプトを構成する。
7. **create_db2_events** コマンドを使用して、問題管理リソース (pman) および SP GUI 透視図のモニター事象をインストールする。透視図では、構成およびカスタマイズがさらに必要です。透視図の詳細については、PSSP 管理の手引きを参照してください。
8. ha_db2stop コマンドを使用して、 HACMP ES フェールオーバー回復を起こさずにデータベース区画を遮断します。このコマンドを使用するには、ファイルをデータベース・ユーザーのホーム・ディレクトリーにコピーし、そのユーザーに許可と所有権が設定されていることを確認してください。フェールオーバー回復なしでデータベースを停止するには、そのユーザーとして次のように入力します。

   ```
   ha_db2stop
   ```

   注: コマンドが戻るのを待機することが必要です。 ctrl-C 割り込みを使用して終了するか、処理を強制終了すると、フェールオーバー回復を早まって再度使用可能にしてしまい、一部のデータ区画が停止しない可能性があります。

HACMP ES を使用した DB2 回復スクリプトの操作

HACMP ES は、次の方法で DB2 回復スクリプトを呼び出します。

- node_up_local (ノードの始動)

HACMP は node_up 順序列を実行し、このノードで所有されている (カスケードを使用) か割り当てられている (回転を使用) リソース・グループで指定されたボリューム・グループ、論理ボリューム、ファイル・システム、および IP アドレスを獲得する。

node_up_local_complete が実行されると、 rc.db2pe を含むアプリケーション・サーバー定義が開始され、この物理ノード上のアプリケーション・サーバー定義で指定されたデータベース区画が始動する。

注: rc.db2pe は、始動モードで実行されると、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、 reg.parms.DATABASE で指定された DB2 パラメーターを調整します。

各ノードは始動時にこの順序に従います。複数の HACMP クラスターを持っていて、並列始動すると、一度に複数のノードが始動します。

- node_down_remote (フェールオーバー)

  HACMP は、指定引き受けノード上のリソース・グループで指定されたボリューム・グループ、論理グループ、ファイル・システム、および IP アドレスを獲得する。

  node_down_remote_complete が実行されると、HACMP は、このデータベース区画用のリソース・グループで指定されたアプリケーション・サーバー始動スクリプトとして rc.db2pe を実行する。

  注: rc.db2pe は、相互引き受けモードで実行されると、そこで実行される DB2 データベース区画を停止し、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、 failover.parms.DATABASE で指定された DB2 パラメーターを調整した後、物理引き受けノード上の両方のデータベース区画を始動します。

- node_up_remote (障害ノードの再統合 - リソース・グループの引き受けは必ずカスケードが行う)

  node_up_remote が古い引き受けノードで実行されると、アプリケーション・サーバー定義により、rc.db2pe が停止モードで実行される。

  注: rc.db2pe は、再統合モード (相互引き受け) で実行されると、そこで実行される両方の DB2 データベース区画を停止し、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、 reg.parms.DATABASE で指定された DB2 パラメーターを調整した後、この物理引き受けノードで保持されるデータベース区画だけを始動します。

古い引き受けノードは、再統合されるノードに所有されるリソース・グループで指定されたボリューム・グループ、論理ボリューム、ファイル・システム、および IP アドレスを解放する。

HACMP は、再統合されるノードが現在所有するリソース・グループで指定されたボリューム・グループ、論理ボリューム、ファイル・システム、および IP アドレスを再獲得する。

node_up_local_complete が実行されると、 rc.db2pe を含むアプリケーション・サーバー定義が開始され、この再統合物理ノード上のアプリケーション・サーバー定義で指定されたデータベース区画が始動する。

注: rc.db2pe は、始動モードで実行されると、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、 reg.parms.DATABASE で指定された DB2 パラメーターを調整します。

- node_down_local (ノード停止または引き受けによる停止)

  node_down_local が停止ノードで実行されると、アプリケーション・サーバー定義により、 rc.db2pe が停止モードで実行される。

  注: rc.db2pe は、停止モードで実行されると、パラメーター (parms) ファイルと一致するデータベース・ディレクトリー中の各 DATABASE に合わせて、 failover.parms.DATABASE で指定された DB2 パラメーターを調整した後、 DB2 データベース区画 (これは引き受け用) を停止します。

  HACMP は、ノードが現在所有するリソース・グループで指定されたボリューム・グループ、論理ボリューム、ファイル・システム、および IP アドレスを解放する。

- db2_proc_recovery (db2 処理停止)

  すべてのノードで db2_proc_restart スクリプトが実行される。障害の発生したノードでは、正しい DB2 データベース区画が再始動されます。

- db2_paging_recovery (ページング・スペースの回復)

  すべてのノードで db2_paging_action スクリプトが実行される。ノードでページング・スペースが全容量の 70 パーセントを超える場合は、 wall コマンドが発行される。ノードでページング・スペースが全容量の 90 パーセントを超える場合は、この物理ノード上の DB2 データベース区画が停止され、再始動されます。

- nfs_auto_recovery (nfs または自動マウント処理の失敗)

すべてのノードで、rc.db2pe スクリプトが NFS モードで実行される。NFS 処理の実行が停止すると、実行が再開されます。自動マウント処理の実行が停止すると、同様の方法で再開されます。

- network_down_complete (ネットワーク障害 - SP スイッチ)

  net_down スクリプトが呼び出される。この呼び出しにより、SP スイッチ・ネットワークとしてのネットワークが検査され、ダウンしているかどうかが検証されます。ダウンしていれば、ユーザー定義の時間間隔だけ待機されます。省略時の時間間隔は、100 秒です。

  network_up_complete 事象で指示されたとおりに SP スイッチ・ネットワークが復旧すると、回復は実行されない。

  時間制限に達すると、HACMP はフェールオーバーで停止する。

注: すべての事象は、SP 問題管理および SP 透視図 GUI を使用してモニターできます。

他のスクリプト・ユーティリティー

他にも、以下のようなスクリプト・ユーティリティーを使用できます。

- ha_cmd は、コントロール・ワークステーションから SP ノードで HACMP を始動するために使用するコマンドです。構文は次の通りです。

```
ha_cmd <noderange> <START|STOP|TAKE|FORCE>
ここで、
   <noderange> は SP ノード範囲の pcp または pexec スタイルです。
      たとえば、「ha_cmd 3-6 START」とすると、HACMP はノード 3、4、5、6 で始動します。
               「ha_cmd 5 TAKE」は、ノード 5 の HACMP をシャットダウンして
               引き受けを行います。
```

- ha_mon は、 SP コントロール・ワークステーションから HACMP の hacmp_out ファイルをモニターするためのコマンドです。構文は次の通りです。

```
ha_mon <node>
ここで、
   <node> は、モニターされる SP ノードを表します。
   ha_mon は、指定するノード上の /tmp/hacmp.out ファイルに「-f を追加」します。
```

- db2_turnoff_recov は、すべての HACMP (非フェールオーバー) 回復を一時的に使用不能にする、極めて珍しい状況下で使用するコマンドです。 DB2 処理、ページング、NFS、または自動マウント回復が開始されなくなります。この機能は、HACMP 規則ファイルから該当する回復の事象スタンザを削除します。 HACMP を停止し、再始動する必要があります。構文は次の通りです。

```
db2_turnoff_recov <nodelist>
```

- db2_turnon_recov は、HACMP (非フェールオーバー) 回復を再び使用可能にするためのコマンドです。このコマンドは、 db2_turnoff_recov の後に使

用して HACMP の規則ファイルを復元し、ユーザー定義事象による回復を実行できるようにします。 HACMP を停止し、再始動する必要があります。構文は次の通りです。

```
db2_turnon_recov <nodelist>
```

HACMP クラスターのモニター

HACMP ES にある既存のモニター・ユーティリティーに加えて、 DB2 HACMP ES 構成をモニターする SP 問題管理 (pman) 事象を作成するためのスクリプトが提供されています。 SP コントロール・ワークステーションから HACMP 状況をモニターするには、次のようにします。

- コントロール・ワークステーションで HACMP クライアント・コードをインストールします。
- /usr/sbin/cluster/etc/clhosts ファイルを編集し、モニターしたいノードの SP イーサネット IP アドレスを含めます。
- コマンド startsrc -s clinfo を呼び出して、クラスターのモニターを開始します。

HACMP は、クラスターをモニターするためのインターフェース、つまり、/usr/sbin/cluster/clstat を提供します。

HACMP RS およびユーザー定義事象の SP 透視図 GUI で問題管理モニターを使用するには、次のようにします。

1. create_db2_events <nodelist> を呼び出す。ここで、*nodelist* には、 pcp または pexec スタイル・ノードが含まれます。このスクリプトは透視図により、モニターするための 5 つの pman 事象を作成します。

   注: これらの事象の作成では、リソース変数 PSSP.pm.User_state12-16 が使用されます。これらのリソース変数がすでに他の目的で使用されている場合は、 create_db2_events および update_db2_events を更新して、異なるリソース変数を使用する必要があります。

2. コントロール・ワークステーションで透視図を開始する。ランチ・パッドから、事象透視図を選択します。 5 つの事象、つまり、db2_hacmp_recovery、 db2_process_recovery、db2_paging_err、db2_nfs_err、および Errlog_PERM_entry があるはずです。
3. 個々の事象をダブルクリックする。表示される画面上で、事象の条件を (定義表の中で) 登録してください。下矢印の横にある Name: "unnamed" をクリックし、条件として指定する事象と同じ名前を選択します。 「応答オプション (Response Options)」 タブを選択します。画面上部のボタン (「透視図にメッセージを送信する事象セッション (Send Message to Perspectives

event session)」) をクリックします。これらの事象の実行で使用するコマンド、errlog 項目、および SNMP トラップを指定できます。事象ログの表示は透視図セッションでのみ保持されます。したがって、個々の事象に応じた AIX エラー・ログ項目を作成することもできます。**「了解 (OK)」**を選択し、ウィンドウを閉じます。

4. 「透視図 (Perspectives)」ランチ・パッドから、「ハードウェア透視図 (hardware Perspective)」を選択する。
5. ハードウェア・フレーム GUI が表示されたら、「視点 (View)」—>「モニター (Monitor)」を選択する。すると、SP でモニターできる事象のリストが表示されます。リストを下までスクロールすると、2 つの追加事象があります。一方は HACMP DB2 回復用 (db2_ha_ind) で、他方は SP ノード PERM エラー用 (Errlog_PERM_mon) です。モニターしたい事象を選択します。 (事象が発生すると、ノードは赤い「X」を表示します。モニターするすべての条件を満たすと、ノードの表示は緑色になります。) host_responds、 switch_responds、および node_power_LED が通常使用されます。また、DB2 HACMP 回復やノード上の PERM エラーもモニターできます。

注: pman および透視図用の db2_hacmp_mon および db2_hacmp_recovery 変数は、 HACMP クラスター状況を反映しません。むしろ、これらの変数は、DB2 を開始したり停止したりするための rc.db2pe 操作の状況を反映します。 HACMP clstat モニターには「実際の」 HACMP 状況が表示され、HACMP クラスター状態が反映されます。 db2_hacmp_ind で HACMP 状況に似たモニターを反映させたい場合は、 /etc/inittab ファイルに次の行を追加します。

```
haind:2:wait:/usr/bin/db2_update_events HAIND OFF 2>&1 >/dev/null
```

実装で NetView を使用する計画であれば、構成のモニターに HAVIEW (HACMP の一部) を使用することを検討してください。この製品の構成については、 NetView の資料を参照してください。

DB2 SP HACMP ES のインストール

DB2 ユニバーサル・データベースで HACMP ES をインストールをするための計画に役立つ資料として、以下にインストールおよび移行プロセスを段階的に示した説明を行います。

DB2 SP HACMP ES の新規インストール

HACMP ES をインストールするには、次のようにします。

1. 個々の SP ノードに AIX オペレーティング・システムをインストールする (SP インストールおよび管理の手引きを参照してください)。コントロール・ワークステーションでも各 SP ノードでも、適切なページング・スペースを使用できることを確認します。他の修正可能構成パラメーターと一緒にスイッチ構成が考慮され、実装されていることを確かめてください。使用する SP モニター (透視図) を適切な位置に置きます。 SP dsh、pcp、および pexec コマンドが動作することを確かめます。
2. 実際のデータベース・レイアウトを設計する。この設計には少なくとも、使用するノードの数、物理ノードに対する DB2 データベース区画のマッピング、ノードまたは区画あたりのディスク要件、表スペースに関する考慮事項を含めます。また、DB2 インスタンスの主要な所有者となる人、およびその所有者と他のユーザーが必要とするアクセス許可も考慮してください。
3. 冗長アダプター、ミラーリングされるディスク、ディスクの兄弟接続をはじめとする、外部 SSA ディスクの構成を計画する。
4. データベースのレイアウトおよび SSA 構成を使用して、HACMP の計画、インストール、および管理の手引きにある HACMP のワークシートを完成させる。
5. 外部 SSA ディスク構成を実装する。全ドライブでマイクロコード・レベルが一貫していることを確認し、 Maymap ユーティリティーを使用して、ワークシートを検証し、相違をなくします。
6. 各 SP ノードに DB2 UDB EEE をインストールする。
7. 各 SP ノードに HACMP ES をインストールする。
8. **db2_inst_ha** コマンドを使用して、 SP パッケージに DB2 UDB EEE HACMP ES をインストールする。
9. DB2 のメイン・インスタンス・ユーザーを作成し、すべてのノードにアクセスできることを確認する。これにより、高可用性ユーザーになったというわけではありません。 SP コントロール・ワークステーション上で一時的に SP ユーザーとなることが可能です。
10. DB2 インスタンスおよびデータベースを作成する。 **db2start** を呼び出してこの DB2 インスタンスおよびデータベースを操作可能にしてから、次のステップに進む前に、**db2stop** を呼び出します。
11. HACMP を追加する前にデータベースをロードしたい場合は、この時点で実行してください。
12. HACMP ワークシートと本書の情報にしたがって、 SP ノード・トポロジーおよびリソース上に HACMP ES を構成する。

13. DB2 メイン・インスタンス・ユーザーの NFS サーバー・ノードから始めて、本書で指定された内容に従って、全ノード上で (/etc/security/user および /etc/passwd を修正して) このユーザーを変更する。このユーザーは、高可用性 NFS ユーザーとなります。また、このノードおよびバックアップでは、/etc/exports が更新されます。すべてのノードは、スイッチ別名 IP アドレス全体で、 NFS (各ノードの /etc/filesystems で項目を持つ) を使用して、このディレクトリーをマウントすることができます。
14. メイン・インスタンス・ユーザーのホーム・ディレクトリーを「tar 圧縮」し、新しい位置のホーム・ディレクトリーを「tar 圧縮解除」する。
15. 個々の SP ノードに NFS ファイル・システムを作成し、新しいメイン・インスタンスのホーム・ディレクトリーをマウントする。
16. NFS サーバー・ノードで HACMP を始動する。 /tmp/hacmp.out を調べて、 HACMP の始動が成功したことを確かめてください。 **ha_mon** コマンドは、このファイルを作成されたとおりにモニターするために使用できます。
17. 他のノードを一度に 1 つずつ立ち上げる。 /tmp/hacmp.out を調べて、各ノードが正常に完了したことを確かめます。 **ha_mon** コマンドは、このファイルを作成されたとおりにモニターするために使用できます。
18. 透視図および問題管理を使用して、任意のモニターをセットアップする。
19. 各ノードで並行保守アクションをシミュレートして、各ノードでフェールオーバーが機能するかどうかを検証する。 (TAKE オプションを指定して) **ha_cmd** コマンドを使用すれば、引き受けで HACMP を正常に停止できます。 /tmp/hacmp.out を調べ、モニター・ツールを使用して、引き受けおよび再統合が成功したことを検証します。

DB2 SP HACMP ES の移行

HACMP をインストールしていない環境から HACMP をインストールした環境へ移行する場合は、以下の概説を考慮してください。

1. 既存の外部ディスクを、高可用性で、兄弟接続され、ミラーリングされた構成に変換する。異なるノード上の異なる論理ボリュームの名前は兄弟接続時に固有名にしなければならない ことを念頭に置いて、この構成を達成するために余分のハードウェアおよびディスクを追加します。これは、ボリューム・グループ、論理グループ、およびファイル・システムに適用されます。
2. HACMP の計画、および本書にあるワークシートを含め、関連ワークシートを完成させる。

3. 外部 SSA ディスク構成の変更を実装する。全ドライブでマイクロコード・レベルが一貫していることを確認し、 Maymap ユーティリティーを使用して、ワークシートでの検証と相違の訂正を行います。

   注: RAID5 構成では SSA ディスクがサポートされています。同一の RAID ループにある 2 つの SSA アダプターだけが、許可されている構成です。兄弟接続されている RAID の HACMP 構成には、 1 つのノードにつき 1 つしかアダプターはサポートされていません。この構成では、アダプターはディスクへのアクセスの障害の単一ポイントであり、アダプターの故障を検出したり HACMP フェールオーバー事象にこれをプロモートしたりするために、さらに構成することが勧められています。 SSA アダプターに障害が起きた場合には、 AIX エラー通知を使用してノードを構成することが最も簡単な方法です。 AIX エラー通知についての詳細は、*HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide* を参照してください。

4. 各 SP ノードに HACMP ES をインストールする。
5. **db2_inst_ha** コマンドを使用して、 SP パッケージに DB2 UDB EEE HACMP ES をインストールする。
6. HACMP ワークシートと本書の情報にしたがって、 SP ノード・トポロジーおよびリソース上に HACMP ES を構成する。
7. メイン DB2 インスタンス・ユーザーの NFS サーバー・ノードから始めて、本書で指定された内容に従って、全ノード上で (`/etc/security/user` および `/etc/passwd` を修正して) このユーザーを変更する。このユーザーは、高可用性 NFS ユーザーとなります。また、このノードおよびバックアップでは、`/etc/exports` が更新されます。すべてのノードは、スイッチ別名 IP アドレス全体で、 NFS (各ノードの `/etc/filesystems` で項目を持つ) を使用して、このディレクトリーをマウントすることができます。
8. メイン・インスタンス・ユーザーのホーム・ディレクトリーを「tar 圧縮」し、新しい位置のホーム・ディレクトリーを「tar 圧縮解除」する。
9. 個々の SP ノードに NFS ファイル・システムを作成し、新しいメイン・インスタンスのホーム・ディレクトリーをマウントする。
10. NFS サーバー・ノードで HACMP を始動する。 `/tmp/hacmp.out` を調べて、 HACMP の始動が成功したことを確かめてください。 **ha_mon** コマンドは、このファイルを作成されたとおりにモニターするために使用できます。
11. 他のノードを一度に 1 つずつ立ち上げる。 `/tmp/hacmp.out` を調べて、各ノードが正常に完了したことを確かめます。 **ha_mon** コマンドは、このファイルを作成されたとおりにモニターするために使用できます。

12. 透視図および問題管理を使用して、任意のモニターをセットアップする。
13. 各ノードで並行保守アクションをシミュレートして、各ノードでフェールオーバーが機能するかどうかを検証する。 (TAKE オプションを指定して) **ha_cmd** コマンドを使用すれば、引き受けで HACMP を正常に停止できます。 `/tmp/hacmp.out` を調べ、モニター・ツールを使用して、引き受けおよび再統合が成功したことを検証します。

DB2 SP HACMP ES ワークシート

以下に示すワークシートは、外部 SSA ディスク構成の準備で記入した HACMP ワークシートと一緒に使用します (HACMP のワークシートは、HACMP の計画、インストール、および管理の手引きにあります)。それぞれに、完全な例とブランク・ワークシートが準備されています。

次の図には、最初のサンプル・ワークシートに記されている外部ディスクに対するデータベース構成が示されています。このデータベースの作成で使用したステートメントは、次のとおりです。

```
db2 create database pwq on /newdata
```

障害点を持たない論理ボリュームでは、 SSA 外部アダプターと外部 SSA ディスクの両方がミラーリングされています。この図では、**maymap** コマンドの出力に似た構成が説明されています。 Maymap は (AIXTOOLS で使用できる) ユーティリティーで、外部 SSA ディスク構成を表示できます。このユーティリティーをセットアップの計画時に使用することをお勧めします。

DB2 4 ノード・データベースの外部ディスクのセットアップ例

- 高可用性の兄弟接続を示します。

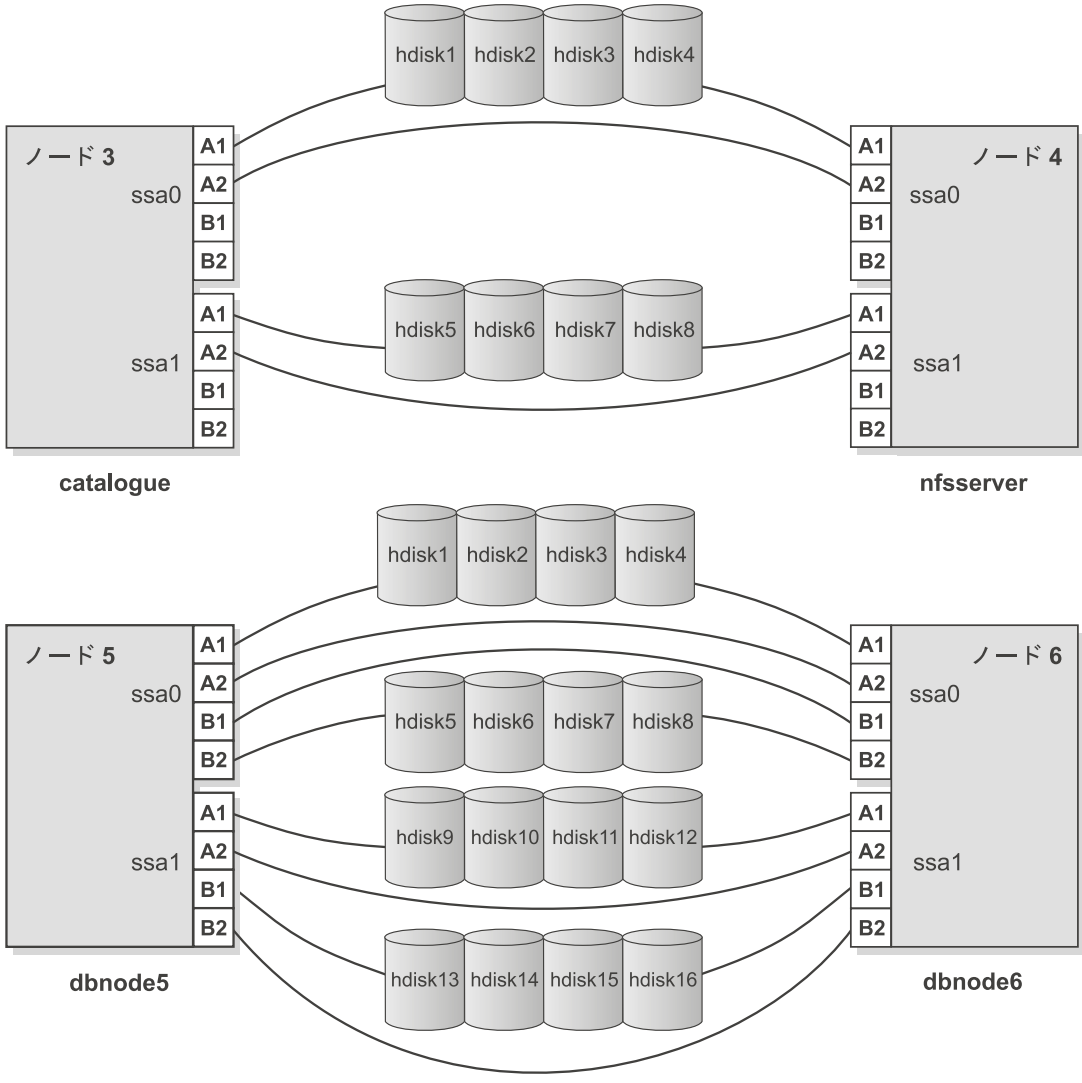


図 55. DB2 4 ノード・データベースの外部ディスクのセットアップ例

以下の表を検討する前に、ボリューム・グループに対する定足数設定、および論理ボリュームに対するミラーリング書き込みの整合性設定に関して、HACMP 資料を読んでおくことをお勧めします。両者で使用する設定は、可用性とパフォーマンスに直接影響を与えます。必ず、両方の設定を検討し、その

意味を理解してください。「定足数」と「ミラーリング書き込みの整合性」の両方で使用する一般的な設定値は、「off」です。

表 25. HACMP ボリューム・グループ、論理ボリューム、およびファイル・システム

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント (MB)	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者
3	havg3	8	hlv300	10	2	hdisk1 hdisk5	/newdata /pwq /NODE0003	hlog301	Catalognode マウント・ポイント ; ノード 4	root *
3	havg3	8	hlog301	1	2	hdisk1 hdisk5	適用外	適用外	Catalognode jfslog; ノード 4	root *
3	havg3	8	hlv301	10	2	hdisk2 hdisk6	適用外	適用外	Catalognode rawtemp スペース ; ノード 4	pwq **
4	havg4	8	hlv400	10	2	hdisk3 hdisk7	/dbmnt	hlog401	nfsserver pwq ホーム ; ノード 3	root *
4	havg4	8	hlog401	1	2	hdisk3 hdisk7	適用外	適用外	nfsserver jfslog; ノード 3	root *
5	havg5	8	hlv500	10	2	hdisk1 hdisk9	/newdata/ pwq/ NODE0005	HLOG501	Dbnode5 マウント・ポイント; ノード 6	root *
5	havg5	8	hlog501	1	2	hdisk1 hdisk9	適用外	適用外	Dbnode5 jfslog; ノード 6	root *
5	havg5	8	hlv501	10	2	hdisk2 hdisk10	適用外	適用外	Dbnode5 raw temp スペース ; ノード 6	pwq **
5	havg5	8	hlv502	100	2	hdisk2 hdisk10	適用外	適用外	Dbnode5 raw 表スペース; ノード 6	pwq **
5	havg5	8	halv503	100	2	hdisk3 hdisk11	適用外	適用外	Dbnode5 raw 表スペース ; ノード 6	pwq **
5	havg5	8	halv504	100	2	hdisk3 hdisk11	適用外	適用外	Dbnode5 raw 表スペース; ノード 6	pwq **
5	havg5	8	halv505	100	2	hdisk4 hdisk12	/dbdata5	hlog501	Dbnode6 システム表スペース; ノード 6	root *
6	havg6	8	hlv600	10	2	hdisk5 hdisk13	/newdata/ pwq/ NODE0006	hlog601	Dbnode6 マウント・ポイント; ノード 5	root *

表 25. *HACMP* ボリューム・グループ、論理ボリューム、およびファイル・システム *(続き)*

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント (MB)	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者
6	havg6	8	hlog601	1	2	hdisk5 hdisk13	適用外	適用外	Dbnode6 jfslog; ノード 5	root *
6	havg6	8	hlv601	10	2	hdisk6 hdisk14	適用外	適用外	Dbnode6 raw temp スペース; ノード 5	pwq **
6	havg6	8	hlv602	100	2	hdisk6 hdisk14	適用外	適用外	Dbnode6 raw 表スペース; ノード 5	pwq **
6	havg6	8	hlv603	100	2	hdisk7 hdisk15	適用外	適用外	Dbnode6 raw 表スペース; ノード 5	pwq **
6	havg6	8	hlv604	100	2	hdisk7 hdisk15	適用外	適用外	Dbnode6 raw 表スペース; ノード 5	pwq **
6	havg6	8	hlv605	100	2	hdisk8 hdisk16	/dbdata6	hlog601	Dbnode6 システム表スペース; ノード 5	root *

注:

1. * jfs ファイル・システムの論理ボリュームおよびログはルート許可を保持します。
2. ** 生データベース・スペースは、 /dev 生ファイル項目 (/dev/rxxxx) に対するデータベース・ユーザー許可を取得します。

表 26. *HACMP* ボリューム・グループ、論理ボリューム、およびファイル・システム *(ブランク)*

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント (MB)	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者

表 26. HACMP ボリューム・グループ、論理ボリューム、およびファイル・システム (ブランク) (続き)

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント (MB)	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者

表 26. HACMP ボリューム・グループ、論理ボリューム、およびファイル・システム (ブランク) (続き)

SP ノード	ボリューム・グループ名	PP サイズ (MB)	論理ボリューム名	PP の #	コピー	ハード・ディスク・リスト	ファイル・システム・マウント・ポイント (MB)	ファイル・システム・ログ論理ボリューム	ノードの説明とバックアップ	/dev 論理装置のユーザー所有者

表 27. HACMP NFS サーバーの計画

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc /filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)
3	/dbmnt	4	nfs_boot_3 nfs_client_3	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6

表 27. HACMP NFS サーバーの計画 (続き)

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc /filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)
4	/dbmnt	3	nfs_server_boot nfs_server	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
5	適用外	適用外	nfs_boot_5 nfs_client_5	nfs_server:/ dbmnt as /dbi	/dbi/pwq	適用外
6	適用外	適用外	nfs_boot_6 nfs_client_6	nfs_server:/ dbmnt as /dbi	/dbi/pwq	適用外

注:

1. /etc/passwd は必ず全ノードで同じものを指定する。これは、コントロール・ワークステーションから同期化できます。
2. 外部ファイル・システムにデータベース・インスタンス所有者の許可があることを確かめる。
3. /etc/filesystems には、マウント・パラメーター、つまり、hard、bg、intr、および rw を指定しなければならない。
4. /etc/exports に指定される

```
-root=ip1:ip2:ip3
```

は、サーバーとそのバックアップに限られる。

表 28. HACMP NFS サーバーの計画 (ブランク)

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc /filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)

表 28. HACMP NFS サーバーの計画 (ブランク) (続き)

SP ノード	外部ファイル・システム	バックアップ・ノード	SP スイッチ・ブートおよびサービス IP 別名のペア	マウントするファイル・システム (/etc /filesystems)	データベース・ホーム・ディレクトリーとして指定するファイル・システム	ファイル・システムをエクスポートする宛先アドレス (/etc/exports)

第13章 Windows NT 環境での高可用性

データベース・システムは、マシンが故障した場合に、その故障したマシン上のデータベース・サーバーが別のマシンで実行できるようにセットアップすることができます。 Windows NT では、 Microsoft Cluster Server (MSCS) でフェールオーバーのサポートを実装できます。 MSCS を使用するには、 MSCS 機能を持つ Windows NT Version 4.0 Enterprise Edition をインストールする必要があります。

MSCS は、物理ディスクと IP アドレスのフェールオーバー・サポートなど、クラスター環境で障害検出およびリソースの再始動の両方を実行することができます。 (失敗したマシンが再びオンラインになるとき、リソースがフォールバックするように前もって構成しておかない限り、自動的にフォールバックされることはありません。詳細については 298ページの『フォールバックの考慮事項』を参照してください。)

フェールオーバーをサポートするように DB2 インスタンスを使用可能にする前に、次の計画段階のステップを実行してください。

1. データ記憶域に使用したいディスクを決定します。各データベース・サーバーは、使用するために最低 1 つのディスクに割り当てられる必要があります。データを保管するのに使用するデータは共用ディスク・サブシステムに接続されているべきであり、さらに MSCS ディスク・リソースとして構成される必要があります。
2. リモート要求をサポートするのに使用したいデータベース・サーバーごとに、必ず 1 つの IP アドレスがあるようにします。

フェールオーバー・サポートを設定する場合、既存のインスタンスに設定したり、フェールオーバーの実装時に新しいインスタンスを作成したりすることができます。

フェールオーバー・サポートを使用可能にするには、以下のステップを実行します。

1. DB2MSCS ユーティリティーに入力ファイルを作成します。
2. **db2mscs** コマンドを呼び出します。
3. 区分データベース・システムを使用中である場合、データベース・ドライブ・マッピングを登録して相互引き受けを使用可能にします。 298ページの

『区分データベース環境で相互引き受け構成にデータベース・ドライブ・マッピングを登録する』を参照してください。

フェールオーバー・サポート用のインスタンスを使用可能にした後、構成は、図56 のようになります。

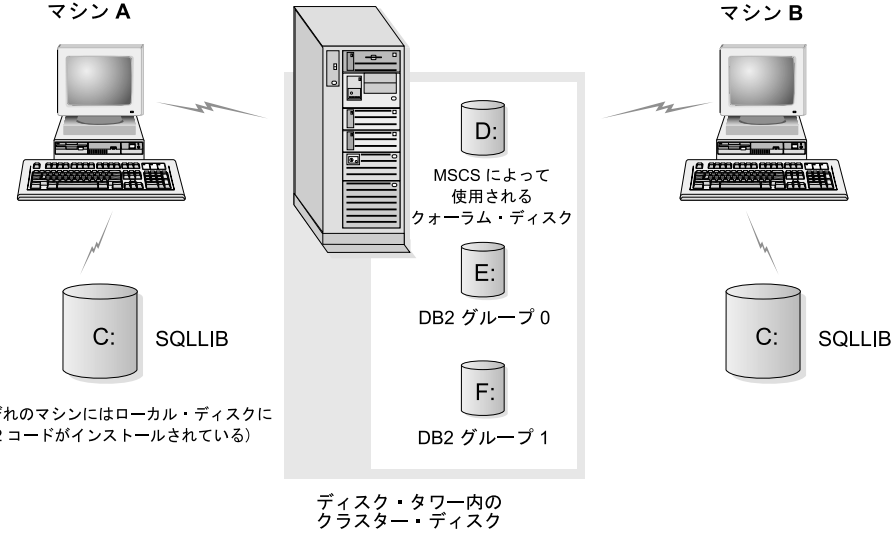


図 56. MSCS 構成の例

次のセクションでは、フェールオーバー・サポートの様々なタイプ、およびそれらを実装する方法を説明します。以下で説明されるステップのいずれかを実行する前に、 MSCS ソフトウェアを MSCS クラスターで使用したいすべてのマシン上にインストールしておく必要があります。さらに、DB2 もすべてのマシンにインストールしてください。

フェールオーバーの構成

2 つのタイプの構成が使用可能です。

- ホット・スタンドバイ
- 相互引き受け

現在、MSCS は 2 つのマシンのクラスターをサポートしています。

区分データベース環境では、必ずしもすべてのクラスターが同タイプの構成を持つ必要はありません。ホット・スタンドバイを使用するためにセットアップされるクラスターをいくつか、相互引き受けのためにセットアップされる他の

クラスターをいくつか持つことができます。たとえば、DB2 インスタンスが 5 つのワークステーションで構成される場合、そのうち 2 つを相互引き受けの構成を使用するため、 2 つをホット・スタンドバイを使用するため、 1 つのマシンをフェールオーバー・サポート用に構成しないでおくことができます。

ホット・スタンドバイ構成

ホット・スタンドバイ構成では、 MSCS クラスターの 1 つのマシンが専用のフェールオーバー・サポートを提供し、他のマシンがデータベース・システムに参加します。データベース・システムに参加しているマシンに障害が起こると、そのマシン上のデータベース・サーバーはフェールオーバー・マシンで開始します。区分データベース・システムで、あるマシン上で複数の論理ノードを実行していて、そのマシンに障害が起こる場合、論理ノードはフェールオーバー・マシンで開始します。 図57 にホット・スタンドバイ構成の例を示します。

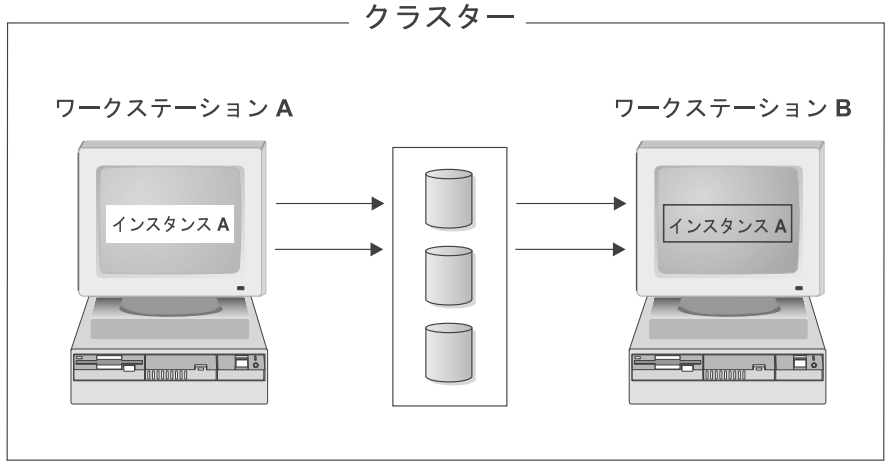


図 57. ホット・スタンドバイ構成

相互引き受け構成

相互引き受け構成では、両方のワークステーションがデータベース・システムに参加します (つまり、各マシンに実行しているデータベース・サーバーが最低 1 つある)。 MSCS クラスターのワークステーションのいずれかに障害が起こると、障害が起きたマシン上のデータベース・サーバーは他のマシンで実行を開始します。相互引き受け構成では、あるマシン上のデータベース・サーバーは、別のマシン上のデータベース・サーバーとは別個に障害を起こす場合が

あります。指定されたどの時点であっても、すべてのデータベース・サーバーはどのマシン上でも活動状態であることができます。 図58 に相互引き受け構成の例を示します。

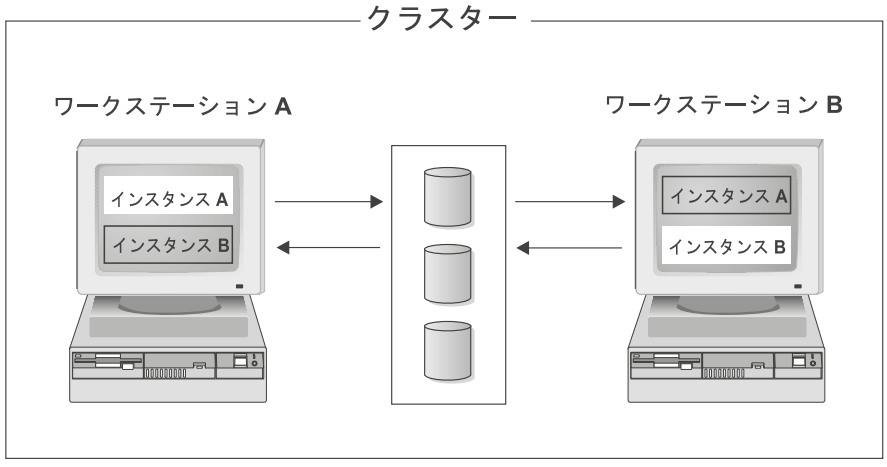


図 58. 相互引き受け構成

DB2MSCS ユーティリティーの使用方法

DB2MSCS ユーティリティーは、DB2 用のインフラストラクチャーを作成し、MSCS サポートを使用して Windows NT 環境でフェールオーバーをサポートするのに使用します。このユーティリティーを使用して、単一区画環境、および区分データベース環境の両方でフェールオーバーを使用可能にできます。

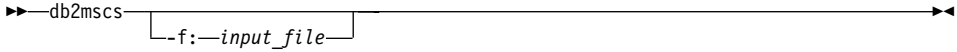
db2mscs コマンドは、インスタンスが所有するマシン上の、各インスタンスごとに 1 回呼び出します。 MSCS クラスター中のあるマシンで実行する DB2 インスタンスが 1 つだけである場合、このユーティリティーはホット・スタンドバイ構成をセットアップします。 MSCS クラスター中の各マシンで実行する 1 つのインスタンスがある場合、インスタンスが所有する各マシンで DB2MSCS を 1 回実行し、相互引き受け構成をセットアップします。

DB2MSCS ユーティリティーは次の処理を実行します。

1. DB2MSCS.CFG と呼ばれる入力ファイルから、必要な MSCS および DB2 パラメーターを読み取ります。入力パラメーターのフル・セットに関する情報は、289ページの『DB2MSCS.CFG ファイルの指定』を参照してください。

2. 入力ファイル中のパラメーターの妥当性検査をします。
3. DB2 リソース・タイプを登録します。
4. MSCS グループ (複数の場合もある) を作成して MSCS および DB2 リソースを入れます。
5. IP リソースを作成します。
6. Network Name リソースを作成します。
7. MSCS ディスクをグループに移動します。
8. DB2 リソース (複数の場合もある) を作成します。
9. DB2 リソースに必要なすべての従属関係を追加します。
10. クラスター化されていない DB2 インスタンスを、クラスター化されたインスタンスに変換します。
11. すべてのリソースをオンライン上に出します。

コマンド構文は、以下のとおりです。



ここで、

-f:*input_file*

MSCS ユーティリティーが使用する DB2MSCS.CFG 入力ファイルを指定します。このパラメーターが指定されないと、 DB2MSCS ユーティリティーは現行ディレクトリーにある DB2MSCS.CFG ファイルを読み取ります。

DB2MSCS.CFG ファイルの指定

DB2MSCS.CFG ファイルは、 DB2MSCS ユーティリティーが読み取るパラメーターを含む ASCII テキスト・ファイルです。書式 PARAMETER_KEYWORD=*parameter_value* を使用して、個別の行で各入力パラメーターを指定します。次に例を示します。

```
CLUSTER_NAME=WOLFPACK
GROUP_NAME=DB2 Group
IP_ADDRESS=9.21.22.89
```

2 つの構成ファイルのサンプルが、 /SQLLIB ディレクトリーの /CFG サブディレクトリーに入っています。 1 つ目の DB2MSCS.EE は、単一区画データベース環境の例です。 2 つ目の DB2MSCS.EEE は、区分データベース環境の例です。

DB2MSCS.CFG ファイルの 2 つのパラメーターは次のとおりです。

DB2_INSTANCE

DB2 インスタンスの名前。インスタンス名が指定されていない 場合、省略時のインスタンス (DB2INSTANCE 環境変数の値) が使用されます。

このパラメーターにはグローバルな効力範囲があり、 DB2MSCS.CFG ファイル中で 1 度だけ指定します。

このパラメーターは任意指定です。

例:

```
DB2_INSTANCE=DB2
```

インスタンスはすでに存在しているはずです。インスタンスの作成についての詳細は、*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* を参照してください。

DB2_LOGON_USERNAME

DB2 サービスのログオン・アカウントの名前。

このパラメーターにはグローバルな効力範囲があり、 DB2MSCS.CFG ファイル中で 1 度だけ指定します。

このパラメーターは、 DB2 エンタープライズ拡張エディション インスタンスにのみ必要です。

例:

```
DB2_LOGON_USERNAME=db2user
```

DB2_LOGON_PASSWORD

DB2 サービスのログオン・アカウントのパスワード。
DB2_LOGON_USERNAME パラメーターが提供されたのに DB2_LOGON_PASSWORD パラメーターが提供されない場合、DB2MSCS ユーティリティーはパスワードを要求します。パスワードは、コマンド行に入力されても表示されません。

このパラメーターにはグローバルな効力範囲があり、 DB2MSCS.CFG ファイル中で 1 度だけ指定します。

このパラメーターは、 DB2 エンタープライズ拡張エディション インスタンスにのみ必要です。

例:

```
DB2_LOGON_PASSWORD=xxxxxx
```

CLUSTER_NAME

MSCS クラスターの名前。この行に続いて指定されるすべてのリソースは、別の CLUSTER_NAME タグが指定されるまでこのクラスターで作成されます。

このパラメーターは、各クラスターごとに 1 度指定します。

このパラメーターは任意指定です。指定されないと、ローカル・マシン上の MSCS クラスター名が使用されます。

例:

```
CLUSTER_NAME=WOLFPACK
```

GROUP_NAME

MSCS グループの名前。このパラメーターが指定されると、 MSCS グループがなかった場合に新しいグループが作成されます。グループがあった場合、ターゲット・グループとして使用されます。この行に続いて作成される MSCS リソースはすべて、別の GROUP_NAME キーワードが指定されるまでこのグループに作成されます。

このパラメーターは、各グループごとに 1 度指定します。

このパラメーターは必須です。

例:

```
GROUP_NAME=DB2 Group
```

DB2_NODE

現行の MSCS グループに含まれるデータベース区画サーバー (ノード) のノード番号。複数の論理ノードが同じマシン上に存在する場合、各ノードは個別の DB2_NODE キーワードを必要とします。

このパラメーターは、DB2 リソースが正しい MSCS グループで作成されるように、 GROUP_NAME パラメーターの後で指定します。

このパラメーターは、 DB2 エンタープライズ拡張エディション インスタンスにのみ必要です。

例:

```
DB2_NODE=0
```

IP_NAME

IP アドレス・リソースの名前。 IP_NAME の名前は任意ですが、固有でなければなりません。このパラメーターが指定されると、IP アドレスの MSCS リソースが作成されます。

このパラメーターは、リモート TCP/IP 接続に必要です。このパラメーターを、区分データベース環境でインスタンス所有のマシンに指定する必要があります。このパラメーターは、単一区画データベース環境では任意指定です。

例:

```
IP_NAME=IP Address for DB2
```

注: DB2 クライアントは、この IP リソースの TCP/IP アドレスを使用して、 TCP/IP ノード・エントリーをカタログ化する必要があります。 MSCS IP アドレスを使用することにより、データベース・サーバーが他のマシンに対して障害を起こした場合でも、 IP アドレスが故障したマシン上で使用可能なので、 DB2 クライアントはデータベース・サーバーに接続できます。

IP リソースの属性は次のとおりです。

IP_ADDRESS

IP リソースの TCP/IP アドレス。このキーワードは、先行する IP リソースに TCP/IP アドレスを設定するのに使用します。

このパラメーターは、IP_NAME パラメーターが指定される場合に必要です。

例:

```
IP_ADDRESS=9.21.22.34
```

IP_SUBNET

先行する IP リソースのサブネット・マスク。

このパラメーターは、IP_NAME パラメーターが指定される場合に必要です。

例:

```
IP_SUBNET=255.255.255.0
```

IP_NETWORK

先行する IP リソースが所属している MSCS ネットワークの名前。このパラメーターが指定されていないと、システムが検出する最初の MSCS ネットワークが使用されます。

このパラメーターは任意指定です。

例:

```
IP_NETWORK=Token Ring
```

NETNAME_NAME

Network Name リソース名。 Network Name リソースを作成するのに指定します。

このパラメーターは、単一区画データベース環境の場合、任意指定です。区分データベース環境の場合は必須です。

例:

```
NETNAME_NAME=Network name for DB2
```

Network Name リソースの属性は次のとおりです。

NETNAME_VALUE

Network Name の値。

このパラメーターは、NETNAME_NAME パラメーターが指定される場合に必要です。

例:

```
NETNAME_VALUE=DB2SRV
```

NETNAME_DEPENDENCY

Network Name リソースの従属関係リスト。各 Network Name リソースには、IP アドレス・リソース上に従属関係がなければなりません。このパラメーターが指定されないと、 Network Name リソースはグループで最初の IP リソースとの従属関係になります。

このパラメーターは任意指定です。

例:

```
NETNAME_DEPENDENCY=IP Address for DB2
```

DISK_NAME

現行のグループに移動される物理ディスク・リソース名。必要なだけディスク・リソースを指定してください。

注:

1. ディスク・リソースはすでに存在しているはずです。
2. DB2MSCS ユーティリティーが MSCS サポートに DB2 インスタンスを構成すると、インスタンス・ディレクトリーがグループで最初の MSCS ディスクにコピーされます。インスタンス・ディレクトリーに異なる MSCS ディスクを指定するには、INSTPROF_DISK パラメーターを使用します。

例:

```
DISK_NAME=Disk E:
DISK_NAME=Disk F:
```

INSTPROF_DISK

MSCS ディスクに DB2 インスタンス・ディレクトリーを入れるように指定する任意指定パラメーター。このパラメーターが指定されていない と、 DB2MSCS ユーティリティーは、インスタンス・ディレクトリーと同じグループに属する*最初の* MSCS ディスクを使用します。

DB2 インスタンス・ディレクトリーは、 X:¥DB2PROFS ディレクトリー (X は MSCS ディスクのドライブ文字) の下の MSCS ディスクに作成されます。

例:

```
INSTPROF_DISK=Disk E:
```

TARGET_DRVMAP_DISK

データベース・ドライブ・マッピングのターゲット MSCS ディスクを指定する任意指定パラメーター。ノードと同じグループに属さない MSCS ディスク上にデータベースが作成された場合、データベース区画はターゲット・ドライブ・マップ・ディスクに含まれます。このパラメーターが指定されない場合、データベース・ドライブ・マッピングは、 DB2DRVMP ユーティリティーを使用して、手作業で登録しなければなりません。

例:

```
TARGET_DRVMAP_DISK = Disk E:
```

単一区画データベース・システムにフェールオーバーをセットアップする

DB2MSCS ユーティリティーを単一区画データベース・システムに対して実行すると、 1 つの MSCS グループに DB2 およびすべての従属する MSCS リソース (IP アドレス、 Network Name、およびディスク) が入れられます。たとえば、単一区画データベース・システムの DB2MSCS.CFG ファイルの内容は次のようになります。

```
#
#  DB2MSCS.CFG for a single-partition database system
#
DB2_INSTANCE=DB2
CLUSTER_NAME=MSCS
GROUP_NAME=DB2 Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
```

```
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

2 つの単一区画データベース・システムに相互引き受け構成をセットアップする

2 つの単一区画データベース・システムを個別のマシンのそれぞれにセットアップして、片方のマシン上のデータベース・システムが障害を起こしても、もう一方の MSCS ノードで再開するようにできます。

この構成にフェールオーバー・サポートをセットアップするには、インスタンスが所有する各マシン上で DB2MSCS ユーティリティーを 1 度実行する必要があります。各データベース・システムごとに、構成ファイルを調整してください。

DB2 インスタンスの名前を DB2A および DB2B と想定すると、 DB2A インスタンスの DB2MSCS.CFG ファイルは次のようになります。

```
#
#  DB2MSCS.CFG for first single-partition database system
#
DB2_INSTANCE=DB2A
CLUSTER_NAME=MSCS
GROUP_NAME=DB2A Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

DB2A インスタンスの DB2MSCS.CFG ファイルは次のようになります。

```
#
#  DB2MSCS.CFG for second single-partition database system
#
DB2_INSTANCE=DB2B
CLUSTER_NAME=MSCS
GROUP_NAME=DB2B Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk F:
```

完全な例は、301ページの『例 - 相互引き受けに 2 つの単一区画インスタンスをセットアップする』を参照してください。

区分データベース・システムに複数の MSCS クラスターをセットアップする

複数の区分データベース・システムに対して DB2MSCS ユーティリティーを実行している場合、 1 つの MSCS グループが、システムに参加する各物理マシンごとに作成されます。 DB2MSCS.CFG ファイルには複数のセクションが含まれていなければならず、各セクションには GROUP_NAME パラメーターおよびそのグループに必要なすべての従属リソースに、異なる値がなければなりません。

さらに、各 MSCS グループのデータベース区画サーバーごとに、 DB2_NODE パラメーターを指定する必要があります。複数の論理ノードがある場合、各論理ノードには個別の DB2_NODE キーワードが必要です。

たとえば、 4 つのマシン上で 4 つのデータベース区画サーバーから構成される複数区画データベース・システムがあり、相互引き受け構成を使用して 2 つの MSCS クラスターを構成すると仮定します。 DB2MSCS.CFG 構成ファイルを次のようにセットアップします。

```
#
#  DB2MSCS.CFG for one partitioned database system with
#  multiple clusters
DB2_INSTANCE=DB2MPP
DB2_LOGON_USERNAME=db2user
DB2_LOGON_PASSWORD=xxxxxx
CLUSTER_NAME=MSCS1
# Group 1
GROUP_NAME=DB2 Group 1
DB2_NODE=0
IP_NAME=...

...
# Group 2
GROUP_NAME=DB2 Group 2
DB2_NODE=1
IP_NAME=...

...

CLUSTER_NAME=MSCS2
# Group 3
GROUP_NAME=DB2 Group 3
DB2_NODE=2
IP_NAME=...

...
# Group 4
GROUP_NAME=DB2 Group 4
```

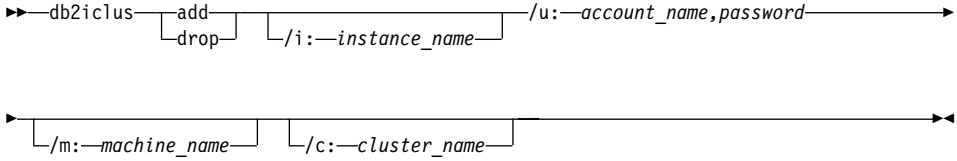
```
DB2_NODE=3
IP_NAME=...

...
```

完全な例は、304ページの『例 - 相互引き受けに 4 つのノードを持つ区分データベース・システムをセットアップする』を参照してください。

MSCS システムの保守

DB2MSCS ユーティリティーを実行すると、 MSCS クラスターのすべてのマシンにフェールオーバー・サポート用のインフラストラクチャーが作成されます。マシンからサポートを除去するには、 drop オプションを付けて **db2iclus** コマンドを使用します。マシンでのサポートを再度使用可能にしたい場合、 add オプションを使用します。

コマンド構文は、以下のとおりです。



ここで、

add
MSCS クラスターにフェールオーバー・サポートを追加して、使用可能にします。すると、DB2 リソース (データベース・サーバー) がこのマシンでフェールオーバーできます。

drop
MSCS クラスターからフェールオーバー・サポートを除去して、マシンから除去します。

/i: *instance_name*
インスタンスの名前。 (このパラメーターは、DB2INSTANCE 環境変数の設定をオーバーライドします。)

/u: *account_name*, *password*
DB2 サービスのログオン・アカウント名として使用されるドメイン・アカウント。たとえば、次のようなものがあります。

```
/u:domainA¥db2nt,password
```

このパラメーターは、 add オプションにのみ必要です。

/m:*machine_name*
MSCS クラスターに追加したい、または MSCS クラスターから除去したいコンピューター名。フェールオーバー・サポートを修正しているマシン以外のマシンからコマンドを実行する場合、このオプションを指定します。

/c: *cluster_name*
LAN と呼ばれる MSCS クラスター名。この名前は、MSCS クラスターが最初に作成されるときに指定されます。

フォールバックの考慮事項

デフォルトでは、グループは元の (障害を起こした) マシンにフォールバックするようには設定されません。フェールオーバー後にフォールバックするように手作業で DB2 グループを構成しない限り、フェールオーバーの原因が解決された後は代替 MSCS ノードで実行し続けます。

DB2 グループを、元のマシンに自動的にフォールバックするように構成する場合、 DB2 リソースを含む DB2 グループのすべてのリソースは、元のマシンが使用可能になるとすぐにフォールバックします。フォールバックの間にデータベース接続が存在している場合、 DB2 リソースをオフラインにすることはできず、フォールバック処理は失敗します。

フォールバック処理中にすべてのデータベース接続をデータベースから強制的にオフにしたい場合、 DB2_FALLBACK レジストリー変数を ON に設定します。この変数は、次のように設定してください。

```
db2set DB2_FALLBACK=ON
```

このレジストリー変数の設定後は、クラスター・サービスをリブートしたり再始動したりする必要はありません。

区分データベース環境で相互引き受け構成にデータベース・ドライブ・マッピングを登録する

区分データベース環境でデータベースを作成する場合、データベースの作成位置を示すドライブ文字を指定することができます。

注: 単一区画データベース環境にはデータベース・ドライブ・マッピングを設定しません。

CREATE DATABASE コマンドを実行するとき、指定されるドライブが、インスタンスに参加するすべてのマシンで同時に使用可能になっていることが期待されます。これは不可能なので、DB2 はデータベース・ドライブ・マッピングを使用して、各マシンごとに同じドライブに別の名前を割り当てます。

たとえば、DB2 という名前の DB2 インスタンスに、 2 つのデータベース区画サーバーが入っているとします。

```
NODE0 がマシン WOLF_NODE_0 で活動状態
NODE1 がマシン WOLF_NODE_1 で活動状態
```

また、共用ディスク F: が NODE0 として同じグループに所属しており、共用ディスク F: が NODE1 として同じグループに所属していると仮定します。

共用ディスク E にデータベースを作成するには、コマンドは次のようになります。

```
db2 create database mppdb on e:
```

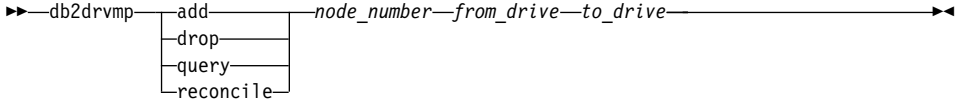
コマンドを成功させるために、ドライブ E: は両方のマシンで使用可能でなければなりません。相互引き受け構成では、各データベース区画サーバーが異なるマシン上で活動状態であり、クラスター・ディスク E: は 1 つのマシンにだけ使用可能です。この状態では、 CREATE DATABASE コマンドは常に失敗します。

この問題を解決するには、データベース・ドライブを次のようにマップしてください。

```
NODE0 については、マッピングはドライブ F: からドライブ E: へ
NODE1 については、マッピングはドライブ E: からドライブ F: へ
```

こうすると、 NODE0 のドライブ F: へのデータベース・アクセスがドライブ E: にマップされ、 NODE1 のドライブ E: へのデータベース・アクセスがドライブ F: にマップされます。ドライブ・マッピングを使用して、 CREATE DATABASE コマンドは NODE0 にドライブ E: で、および NODE1 にドライブ F: でデータベース・ファイルを作成します。

db2drvmp コマンドを使用してドライブ・マッピングをセットアップします。コマンド構文は、以下のとおりです。



パラメーターは、以下のとおりです。

add 新しいデータベース・ドライブ・マップを割り当てます。

drop 既存のデータベース・ドライブ・マップを除去します。

query データベース・マップの照会。

reconcile 登録内容が損傷を受けたときにデータベース・マップ・ドライブを修理します。詳細については、『データベース・ドライブ・マッピングの調整』を参照してください。

node_number ノード番号。このパラメーターは、add および drop 操作に必要です。

from_drive マップ元のドライブ文字。このパラメーターは、add および drop 操作に必要です。

to_drive マップ先のドライブ文字。このパラメーターは、add 操作に必要です。他の操作には適用しません。

F: から E: のデータベース・ドライブ・マッピングを NODE0 に設定したい場合、次のコマンドを使用できます。

```
db2drvmp add 0 F E
```

注: データベース・ドライブ・マッピングは表スペース、コンテナー、または他のデータベース記憶域オブジェクトには適用されません。

同様に、 E: から F: のデータベース・ドライブ・マッピングを NODE1 に設定したい場合、次のコマンドを出します。

```
db2drvmp add 1 E F
```

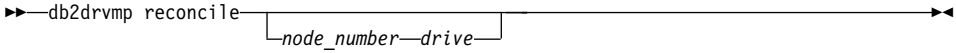
注: データベース・ドライブ・マッピングのセットアップ、または変更の効果はすぐに出るわけではありません。データベース・ドライブ・マッピングを活動化するには、 Cluster Administrator ツールを使用して DB2 リソースをオフラインにしてからオンラインにします。

DB2MSCS.CFG ファイルで TARGET_DRVMAP_DISK キーワードを使用すると、ドライブ・マッピングを自動的に行うようにすることができます。

データベース・ドライブ・マッピングの調整

データベースが実際にデータベース・ドライブ・マッピングを持つマシン上で作成されると、マップが隠しファイル中のドライブで保管されます。これはデータベース・ドライブが、データベースの作成後に除去されることのないようにするためのものです。たとえば、間違えてデータベース・ドライブ・マッピングを除去してしまう場合、データベース・ドライブ・マッピングを*調整* しま

す。マップを調整するには、データベースを含む各データベース区画サーバーごとに **db2drvmp reconcile** コマンドを実行します。コマンド構文は、以下のとおりです。



パラメーターは、以下のとおりです。

node_number 修理されるノードのノード番号。 *node_number* が指定されない場合、コマンドはすべてのノードへのマッピングを調整します。

drive 調整するドライブ。ドライブが指定されない場合、コマンドはすべてのドライブへのマッピングを調整します。

db2drvmp コマンドは、データベース区画サーバーが管理するデータベース区画でマシン上のすべてのドライブを走査し、データベース・ドライブ・マッピングを必要に応じてレジストリーに再度適用します。

例 - 相互引き受けに 2 つの単一区画インスタンスをセットアップする

この例の目的は、相互引き受け構成でフェールオーバー・サポートを使用し、2 つの単一区画データベース・インスタンスをセットアップすることです。この例では、4 つのサーバーが 2 つの MSCS クラスターに構成されます。相互引き受け構成を使用することにより、マシンのいずれかが障害を起こした場合、そのマシン用に構成されたデータベース・サーバーは、 MSCS ソフトウェアを使用して構成されたとおり、代替マシンにフェールオーバーし、その代替マシン上で実行します。

結果の構成には 2 つの MSCS クラスターがあります。各クラスターには、次のものがあります。

- 2 つのサーバー。それぞれ 64 MB のメモリーおよび 2 GB のローカル SCSI ディスクが 1 つあります。
- それぞれが 2 GB の 3 つの SCSI ディスクを持つ 1 つの SCSI ディスク・タワー。

加えて、各マシンには 1 つの 100X イーサネット・アダプター・カードがインストールされています。

各マシンには次のソフトウェアがインストールされています。

- MSCS 機能を持つ Windows NT Version 4.0 Enterprise Edition

- 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 7

結果のネットワーク構成は次のとおりです。

サーバー 1:	サーバー 2:
• マシン名: db2test1	• マシン名: db2test2
• TCP/IP ホスト名: db2test1	• TCP/IP ホスト名: db2test2
• IP アドレス: 9.9.9.1 (サブネット・マスク: 255.255.255.0	• IP アドレス: 9.9.9.2 (サブネット・マスク: 255.255.255.0
• MSCS クラスター名: ClusterA	• MSCS クラスター名: ClusterA

ネットワーク中の両方のマシンが TCP/IP で構成され、イーサネット 100 T-base Hub を使用して私用の LAN に接続されます。ドメイン・ネーム・サーバー (DNS) はありませんが、すべてのマシンにはローカル TCP/IP hosts ファイルがあります。このファイルには以下のエントリーが含まれています。

```
9.9.9.1 db2test1 # for Server 1
9.9.9.2 db2test2 # for Server 2
9.9.9.3 ClusterA # for MSCS ClusterA
9.9.9.4 db2tcp1 # for DB2 remote client connection to Server 1
9.9.9.5 db2tcp2 # for DB2 remote client connection to Server 2
```

仮のタスク

次のタスクを実行する前に、両方のマシンが同じドメイン、 DB2NTD に所属していると仮定します。

1. ローカル管理者グループのメンバーである DB2 の定義域アカウントを、DB2 が実行する予定のマシン上で作成します。すべてのタスクを実行するためのアカウントを使用します。
   - ユーザー名を db2nt に設定します。
   - パスワードを db2nt に設定します。
2. MSCS 機能をマシン db2test1 および db2test2 にインストールします。
   - MSCS クラスターに ClusterA という名前を付けます。
   - クラスター IP アドレスは 9.9.9.3 です。
   - 共用ディスク D: は MSCS ソフトウェアによって使用されます。
   - 共用ディスク E: および F: は DB2 によって使用されます。
3. マシン db2test1 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 バージョン 7 をインストールします。ローカル・ドライブである C:¥SQLLIB に、ソフトウェアをインストールします。

4. マシン db2test2 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 バージョン 7 をインストールします。ローカル・ドライブである C:¥SQLLIB に、ソフトウェアをインストールします。

次のステップは、各インスタンスごとに DB2MSCS.CFG ファイルをセットアップしてから、各インスタンスごとに DB2MSCS ユーティリティーを実行することです。

DB2MSCS ユーティリティーを実行する

db2test1 マシンをセットアップするには、次のタスクを実行してください。

1. マシン db2test1 では、ユーザー db2nt としてログオンします。パスワードは db2nt です。
2. まだ存在しなければ、DB2 インスタンス DB2A を作成します。インスタンスを作成するコマンドは次のとおりです。

   ```
   db2icrt DB2A
   ```

3. マシン db2test1 で DB2 インスタンスに DB2MSCS.CFG ファイルを設定します。

   ```
   #
   #  DB2MSCS.CFG for database system
   #  on machine db2test1
   DB2_INSTANCE=DB2A
   CLUSTER_NAME=ClusterA
   #
   # Group 1
   GROUP_NAME=DB2A Group
   IP_NAME=IP Address for DB2A
   IP_ADDRESS=9.9.9.4
   IP_SUBNET=255.255.255.0
   IP_NETWORK=ClusterA
   NETNAME_NAME=Network name for DB2A
   NETNAME_VALUE=DB2SRV1
   NETNAME_DEPENDENCY=IP Address for DB2A
   DISK_NAME=Disk E:
   INSTPROF_DISK=Disk E:
   ```

4. 以下のように DB2MSCS ユーティリティーを実行してください。

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

5. db2nt アカウントからログアウトします。
6. マシン db2test2 でユーザー db2nt としてログオンします。このユーザーはローカル管理者グループに属しています。パスワードは db2nt です。
7. まだ存在しなければ、DB2 インスタンス DB2B を作成します。インスタンスを作成するコマンドは次のとおりです。

   ```
   db2icrt DB2B
   ```

8. マシン db2test2 で DB2 インスタンスに DB2MSCS.CFG ファイルを設定します。

```
#
#  DB2MSCS.CFG for database system
#  on machine db2test2
DB2_INSTANCE=DB2B
CLUSTER_NAME=ClusterA
#
# Group 1
GROUP_NAME=DB2B Group
IP_NAME=IP Address for DB2B
IP_ADDRESS=9.9.9.5
IP_SUBNET=255.255.255.0
IP_NETWORK=ClusterA
NETNAME_NAME=Network name for DB2B
NETNAME_VALUE=DB2SRV2
NETNAME_DEPENDENCY=IP Address for DB2B
DISK_NAME=Disk F:
INSTPROF_DISK=Disk F:
```

9. 以下のように DB2MSCS ユーティリティーを実行してください。

```
db2mscs -f:DB2MSCS.CFG
```

10. db2nt アカウントからログアウトします。

例 - 相互引き受けに 4 つのノードを持つ区分データベース・システムをセットアップする

この例の目的は、相互引き受け構成でフェールオーバー・サポートを使用し、4 つのノードを持つ単一区画データベース・インスタンスをセットアップすることです。この例では、4 つのサーバーが 2 つの MSCS クラスターに構成されます。相互引き受け構成を使用することにより、マシンのいずれかが障害を起こした場合、そのマシン用に構成されたデータベース区画サーバーは、MSCS ソフトウェアを使用して構成されたとおり、代替マシンにフェールオーバーし、その代替マシン上で論理ノードとして実行します。

結果の構成には 2 つの MSCS クラスターがあります。各クラスターには、次のものがあります。

- 2 つのサーバー。それぞれ 64 MB のメモリーおよび 2 GB のローカル SCSI ディスクが 1 つあります。
- それぞれが 2 GB の 3 つの SCSI ディスクを持つ 1 つの SCSI ディスク・タワー。

加えて、各マシンには 1 つの 100X イーサネット・アダプター・カードがインストールされています。

各マシンには次のソフトウェアがインストールされています。

- MSCS 機能を持つ Windows NT Version 4.0 Enterprise Edition
- 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 7

結果のネットワーク構成は次のとおりです。

サーバー 1: • マシン名: db2test1 • TCP/IP ホスト名: db2test1 • IP アドレス: 9.9.9.1 (サブネット・マスク: 255.255.255.0 • MSCS クラスター名: ClusterA	サーバー 2: • マシン名: db2test2 • TCP/IP ホスト名: db2test2 • IP アドレス: 9.9.9.2 (サブネット・マスク: 255.255.255.0 • MSCS クラスター名: ClusterA
サーバー 3: • マシン名: db2test3 • TCP/IP ホスト名: db2test3 • IP アドレス: 9.9.9.3 (サブネット・マスク: 255.255.255.0 • MSCS クラスター名: ClusterB	サーバー 4: • マシン名: db2test4 • TCP/IP ホスト名: db2test4 • IP アドレス: 9.9.9.4 (サブネット・マスク: 255.255.255.0 • MSCS クラスター名: ClusterB

ネットワーク中のすべてのマシンが TCP/IP で構成され、イーサネット 100 T-base Hub を使用して私用の LAN に接続されます。ドメイン・ネーム・サーバー (DNS) はありませんが、すべてのマシンにはローカル TCP/IP hosts ファイルがあります。このファイルには以下のエントリーが含まれています。

```
9.9.9.1 db2test1 # for Server 1
9.9.9.2 db2test2 # for Server 2
9.9.9.3 db2test3 # for Server 3
9.9.9.4 db2test4 # for Server 4
9.9.9.5 ClusterA # for MSCS Cluster 1
9.9.9.6 ClusterB # for MSCS Cluster 2
9.9.9.7 db2tcp # for DB2 remote client connection
```

仮のタスク

次のタスクを実行する前に、4 つのすべてのマシンが同じドメイン、 DB2NTD に所属していると仮定します。

1. ローカル管理者グループのメンバーである DB2 の定義域アカウントを、DB2 が実行する予定のマシン上で作成します。すべてのタスクを実行するためのアカウントを使用します。
   - ユーザー名を db2nt に設定します。

- パスワードを db2nt に設定します。

2. *"password never expires"* 特性で、 2 番目の定義域アカウントを作成します。このアカウントは、DB2 サービスと関連付けられます。
   - ユーザー名を db2mpp に設定します。
   - パスワードを db2mpp に設定します。
3. MSCS 機能をマシン db2test1 および db2test2 にインストールします。
   - MSCS クラスターに ClusterA という名前を付けます。
   - クラスター IP アドレスは 9.9.9.5 です。
   - 共用ディスク D: は MSCS ソフトウェアによって使用されます。
   - 共用ディスク E: および F: は DB2 によって使用されます。
4. MSCS 機能をマシン db2test3 および db2test4 にインストールします。
   - MSCS クラスターに ClusterB という名前を付けます。
   - クラスター IP アドレスは 9.9.9.6 です。
   - 共用ディスク D: は MSCS ソフトウェアによって使用されます。
   - 共用ディスク E: および F: は DB2 によって使用されます。
5. DB2 エンタープライズ拡張エディションをマシン db2test1 にインストールします。
   - *"This machine will be the instance-owning database partition server"* オプションを選択します。
   - DB2 サービスのアカウントは db2mpp です。パスワードは db2mpp です。
   - ローカル・ドライブである C:¥SQLLIB に、ソフトウェアをインストールします。
6. DB2 エンタープライズ拡張エディションをマシン db2test2、db2test3、および db2test4 にインストールします。
   - *"This machine will be a new node on an existing partitioned database system"* オプションを選択します。
   - db2test1 をインスタンス所有のマシンとして選択します。
   - DB2 サービスのアカウントは db2mpp です。パスワードは db2mpp です。
   - ローカル・ドライブである C:¥SQLLIB に、ソフトウェアをインストールします。

次のステップは、DB2MSCS.CFG ファイルをセットアップし、 DB2MSCS ユーティリティーを実行することです。

DB2MSCS ユーティリティーを実行する

db2test1 マシンをセットアップするには、次のタスクを実行してください。

1. ユーザー db2nt としてログオンします。このユーザーはローカル管理者グループに属しています。パスワードは db2nt です。
2. DB2MSCS.CFG ファイルをセットアップします。

```
#
#  DB2MSCS.CFG for one partitioned database system with
#  multiple MSCS clusters
DB2_INSTANCE=DB2MPP
CLUSTER_NAME=ClusterA
DB2_LOGON_USERNAME=db2mpp
DB2_LOGON_PASSWORD=db2mpp
# Group 1
# for DB2 node 0
GROUP_NAME=DB2NODE0
DB2_NODE=0
IP_NAME=IP Address for DB2
IP_ADDRESS=9.9.9.7
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
NETNAME_NAME=Network name for DB2
NETNAME_VALUE=DB2WOLF
NETNAME_DEPENDENCY=IP Address for DB2
DISK_NAME=Disk E:
INSTPROF_DISK=Disk E:
#

# Group 2
# for DB2 node 1
GROUP_NAME=DB2NODE1
DB2_NODE=1
DISK_NAME=Disk F:
#

CLUSTER_NAME=ClusterB
# Group 3
# for DB2 node 2
GROUP_NAME=DB2NODE2
DB2_NODE=2
DISK_NAME=Disk E:

#
# Group 4
# for DB2 node 3
GROUP_NAME=DB2NODE3
DB2_NODE=3
DISK_NAME=Disk F:
```

3. 以下のように DB2MSCS ユーティリティーを実行してください。

```
db2mscs -f:DB2MSCS.CFG
```

4. db2nt アカウントからログアウトします。

最終ステップは 2 つの MSCS クラスターに、データベース・ドライブ・マッピングを登録することです。

ClusterA にデータベース・ドライブ・マッピングを登録する

MSCS クラスター ClusterA にデータベース・ドライブ・マッピングを登録するには、次のタスクを実行します。

1. マシン db2test1 では、ユーザー db2mpp としてログオンします。これは DB2 サービスと関連付けられるアカウントです。パスワードは db2mpp です。
2. データベース・ドライブ・マッピングを登録するには、次のコマンドを入力します。

```
db2drvmp add 0 F E

db2drvmp add 1 E F
```

3. すべての DB2 リソースをオフラインにしてからオンラインにします。

ClusterB にデータベース・ドライブ・マッピングを登録する

MSCS クラスター ClusterB にデータベース・ドライブ・マッピングを登録するには、次のタスクを実行します。

1. マシン db2test3 では、ユーザー db2mpp としてログオンします。これは DB2 サービスと関連付けられるアカウントです。パスワードは db2mpp です。
2. データベース・ドライブ・マッピングを登録するには、次のコマンドを入力します。

```
db2drvmp add 2 F E
db2drvmp add 3 E F
```

3. すべての DB2 リソースをオフラインにしてからオンラインにします。

MSCS 環境で DB2 を管理する

MSCS クラスターを使用している場合、DB2 インスタンスには毎日の操作、データベースの展開、およびデータベース構成に関係する付加的な計画を必要とします。 DB2 を MSCS ノードで透過的に実行するには、付加的な管理用タスクを実行する必要があります。すべての DB2 従属のオペレーティング・システム・リソースは、すべての MSCS ノードで使用可能でなければなりません。これらのオペレーティング・システムのいくつかは、MSCS の効力範囲外です。つまり、MSCS リソースとして定義できません。同じオペレーティン

グ・システム・リソースがすべての MSCS ノードで使用可能であるように、各システムを確実に構成することが必要です。以下のセクションでは、行う必要がある付加的な作業を説明します。

DB2 リソースの開始および停止

DB2 リソースは、 Cluster Administrator ツールから開始したり停止したりするべきです。いくつかのメカニズムは **db2start** コマンド、および「コントロール パネル」からの**「サービス」**オプションなどの DB2 インスタンスを開始するのに使用できます。しかし、DB2 が Cluster Administrator から開始しない場合、 MSCS ソフトウェアは DB2 インスタンスの状態を感知しません。 DB2 インスタンスが Cluster Administrator を使用して開始され、 **db2stop** コマンドを使用して停止される場合、 MSCS ソフトウェアは **db2stop** コマンドをソフトウェア障害として解釈し、 DB2 を再始動しようとします。 (現在の MSCS インターフェースは、*リソース状態* の通知をサポートしません。)

同様に、**db2start** を使用して DB2 インスタンスを開始する場合、 MSCS はリソースがオンライン上にあるかどうか検出できません。データベース・サーバーが失敗すると、 MSCS は DB2 リソースをクラスター中のフェールオーバー・マシン上でオンラインにすることはありません。

3 つの操作が DB2 インスタンスに適用します。

オンライン

この操作は、**db2start** コマンドの使用と同じです。 DB2 がすでに活動状態である場合、この操作を使用して、 MSCS に DB2 が活動状態であることを単に通知することができます。この操作中にエラーが起きると、Windows NT 事象ログに書き込まれます。

オフライン

この操作は、**db2stop** コマンドの使用と同じです。インスタンスに活動状態の接続がある場合、この操作は失敗します。これは **db2stop** の動作と一貫します。

リソースの失敗

この操作は、**force** オプションを指定した **db2stop** コマンドの使用と同じです。 DB2 はすべてのアプリケーションを DB2 システムから切断し、すべてのデータベース・サーバーを停止します。

スクリプトの実行

スクリプトは、DB2 リソースがオンラインになった前でも、後でも実行できます。これらのスクリプトは、DB2INSTPROF 環境変数に指定される、インスタンス・プロファイル・ディレクトリーの中になければなりません。このディレ

クトリーは、 **db2icrt** コマンドの -p パラメーターによって指定されるディレクトリー・パスです。次のコマンドを発行して、この値を得ることができます。

```
db2set -i:instance_name DB2INSTPROF
```

このファイル・パスは、インスタンス・ディレクトリーがすべてのクラスター・ノードで使用可能になるように、クラスター化ディスク上になければなりません。

これらのスクリプト・ファイルは必須ではなく、インスタンス・ディレクトリーで検出される場合にのみ実行されます。これらはバックグラウンドで MSCS クラスター・サービスによって立ち上げられます。スクリプト・ファイルは、スクリプト・ファイル内のコマンドから戻される情報を取り込むために標準出力をリダイレクトする必要があります。出力は画面には表示されません。

区分データベース環境では、デフォルトで、インスタンス中のすべてのデータベース区画サーバーが同じスクリプトを使用します。インスタンス中の異なるデータベース区画サーバーを見分ける必要がある場合、 DB2NODE 環境変数の異なる割り当てをターゲット・ノード番号に使用してください (たとえば、 db2cpre.bat および db2cpost.bat ファイルで IF ステートメントを使用する)。

DB2 リソースをオンラインにする前にスクリプトを実行する

DB2 リソースをオンラインにする*前に* スクリプトを実行したい場合、スクリプトの名前を db2cpre.bat *にしなければなりません。* DB2 は Windows NT コマンド行プロセッサー (CLP) からこのバッチ・ファイルを立ち上げる関数を呼び出し、 DB2 リソースがオンラインになる前に CLP が実行を完了するのを待機します。このバッチ・ファイルは、 DB2 データベース・マネージャー構成の修正などのタスクをこのバッチ・ファイルに使用することができます。フェールオーバー・システムが無理である場合、データベース・マネージャーのパラメーター値をいくつか変更する必要があるかもしれませんし、 DB2 が使用するシステム・リソースを削減することが必要です。

db2cpre.bat スクリプトに配置されたコマンドは、同期で実行されるべきです。そうしないと、DB2 リソースは、スクリプト中のすべてのタスクが完了する前にオンラインになり、その結果予期しない動作が起きる場合があります。特に、**db2cmd** を db2cpre.bat スクリプトで呼び出すべきではありません。これは、 DB2 コマンドを非同期に **db2cmd** プログラムに対して実行する他のコマンド・プロセッサーを代わりに立ち上げてしまうからです。

db2cpre.bat スクリプトで DB2 CLP コマンドを使用したい場合、コマンドはファイル中に配置し、 DB2 コマンド行プロセッサー用に DB2 環境を初期化するプログラム内から CLP バッチ・ファイルとして実行する必要があり、次に DB2 コマンド行プロセッサーの完了を待機します。次に例を示します。

```
#include <windows.h>

int WINAPI DB2SetCLPEnv_api(DWORD pid);

void main ( int argc, char *argv [ ] )
{
      STARTUPINFO         startInfo   = {0};
      PROCESS_INFORMATION pidInfo     = {0};
      char     title  [32]   = "Run Synchronously";
      char     runCmd [64]  =
                             "DB2 -z c:¥¥run.out -tvf c:¥¥run.clp";

/* Invoke API to set up a CLP Environment */
      if ( DB2SetCLPEnv_api (GetCurrentProcessId ()) == 0 ) 1
      {
         startInfo.cb          = sizeof(STARTUPINFO);
         startInfo.lpReserved  = NULL;
         startInfo.lpTitle     = title;
         startInfo.lpDesktop   = NULL;
         startInfo.dwX         = 0;
         startInfo.dwY         = 0;
         startInfo.dwXSize     = 0;
         startInfo.dwYSize     = 0;
         startInfo.dwFlags     = 0L;
         startInfo.wShowWindow = SW_HIDE;
         startInfo.lpReserved2 = NULL;
         startInfo.cbReserved2 = 0;
               if ( CreateProcessA( NULL,
                             runCmd, 2
                             NULL,
                             NULL,
                             FALSE,
                             NORMAL_PRIORITY_CLASS CREATE_NEW_CONSOLE,
                             NULL,
                             NULL,
                             &startInfo,
                             &pidInfo ) )
         {
            WaitForSingleObject (pidInfo.hProcess, INFINITE);
            CloseHandle (pidInfo.hProcess);
            CloseHandle (pidInfo.hThread);
         }
      }
      return;
}
```

1 API DB2SetCLPEnv_api はインポート・ライブラリー DB2API.LIB によって解決されます。この API は、呼び出される CLP コマンドを許可

する環境を設定します。このプログラムが db2cpre.bat スクリプトから呼び出される場合、コマンド・プロセッサーは CLP コマンドの完了を待機します。

2 runCmd は DB2 CLP コマンドを含むスクリプト・ファイルの名前です。

db2clpex.exe という名前のサンプル・プログラムが、 DB2 インストール・パスの MISC サブディレクトリーに見つかることがあります。この実行可能ファイルは提供される例に類似していますが、 DB2 CLP コマンドをコマンド行引き数として受け入れます。このサンプル・プログラムを使用したい場合、 BIN サブディレクトリーにコピーしてください。この実行可能ファイルは、 db2cpre.bat スクリプトで次のように使用できます (INSTHOME はインスタンス・ディレクトリー)。

```
db2clpex "DB2 -Z INSTHOME¥pre.log -tvf INSTHOME¥pre.clp"
```

すべての DB2 ATTACH コマンドまたは CONNECT ステートメントは、明示的にユーザーを指定する必要があります。そうしないと、クラスター・サービスと関連するユーザー・アカウントのもとで実行されます。 CLP スクリプトはまた TERMINATE コマンドで完了して、 CLP バックグラウンド・プロセスを終了する必要があります。

以下に、db2cpre.bat ファイルの例を示します。

```
db2cpre.bat : 1
------------------------
db2clpex "db2 -z INSTHOME¥pre-%DB2NODE%.log -tvf INSTHOME¥pre.clp" 2 - 5
------------------------

PRE.CLP 6
------------------------
update dbm cfg using MAXAGENTS 200;
get dbm cfg;
terminate;
------------------------
```

1 db2cpre.bat スクリプトは、クラスター・サービスと関連するユーザー・アカウントの下で実行します。 DB2 処置が必要な場合、クラスター・サービスと関連するユーザー・アカウントは、 DB2 で定義されているように有効な SQL 識別子でなければなりません。

2 INSTHOME はインスタンス・ディレクトリーです。

3 ログ・ファイルの名前は、両方の論理ノードが同時にオンラインになったときにファイルの競合を避けるため、各ノードごとに異なっていなければなりません。

4 db2clpex.exe は、コマンド行引き数を使用して、呼び出す CLP コマンドを指定するサンプル・プログラムです。

5 db2clpex.exe サンプル・プログラムは、すべての MSCS クラスター・ノードで使用可能にされなければなりません。

6 この例の CLP コマンドは、エージェントの数に制限を設定します。

DB2 リソースをオンラインにした後にスクリプトを実行する

DB2 リソースをオンラインにした後に スクリプトを実行したい場合、スクリプトの名前を db2cpost.bat にしなければなりません。スクリプトは、DB2 リソースが正常にオンラインになってから、 MSCS から非同期で実行されます。 **db2cmd** コマンドをこのスクリプトで使用し、 DB2 CLP スクリプト・ファイルを実行できます。 **db2cmd** コマンドの -c パラメーターを使用して、ユーティリティーがタスクの完了時にすべてのウィンドウをクローズするように指定します。次に例を示します。

```
db2cmd -c db2 -tvf mycmds.clp
```

-c パラメーターは、 **db2cmd** コマンドに対する最初の引き数でなければなりません。これにより、バックグラウンドのコマンド・プロセッサーを孤立しないようにできます。

db2cpost.bat スクリプトは、 DB2 リソースがフェールオーバーし、活動状態になった直後にデータベース活動を実行したい場合に役立ちます。たとえば、ユーザー・アクセス用にプライム状態になるように、インスタンス中のデータベースを再始動または活動化することができます。

以下に、db2cpost.bat スクリプトの例を示します。

```
db2cpost.bat 1
------------------------
db2cmd -c db2 -z INSTHOME¥post-%DB2NODE%.log -tvf INSTHOME¥post.clp 2 - 4
------------------------

POST.CLP 5
------------------------
restart database SAMPLE;
connect reset;
activate database SAMPLE;
terminate;
------------------------
```

1 db2cpost.bat スクリプトは、クラスター・サービスと関連するユーザー・アカウントの下で実行します。 DB2 処置が必要な場合、クラスター・サービスと関連するユーザー・アカウントは、 DB2 で定義されているように有効な SQL 識別子でなければなりません。

2 INSTHOME はインスタンス・ディレクトリーです。

3 ログ・ファイルの名前は、両方の論理ノードが同時にオンラインになったときにファイルの競合を避けるため、各ノードごとに異なっていなければなりません。

4 **db2cmd** コマンドを使用できます。 db2cpost.bat スクリプトが非同期に実行できるためです。 -c パラメーターを使用して、コマンド・プロセッサーを終了する必要があります。

5 この例の CLP スクリプトには、データベースを再始動および活動化させるコマンドが含まれています。このスクリプトは、データベース・マネージャーの起動直後、データベースを活動状態に戻します。区分データベース・システムでは、複数の DB2 リソースが同時にオンラインになるので、 ACTIVATE DATABASE コマンドを除去する必要があります。 RESTART DATABASE コマンドは、別のノードがデータベースを活動化するために失敗することがあります。これが起きた場合、スクリプトを再実行してデータベースが確実に、正しく再始動するようにしてください。

データベースの考慮事項

データベースの作成時、データベース・パスが共用ディスクを必ず参照するようにしてください。これによって、データベースがすべての MSCS ノードで参照可能になります。すべてのログと他のデータベース・ファイルも、DB2 が正常にフェールオーバーするように、クラスター・ディスクを参照することが必要です。これらのステップを実行しないと、ファイルが削除されたまたは使用不可能になった DB2 を参照するので、 DB2 システム障害が発生します。

また、データベース・マネージャーおよびデータベース構成パラメーターの設定が、必ず DB2 の使用するシステム・リソース量がどの MSCS ノードでもサポートされるようにしてください。 *autorestart* データベース構成パラメーターは、フェールオーバー上の最初のデータベース接続がデータベースを一貫した状態にするように、 ON に設定するべきです。 (*autorestart* の省略時設定は ON です。) また、 db2cpost.bat スクリプトを使用してデータベースを再始動および活動化することによっても、データベースを作動可能状態にすることができます。この方法をお勧めします。*autorestart* には従属関係がなく、データベースはユーザー接続要求とは関係なく作動可能状態になるためです。

ユーザーおよびグループ・サポート

DB2 はユーザー認証およびグループ・サポートに関して、 Windows NT に依存しています。 DB2 インスタンスをシームレス (直接) に 1 つの MSCS ノ

ードから別のノードにフェールオーバーするには、各 MSCS ノードを同じ Windows NT セキュリティー・データベースにアクセスさせる必要があります。これは、Windows NT ドメイン・セキュリティーを使用して達成できます。

すべての DB2 ユーザーとグループをドメイン・セキュリティー・データベースで定義してください。 MSCS ノードは、このドメインのメンバーでなければならないか、またはドメインがトラステッド・ドメインであることが必要です。その場合、DB2 はドメイン・セキュリティー・データベースを、 DB2 を実行しているノードに関係なく認証およびグループ・サポートに使用します。

ローカル・アカウントを使用している場合、アカウントは各 MSCS ノードで複製される必要があります。このアプローチは、エラーを起こしやすく、二重のメンテナンスが必要なので、あまりお勧めできません。

すべての MSCS ノードが同じ DCE セルのクライアントである場合、 DCE セキュリティーもサポートされる認証モードです。

MSCS サービスを、DB2 命名規則に従うユーザー・アカウントと関連付ける必要があります。これによって MSCS サービスは、DB2 に対して処置をとることができます。このことは `db2cpre.bat` および `db2cpost.bat` スクリプトで必要になることがあります。

Windows NT ユーザーおよびグループ・サポートについての詳細は、 *管理の手引き: インプリメンテーション* の『DB2 (Windows NT 版) での DB2 ユーザー認証』を参照してください。

通信に関する考慮事項

DB2 は、MSCS 環境で 2 つの LAN プロトコルをサポートします。

- TCP/IP
- NetBIOS

TCP/IP は、クラスター・リソース・タイプであるためサポートされます。DB2 が区分データベース・システムの通信プロトコルとして TCP/IP を使用できるようにするには、 IP アドレス・リソースを作成して、リモート・アプリケーションに調整ノードとして使用したいデータベース区画サーバーを表す DB2 リソースとして同じグループに配置してください。次に Cluster Administrator ツールを使用して従属関係を作成し、 DB2 リソースが開始される前に確実に IP リソースがオンラインになるようにします。これで DB2 クライアントは TCP/IP ノード・ディレクトリー・エントリーをカタログ化し、この TCP/IP アドレスを使用できます。

svcename データベース・マネージャー構成パラメーターと関連する TCP/IP ポートは、インスタンスに参加するすべてのマシン上で DB2 インスタンスが使用するために予約しておく必要があります。ポート番号と関連したサービス名も、すべてのマシン上の services ファイルで同じでなければなりません。

NetBIOS はクラスター・リソースによってサポートされませんが、 NetBIOS を LAN プロトコルとして使用できます。これはプロトコルが LAN 上で NetBIOS 名を確実に固有のものにするからです。 DB2 が NetBIOS 名を登録すると、NetBIOS は名前が LAN で使用されていないことを確かめます。フェールオーバーのシナリオでは、DB2 がシステムからシステムへと移動されると、 DB2 が使用する *nname* は、 MSCS クラスターで 1 つのパートナー・マシンから登録解除され、別のマシンに登録されます。

DB2 NetBIOS サポートは NetBIOS Frames (NBF) を使用します。このプロトコル・スタックは、異なる論理アダプター番号 (LANA) と関連付けることができます。サーバーに一貫した NetBIOS アクセスを保証するために、 NBF プロトコル・スタックと関連した LANA はすべてのクラスター・ノードで同じでなければなりません。これは、「コントロール パネル」の**「ネットワーク」**オプションを使用して構成できます。 NBF を LANA 0 と関連付けることは DB2 の省略時設定で予期されているので、そうする必要があります。

システム時刻の考慮事項

DB2 はシステム時刻を使用して特定の操作にタイム・スタンプを付けます。DB2 フェールオーバーに参加するすべての MSCS ノードには、システム時刻ゾーン、および DB2 がすべてのマシンで一貫した動作をするように同期化されたシステム時刻が必要です。

「コントロール パネル」ダイアログの**「日付と時刻」**オプションを使用してシステム時刻を設定します。 MSCS には、 MSCS ノードがクラスターを形式化するために結合する時に日時を同期化するタイム・サービスがあります。しかし、タイム・サービスは単に 12 時間ごとに時刻を同期化するだけなので、時刻が 1 つのシステム上で変更され、時刻が同期化される前に DB2 がフェールオーバーする場合に問題が起きることがあります。

MSCS クラスター・ノードの 1 つで時刻が変更される場合、次のコマンドを使用して他のクラスター・ノードで手作業で同期化されなければなりません。

```
net time /set /y ¥¥remote_node
```

remote_node には、クラスター・ノードのマシン名が入ります。

区分データベース環境の管理サーバーとコントロール・センターに関する考慮事項

DB2 管理サーバーは、 DB2 ユニバーサル・データベースのインストール中に (任意選択で) 作成されます。これは区分データベース・システムではありません。コントロール・センターは、管理サーバーが提供するサービスを使用して、 DB2 インスタンスおよびデータベースを管理します。

区分データベース・システムでは、 1 つの DB2 インスタンスが複数の MSCS ノードに常駐できます。これは、どの MSCS ノードで DB2 インスタンスが活動状態になっているかに関係なく、インスタンスがずっとアクセス可能になっているように、コントロール・センターの下の複数のシステム上で DB2 インスタンスがカタログ化されなければならないということを暗示しています。

管理サーバーのインスタンス・ディレクトリーは共用ではありません。管理サーバーのディレクトリーのすべてのユーザー定義のファイルを、すべての MSCS ノードに対して二重化して、すべての MSCS ノードに同じレベルの管理を提供しなければなりません。特に、ユーザー・スクリプトおよびスケジュールされた実行可能ファイルを、すべてのノードで使用可能にしなければなりません。また、スケジュールされた活動を MSCS クラスターのすべてのマシン上でもスケジュールしなければなりません。

別の方法として、すべてのマシン上で管理サーバーを重複する代わりに、管理サーバーをフェールオーバーさせることも可能です。次の例の目的では、クラスター中に 2 つの MSCS ノードがあり、 MACH0 および MACH1 という名前だと仮定します。 MACH0 は、管理サーバーが使用するクラスター・ディスクにアクセスできます。さらに、MACH0 および MACH1 両方に管理サーバーがあるとします。以下のステップを実行して、管理サーバーを高度に使用可能にします。

1. 各マシンで **db2admin stop** コマンドを呼び出し、両方のマシンで管理サーバーを停止します。
2. すべての管理クライアント・マシンで、 UNCATALOG NODE コマンドを使用し、 MACH0 および MACH1 で管理サーバーへのすべての参照のカタログ化を解除します。 (LIST NODE DIRECTORY コマンドをクライアント・マシン上で使用して、管理サーバーへの参照が残っていないか判別できます。)
3. MACH1 から **db2admin drop** コマンドを呼び出し、 MACH1 から管理サーバーを除去します。 (このステップは、両方のマシン上に管理サーバーがある場合にのみ実行します。)

4. MACH0 から **db2admin** コマンドを発行し、管理サーバーの名前を決定します。 (省略時の名前は DB2DAS00 です。)
5. DB2MSCS ユーティリティーを使用して管理サーバーにフェールオーバー・サポートをセットアップします。これには、IP およびディスク・リソースで従属関係を持つ DB2DAS00 という名前の MSCS で DB2 リソースを作成することが必要です。 (相互引き受け構成がある場合、 NODE0 のための DB2 リソースを保持するグループにリソースを入れます。) このリソースは、管理サーバーをサポートする MSCS リソースとして使用されます。 DB2MSCS.ADMIN ファイルは次のとおりです。

```
#
# db2mscs.admin for Administration Server
# run db2mscs -f:db2mscs.admin
#
DB2_INSTANCE=DB2DAS00
CLUSTER_NAME=CLUSTERA
DB2_LOGON_USERNAME=db2admin
DB2_LOGON_PASSWORD=db2admin
# put Administration server in the same group as DB2 Node 0
GROUP_NAME=DB2NODE0 1
DISK_NAME=DISK E:
INSTPROF_DISK=DISK E:
IP_NAME= IP Address for Administration Server
IP_ADDRESS=9.9.9.8
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
```

   1 このグループは、既存のグループと同じものかもしれません。このように、インスタンス・プロファイル・ディレクトリーには付加的なディスクは必要ありません。

6. MACH1 で、次のコマンドを呼び出し、DB2DAS00 を管理サーバーとして設定します。

```
db2set -g db2adminserver=DB2DAS00
```

7. MACH0 で、サービス・プログラムを介して DB2DAS00 の開始プロパティーを変更し、自動でなく手作業で立ち上げることができるようにします。これは、DB2DAS00 がこの時点で MSCS に制御されているためです。

管理サーバーがフェールオーバー用に使用可能になると、すべてのリモート・アクセスは管理サーバーとの通信に MSCS IP リソースを使用しなければなりません。管理サーバーは、これで次の特性を持つことになります。

- 管理サーバーのインスタンス・ディレクトリーは管理サーバーでフェールオーバーします。
- クライアントは、活動状態の MSCS ノードに関係なく、単一ノードをカタログ化して管理サーバーと通信するだけですみます。

- 管理サーバーに対するジョブのスケジュールは 1 度だけですみます。
- ローカル・インスタンスは、管理サーバーがローカル・インスタンスと同じ MSCS ノードで活動状態のときにだけ管理サーバーによって制御されます。
- 管理サーバーは、クラスター・サービスが活動状態でない場合はアクセスできません。

制限と制約事項

MSCS 環境で DB2 を実行するには、次の制限があります。

- 共用ディスクに、両方の MSCS ノードで通用する同じ物理ディスク番号がない限り、共用ディスクで物理入出力を使用することはできません。ディスクは区画識別子を使用してアクセスされるので、論理入出力は使用できます。
- MSCS サポートにすべての DB2 リソースを構成することが必要です。そうしないと、 DB2 実行時にシステム・エラーが発生します (DB2 はシステム・リソースがないと適切に操作しません)。たとえば、データベース・ログが MSCS 共用ディスク上にないと、 DB2 はデータベースを再始動できません。
- Cluster Administrator ツールから DB2 インスタンスを管理する必要があります。 MSCS はデータベース・マネージャーをソフトウェア矛盾として開始、または停止するのに使用される他のメカニズムを表示します。たとえば、DB2 を開始するのに MSCS を、停止するのに **db2stop** コマンドを使用すると、 MSCS はこれをソフトウェア障害として検出し、インスタンスを再始動します。これは、コントロール・センターを使用して DB2 を開始および停止してはならないということも示唆しています。
- DB2 をアンインストールするには、まず MSCS を停止する必要があります。

第14章 DB2 と、Sun Cluster 2.2 での高可用性

この章では、 DB2 が Sun Cluster 2.x (SC2.x) と協働して高可用性を実現する方法について詳しく説明しています。また、これらの 2 つのソフトウェア製品間で仲介者として機能する、高可用性エージェントについても説明しています (図59 を参照してください)。

図 59. DB2、Sun Cluster 2.x、および高可用性

高可用性

データ・サービスをホストするコンピューター・システムには、数多くの異なる構成要素があり、各構成要素にはそれに関連した「平均故障間隔」(MTBF) があります。 MTBF は、構成要素が使用可能な状態にある平均時間です。質の高いハード・ディスクの MTBF は、100 万時間の次数 (約 114 年) です。これは長期間に思えますが、200 個のディスクのうち 1 つは、 6 か月以内に故障する可能性があります。

データ・サービスの可用性を上げるための方法は多数ありますが、最も一般的なものは HA クラスターです。クラスターは、高可用性で使用される場合、複数のマシン、私設ネットワーク・インターフェースのセット、 1 つまたは複数の公衆ネットワーク・インターフェース、およびいくつかの共用ディスクから成ります。この特別な構成により、データ・サービスを 1 つのマシンから別のマシンに移動させることが可能になります。データ・サービスをクラスター内の別のマシンに移動することによって、引き続きそのデータにアクセスすることができます。データ・サービスを 1 つのマシンから別のマシンに移動するこ

とをフェールオーバー といいます。これは、図60 で図示されています。

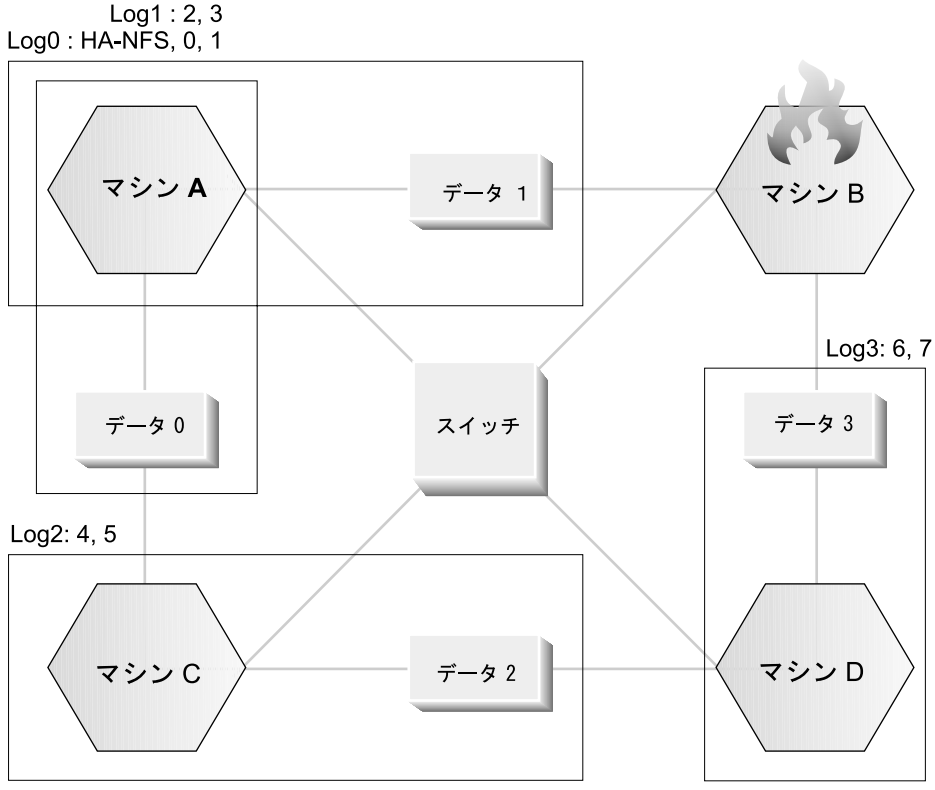


図 60. フェールオーバー

私設ネットワーク・インターフェースは、クラスター内のマシン間でハートビート・メッセージおよび制御メッセージを送信するために使用します。公衆ネットワーク・インターフェースは、 HA クラスターのクライアントと直接通信するために使用します。 HA クラスター内のディスクはクラスター内の 2 つ以上のマシンと接続されており、片方のマシンで障害が起きた場合に別のマシンがそのディスクにアクセスできるようになっています。

HA クラスター上で実行されているデータ・サービスには、 1 つ以上の論理公衆ネットワーク・インターフェースと一連のディスクが関連付けられています。 HA データ・サービスのクライアントは、 TCP/IP 経由でデータ・サービスの論理ネットワーク・インターフェースにのみ接続します。フェールオーバーが起きると、データ・サービスは、論理ネットワーク・インターフェースおよびディスクのセットとともに、別のマシンに移動します。

HA クラスターの利点の 1 つは、サポート・スタッフの援助なしでデータ・サービスを回復できることと、いつでもそれを行えることです。もう 1 つの利点は、冗長度です。マシン自体を含め、クラスター内のすべての部分に冗長度があります。クラスターは、どんな単一の障害が起きても、存続することができます。

高可用性データ・サービスはそれぞれ性質がかなり異なる場合がありますが、いくつかの共通要件があります。高可用性データ・サービスのクライアントは、データ・サービスがどのマシン上にあっても、データ・サービスのネットワーク・アドレスおよびホスト名が同じであり、同じ方法で要求を出すことができると予期します。

高可用性 Web サーバーにアクセスする Web ブラウザーを考慮してみましょう。要求は URL とともに発行されます。 URL には、ホスト名と、Web サーバー上のファイルへのパスの両方が含まれます。ブラウザーは、 Web サーバーのフェールオーバー後もホスト名とパスが同じであると予期します。ブラウザーが Web サーバーからファイルをダウンロードしているときに、サーバーがフェールオーバーされると、ブラウザーは要求を再発行する必要があります。

データ・サービスの可用性は、データ・サービスがユーザーから使用可能である時間の合計により測定されます。可用性の最も一般的な測定単位は、「アップ時間」のパーセンテージです。これはしばしば、複数の「9」によって示されます。

```
99.99% => サービスは、1 年で (最大) 52.6 分ダウンする。
99.999% => サービスは、1 年で (最大) 5.26 分ダウンする。
99.9999% => サービスは、1 年で (最大) 31.5 秒ダウンする。
```

HA クラスターを設計し、テストするときは、

1. クラスターの管理者がシステムに精通しており、フェールオーバーの発生時に起きることを知っていることを確かめます。
2. クラスターの各部分に正しく冗長度があり、障害が起きたときにすばやく置き換えられることを確かめます。
3. 制御環境でテスト・システムに障害を起こし、システムが毎回正確にフェールオーバーされることを確かめます。
4. それぞれのフェールオーバーの理由を記録します。これはそれほど頻繁に起きることではありませんが、クラスターを不安定にする問題に対処するために重要です。たとえば、クラスターの一部が 1 か月に 5 回フェールオーバーを起こした場合、その理由を付きとめてそれを修正してください。

5. フェールオーバーが起きたときに、そのことがクラスターのサポート・スタッフに知らされることを確かめます。
6. クラスターが過負荷にならないようにします。フェールオーバーの後で、残りのシステムが引き続き許容レベルで作業負荷を処理できることを確かめてください。
7. よく障害の起きる構成要素 (ディスクなど) を検査し、問題が起きる前に置き換えます。

フォールト・トレランスおよび連続可用性

データ・サービスの可用性を上げる別の方法は、フォールト・トレランスです。*フォールト・トレラント・マシン*には、すべての冗長度が組み込まれており、 CPU やメモリーを含む任意の部分で起きる単一の障害に対処することができます。フォールト・トレラント・マシンは、隙間産業で最も良く使用され、通常、このマシンを実装するには費用がかかります。地理的に別の場所にあるマシンを含む HA クラスターには、これらの場所のサブセットだけに影響を与える災害から、回復することができるという利点があります。

連続可用性 は、高可用性の上位ステップです。これにより、計画済みおよび非計画のダウン時間からクライアントが保護されます。連続可用性が構成されていると、データ・サービスをホストするマシンの 1 つで障害が起きたり、または保守のためにダウンしたりした場合に、クライアントはまったく影響を受けません。連続可用性の構成は複雑で、実装にさらに費用がかかります。

HA クラスターは可用性を上げる最も一般的な方法です。なぜなら、拡張が容易で、使いやすく、そして比較的実装に費用がかからないからです。

Sun Cluster 2.2

Sun Cluster 2.2 (SC2.2) は、 Sun Microsystems の高可用性クラスター化製品です。 SC2.2 は単一のクラスターで最大 4 台のマシンをサポートします。 4 台の Ultra Enterprise 10000 を使用すると、クラスターは最大で 256 個の CPU と 256 GB の RAM を持つことができます。

サポートされるシステム

システム	UltraSPARC	メモリー容量	入出力
Ultra Enterprise 1	1	64MB ～ 1GB	3 SBus
Ultra Enterprise 2	1 ～ 2	64MB ～ 2GB	4 SBus
Ultra Enterprise 450	1 ～ 4	32MB ～ 4GB	10 PCI

Ultra Enterprise 3000	1 ～ 6	64MB ～ 6GB	9 SBus
Ultra Enterprise 4000	1 ～ 14	64MB ～ 14GB	21 SBus
Ultra Enterprise 5000	1 ～ 14	64MB ～ 14GB	21 SBus
Ultra Enterprise 6000	1 ～ 30	64MB ～ 30GB	45 SBus
Ultra Enterprise 10000	1 ～ 64	512MB ～ 64GB	64 SBus

エージェント

Sun Cluster ソフトウェアには、 SC2.2 製品に付属してサポートされる、多数の高可用性エージェントが含まれています。他の HA エージェント (たとえば、DB2 のエージェント) は、Sun の外部で開発されており、 Sun Cluster ソフトウェアには付属していません。 DB2 の HA エージェントは DB2 に付属しており、 IBM によってサポートされています。

Sun Cluster ソフトウェアは、 Sun Cluster ソフトウェアの各種構成要素に対応するメソッド (スクリプトまたはプログラム) を登録する機会を提供することにより、高可用性データ・サービスを処理します。これらのメソッドを使用して、 SC2.2 ソフトウェアでは、詳しい知識がなくてもデータ・サービスを制御できます。このメソッドには、以下のものがあります。

START
論理ネットワーク・インターフェースがオンラインになる前に、データ・サービスの一部を開始します。

START_NET
論理ネットワーク・インターフェースがオンラインになった後に、データ・サービスの一部を開始します。

STOP 論理ネットワーク・インターフェースがオフラインになった後に、データ・サービスの一部を停止します。

STOP_NET
論理ネットワーク・インターフェースがオフラインになる前に、データ・サービスの一部を停止します。

ABORT
STOP メソッドに似ていますが、クラスター・ソフトウェアによりマシンがダウン状態になる直前に実行されるという点が異なります。この場合、マシンの「健康」が重要であるため、データ・サービスでは、マシンがダウン状態になる前に「最後のお願い」要求を実行することができます。このメソッドは、論理ネットワーク・インターフェースがオフラインになった後に実行されます。

ABORT_NET
ABORT メソッドに似ていますが、論理ネットワーク・インターフェースがオフラインになる前に実行されるという点が異なります。

FM_INIT
障害モニターを初期化します。

FM_START
障害モニターを開始します。

FM_STOP
障害モニターを停止します。

FM_CHECK
hactl コマンドにより呼び出されます。対応するデータ・サービスの現行の状態を戻します。

DB2 エージェントは、スクリプト START_NET、STOP_NET、FM_START、および FM_STOP で構成されています。スクリプト ABORT、ABORT_NET、および FM_CHECK はクラスターの再構成時に実行されません。

高可用性エージェントは、これらのメソッドの 1 つまたは複数で構成されます。メソッドは、 **hareg** コマンドにより SC2.2 に登録されます。メソッドが登録されると、 Sun Cluster は対応するメソッドを呼び出して、データ・サービスを制御します。

サービスの ABORT および STOP メソッドは呼び出されない場合があることを覚えておくことは大切です。これらのメソッドはデータ・サービスの制御シャットダウンを目的としており、データ・サービスは、これらのメソッドを呼び出さずにマシンに障害が起きても回復できなければなりません。

詳細については、 Sun Cluster の資料を参照してください。

論理ホスト

SC2.2 ソフトウェアは、論理ホストの概念を使用します。*論理ホスト* は、一連のディスクと 1 つ以上の論理公衆ネットワーク・インターフェースで構成されます。高可用性データ・サービスは論理ホストと関連しており、論理ホストのディスク・グループにあるディスクを必要とします。論理ホストは、クラスター内のさまざまなマシンによってホスト可能で、実行されているマシンの CPU およびメモリーを「借りる」ことができます。

論理ネットワーク・インターフェース

他の UNIX ベースのオペレーティング・システムと同様、 Solaris にも、ネットワーク・インターフェースの 1 次 IP アドレスに加えて、追加 IP アドレスを持つ機能があります。追加 IP アドレスは、 1 次 IP アドレスが物理ネットワーク・インターフェースに常駐するのと同じ方法で、論理インターフェースに常駐します。以下は、クラスター内の 2 つのマシン上にある論理インターフェースの例です。 2 つのホストがあり、両方とも現在、マシン「thrash」上にあります。

```
scadmin@crackle(202)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 289966 0 289966 0 0 0
hme0 1500 9.21.55.0 9.21.55.98 121657 6098 764122 0 0 0
scid0 16321 204.152.65.0 204.152.65.1 489307 0 476479 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.33 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.17 347317 0 348073 0 0 0
```

1. lo0 は loopback インターフェースです。
2. hme0 は公衆ネットワーク・インターフェース (イーサネット) です。
3. scid0 は最初の私設ネットワーク・インターフェース (SCI (スケーラブル・コヒーレント・インターフェース)) です。
4. scid0:1 は Sun Cluster ソフトウェアが内部で使用する論理ネットワーク・インターフェースです。
5. scid1 は 2 番目の私設ネットワーク・インターフェースです。

```
scadmin@thrash(203)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 1128780 0 118780 0 0 0
hme0 1500 9.21.55.0 9.21.55.92 1741422 5692 757127 0 0 0
hme0:1 1500 9.21.55.0 9.21.55.109 0 0 0 0 0 0
hme0:2 1500 9.21.55.0 9.21.55.110 0 0 0 0 0 0
scid0 16321 204.152.65.0 204.152.65.2 476641 0 489476 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.34 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.18 348199 0 347444 0 0 0
```

1. hme0:1 は論理ホストの論理ネットワーク・インターフェースです。
2. hme0:2 は別の論理ホストの論理ネットワーク・インターフェースです。

論理ホストには、 1 つまたは複数の論理インターフェースが関連付けられています。これらの論理インターフェースは、論理ホストとともにマシン間を移動し、論理ホストと関連したデータ・サービスにアクセスするために使用されます。これらの論理インターフェースは論理ホストとともに移動するので、クライアントは、データ・サービスが常駐するマシンとは無関係に、そのデータ・サービスにアクセスすることができます。

高可用性データ・サービスは、 TCP/IP アドレス INADDR_ANY にバインドされます。これにより、システム上の各 IP アドレスは、データ・サービスのための接続を確実に受け取ることができます。代わりに、データ・サービスが特

定の IP アドレスにバインドされる場合、データ・サービスをホストしている論理ホストに関連した論理インターフェースをバインドしなければなりません。 INADDR_ANY にバインドしておけば、データ・サービスの必要とする新規 IP アドレスがシステムに到着した場合に、その IP アドレスに再バインドする必要がありません。

注: HA インスタンスのクライアントは、論理ホストの論理 IP アドレスのホスト名を使用して、データベースをカタログ化します。ここではマシンの 1 次ホスト名は使用すべきではありません。 DB2 がそのマシン上で実行されているという保証はないからです。

ディスク・グループとファイル・システム

データ・サービス用のディスクは、グループ単位 (またはセット単位) で論理ホストと関連付けられます。クラスターが Sun StorEdge Volume Manager (Veritas) を実行中の場合、 Sun Cluster ソフトウェアは、 Veritas の「vxdg」ユーティリティーを使用して、各論理ホストのディスク・グループをインポートおよびデポート (追放) します。以下に示すのは、2 つの論理ホスト「log0」および「log1」のディスク・グループの例です。これらの論理ホストは「thrash」というマシンによってホストされています。「crackle」というマシンは、現在どの論理ホストのホストにもなっていません。

```
scadmin@crackle(206)# vxdg list
NAME STATE ID
rootdg enabled 899825206.1025.crackle

scadmin@thrash(205)# vxdg list
NAME STATE ID
rootdg enabled 924176206.1025.thrash
data0 enabled 925142028.1157.crackle=
data1 enabled 899826248.1108.crackle
```

ディスク・グループ「data0」および「data1」はそれぞれ、論理ホスト「log0」および「log1」と対応します。ディスク・グループ「data0」は、以下のコマンドを実行することによって「thrash」からデポートすることができます。

```
vxdg deport data0
```

また、以下のコマンドを実行して「crackle」にインポートできます。

```
vxdg import data1
```

これは Sun Cluster ソフトウェアによって自動的に行われるもので、活動状態のクラスター上で手作業で行ってはなりません。

それぞれのディスク・グループには、クラスター内の 2 つ以上のマシン間で共有可能な多数のディスクが含まれています。論理ホストは、自身が属するディスク・グループ内のディスクに物理アクセス可能なマシンにのみ移動できます。

各論理ホストのファイル・システムを制御する 2 つのファイルがあります。

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logical_host>
/etc/opt/SUNWcluster/conf/hanfs/dfstab.<logical_host>
```

logical_host は、関連する論理ホスト名です。

vfstab ファイルは /etc/vfstab ファイルと類似していますが、ディスク・グループが論理ホスト用にインポートされた後でマウントされるファイル・システムの項目を含むという点が異なります。 dfstab ファイルは /etc/dfs/dfstab ファイルと類似していますが、 HA-NFS を介して論理ホスト用にエクスポートされるファイル・システムの項目を含むという点が異なります。各マシンにはこれらのファイルの独自コピーがあるため、クラスター内の各マシンで同じ内容になるように注意を払う必要があります。

注: 論理ホストの vfstab ファイルおよび dfstab ファイルのパスは、 hanfs ディレクトリーが含まれているため、間違いやすくなっています。論理ホストの dfstab ファイルだけが HA-NFS で使用されます。 vfstab ファイルは、 HA-NFS が構成されていなくても使用されます。

以下に示すのは、相互引き受け構成の 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 (EEE) を実行しているクラスターの例です。

```
scadmin@thrash(217)# ls -l /etc/opt/SUNWcluster/conf/hanfs
total 8
-rw-r--r-- 1 root build 173 Apr 14 15:01 dfstab.log0
-rw-r--r-- 1 root build 316 Apr 26 12:07 vfstab.log0
-rw-r--r-- 1 root build 389 Apr 13 21:04 vfstab.log1

scadmin@thrash(218)# cat dfstab.log0
share -F nfs -o root=crackle:thrash:¥
jolt:bump:crackle.torolab.ibm.com:thrash.torolab.ibm.com:¥
jolt.torolab.ibm.com:bump.torolab.ibm.com /log0/home
```

ファイル・システム /log0/home をマウントする許可が与えられているホストは、クラスター内の各マシン上のすべてのネットワーク・インターフェース (論理および物理) から来ています。ファイル・システムはルート許可とともにエクスポートされます。

```
scadmin@thrash(220)# cat vfstab.log0
#device to mount            device  to fsck              mount       FS   fsck mount   options
#                                                       point       type pass at boot
/dev/vx/dsk/data0/data1-stat /dev/vx/rdsk/data0/data1-stat /log0       ufs  2    no      -
/dev/vx/dsk/data0/vol01      /dev/vx/rdsk/data0/vol01      /log0/home  ufs  2    no      -
```

```
/dev/vx/dsk/data0/vol02      /dev/vx/rdsk/data0/vol02      /log0/data  ufs  2    no      -

scadmin@thrash(221)# cat vfstab.log1
#device to mount             device  to fsck               mount       FS   fsck mount   options
#                                                          point       type pass at boot

/dev/vx/dsk/data1/data1-stat /dev/vx/rdsk/data1/data1-stat /log1       ufs  2    no      -
/dev/vx/dsk/data1/vol01      /dev/vx/rdsk/data1/vol01      /log1/home  ufs  2    no      -
/dev/vx/dsk/data1/vol02      /dev/vx/rdsk/data1/vol02      /log1/data  ufs  2    no      -
/dev/vx/dsk/data1/vol03      /dev/vx/rdsk/data1/vol03      /log1/data1 ufs  2    no      -
```

vfstab.log0 ファイルには、 /log0 ディレクトリーの下にあるファイル・システム用に、 3 つの有効な項目が含まれます。論理ホスト log0 のファイル・システムが論理ボリューム装置を使用していることに注意してください。これは、論理ホストに関連したディスク・グループ data0 の一部です。

vfstab ファイル内のファイル・システムは上から下の順にマウントされるので、ファイル・システムが正しい順序でリストされているか確認することは重要です。特定のファイル・システムの下にマウントされるファイル・システムは、上位のファイル・システムより下にリストしなければなりません。論理ホストで必要な実際のファイル・システムは、データ・サービスの必要に依存しており、これらの例とは大きく異なります。

フェールオーバー時に、SC2.2 ソフトウェアは、論理ホストに関連するディスク・グループおよび論理インターフェースが、クラスター内のマシンの間で論理ホストとともに移動することを保証します。高可用性データ・サービスは、フェールオーバーの後に、少なくともこれらのリソースが新規システム上で使用可能にされることを予期します。実際には、多くのデータ・サービスはそれらが高可用性であることに気付いてさえおらず、これらのリソースをフェールオーバー後もまったく同じように「認識する」でしょう。

制御メソッド

制御メソッドは以下を使用して登録されます。

```
hareg(1m)
```

HA サービスが登録されると、 SC 2.2 はクラスターの再構成またはフェールオーバー中の適切な時刻に、 HA サービスに登録されたメソッドを呼び出します。

クラスターの再構成 (制御フェールオーバー) 時には、以下の処理が (指定された順序で) 行われます。マシンがクラッシュした場合、ステップ 5c より前のアクションは行われません。 (クラスターの再構成についての詳細は、 SC2.2 の資料を参照してください。)

1. FM_STOP メソッドが実行されます。
2. STOP_NET メソッドが実行されます。
3. 論理ホストの論理インターフェースがオフラインになります。

```
      - ifconfig hme0:1 0.0.0.0 down
  4. STOP メソッドが実行されます。
  5. ディスク・グループとファイル・システムが移動されます。
       a. 論理ホストのファイル・システムをアンマウントします。
       b. vxdg は 1 つのマシン上にディスク・グループをデポートします。

     - - 以下のステップはマシンがクラッシュした場合にのみ実行されます - -

       c. vxdg は 別のマシン上にディスク・グループをインポートします。
       d. 論理ホスト・ファイル・システムに対して fsck を実行します。
       e. 論理ホスト・ファイル・システムをマウントします。
  6. START メソッドが実行されます。
  7. 論理ホストの論理インターフェースがオンラインになります。
       - ifconfig hme0:1 <ip address> up
  8. START_NET メソッドが実行されます。
  9. FM_INIT メソッドが実行されます。
 10. FM_START メソッドが実行されます。
```

制御メソッドは、以下のコマンド行引き数を使用して実行されます。

```
METHOD <logical hosts being hosted> <logical hosts not being hosted> <time-out>
```

最初の引き数は、現在ホストされている論理ホストのコンマ区切りリストで、2 番目は、現在ホストされていない論理ホストのコンマ区切りリストです。最後の引き数は、メソッドのタイムアウト、つまり、SC2.2 ソフトウェアが打ちきるまでにメソッドを実行できる時間の合計です。

ディスクとファイル・システムの構成

SC2.2 は 2 つのボリューム・マネージャー Sun StorEdge Volume Manager (Veritas) および Solstice Disk Suite をサポートしています。両方とも正常に機能しますが、 StorEdge Volume Manager は、クラスター環境でいくらか有利です。クラスター構成によっては、ディスク格納装置のコントローラー番号が、クラスター内のマシンごとに異なる場合があります。コントローラー番号が違うと、コントローラーのディスク装置へのパスも異なります。 Disk Suite はディスク装置へのパスを直接使用するので、この状況では十分機能しません。StorEdge Volume Manager は、コントローラー番号に関係なく、ディスクそのものを使用するので、コントローラー番号が違っても影響を受けません。

HA の目標はデータ・サービスの可用性を上げることなので、すべてのファイル・システムおよびディスク装置がミラーリングされているか、または RAID 構成になっていることを確認することが重要です。これにより、ディスクの障害のために起きるフェールオーバーは防止され、クラスターの安定度が増します。

HA-NFS

DB2 UDB EEE では、 1 つのインスタンスが複数のマシンにわたって構成されるときに、ファイル共用システムが必要です。一般的な DB2 UDB EEE 構成では、ホーム・ディレクトリーが NFS を介して 1 つのマシンからエクスポートされ、 EEE インスタンスに関与しているすべてのマシン上でマウントされています。相互引き受け構成では、 DB2 UDB EEE は HA-NFS を使用して、高可用性ファイル共用システムを提供します。論理ホストの 1 つが HA-NFS を介してファイル・システムをエクスポートしたら、クラスター内の各マシンは、そのファイル・システムを、 EEE インスタンスのホーム・ディレクトリーとしてマウントします。 HA-NFS についての詳細は、 Sun Cluster の資料を参照してください。

cconsole および ctelnet ユーティリティー

SC2.2 には、2 つの役立つユーティリティー `cconsole` および `ctelnet` が付属しています。これらのユーティリティーを使用すれば、単一のコマンドをクラスター内の複数のマシンに同時に発行することができます。これらのユーティリティーを使用して構成ファイルを編集すれば、確実にクラスター内の全マシンで構成ファイルを等しい状態に保つことができます。また、これらのユーティリティーを使用すれば、各マシンでまったく同じ方法によりソフトウェアをインストールすることもできます。これらのユーティリティーについての詳細は、 Sun Cluster の資料を参照してください。

キャンパス・クラスタリングとコンチネンタル・クラスタリング

クラスター内のマシンが同じ建物にない場合、そのクラスターはキャンパス・クラスター と呼ばれます。キャンパス・クラスターは、単一の障害ポイントとして建物自体を除去する場合に役に立ちます。たとえば、クラスター内のマシンがすべて同じ建物にあり、それが焼け落ちた場合、クラスター全体に影響が及びます。しかし、マシンが複数の建物にあれば、建物の 1 つが燃えても、クラスターは存続します。

コンチネンタル・クラスター は、マシンが別々の都市に分散しているクラスターです。このクラスターの目標は、単一の障害ポイントとして地域を除去することです。このタイプのクラスターは、地震や津波のような災害からの保護を提供します。

現在、Sun Cluster は、10 km (約 6 マイル) の範囲内にあるマシンをサポートできます。このため、 2 つの異なるサイトの間での高速接続を必要とする人にとって、キャンパス・クラスタリングは実際的なオプションとなります。クラ

スターには、2 つの私用内部接続と、共用ディスクのための多数の光ファイバー・ケーブルが必要です。 2 つのサイトの間の高速接続にかかるコストは、その利点によって相殺されます。

一般的な問題

SC2.2 ソフトウェアは Cluster Configuration Database (CCD(4)) を使用して、クラスター構成用にクラスター全体の単一リポジトリーを提供します。 CCD には専用 API があり、 `/etc/opt/SUNWcluster/conf` ディレクトリーの下に保管されます。まれに、CCD の同期がとれなくなり、修正が必要となる場合があります。この状況で CCD を修正する最善の方法は、バックアップ・コピーから復元することです。

CCD をバックアップするには、クラスター内の全マシンでクラスター・ソフトウェアをシャットダウンし、 `/etc/opt/SUNWcluster/conf` ディレクトリーを tar 圧縮して、 tar ファイルを安全な場所に保管します。バックアップを作成したときにクラスター・ソフトウェアがシャットダウンされていないと、 CCD を復元するときに問題が起きる可能性があります。クラスター構成が変更されたときに新規バックアップをとることにより、バックアップ・コピーが最新の状態に保たれるようにしてください。 CCD を復元するには、クラスター内の全マシン上でクラスター・ソフトウェアをシャットダウンし、 `conf` ディレクトリーを `conf.old` に移動して、バックアップ・コピーを tar 解凍します。クラスターは、新規の CCD を使用して開始できます。

DB2 に関する考慮事項

このセクションでは、以下のトピックについて説明します。

HA インスタンスに接続するアプリケーション

高可用性 DB2 インスタンスに依存するアプリケーションは、フェールオーバー時に再接続が可能でなければなりません。論理ホストのホスト名および IP アドレスは同じままなので、別のホスト名に接続したり、データベースを再カタログする必要はありません。

2 つのマシンと 1 つの DB2 ユニバーサル・データベース エンタープライズ・エディションのインスタンスを持つクラスターを考慮してみましょう。EE インスタンスは通常、クラスター内のマシンの 1 つに常駐します。 HA インスタンスのクライアントは、 HA インスタンスに関連した論理ホストの論理 IP アドレス (またはホスト名) に接続します。

HA クライアントによれば、 2 つのタイプのフェールオーバーがあります。 1 つのタイプは、 HA インスタンスをホストしているマシンがクラッシュしたときに起こります。別のタイプは、 HA インスタンスを正常にシャットダウンする機会が与えられたときに起こります。

マシンがクラッシュして HA インスタンスをダウンした場合、データベースへの既存の接続および新規の接続は両方ともハングします。接続がハングするのは、ネットワーク上に、クライアントがデータベースに使用していた IP アドレスを持つマシンがないためです。データベースが正常にシャットダウンされると、 `db2stop force` はデータベースへの既存の接続を中断し、エラー・メッセージが戻されます。

フェールオーバー時に、データベースに関連した論理 IP アドレスはオフラインになります。これは、SC2.2 ソフトウェアによってオフラインにされたためか、または論理ホストをホストしていたマシンがクラッシュしたためです。このとき、データベースへの新規の接続は、少しの間ハングします。

データベースに関連した論理 IP アドレスは、最終的に、DB2 が開始される前に別のマシンに移動されます。この段階では、データベースへの接続はハングしませんが、通信エラーを受け取ります。これは、DB2 がまだシステム上で開始されていないためです。データベースに接続されていた DB2 クライアントも、通信エラーを受け取り始めます。クライアントはまだ接続状態にあると考えますが、論理 IP アドレスをホストし始めたマシンは、既存の接続を識別しません。接続は単純にリセットされ、 DB2 クライアントは通信エラーを受け取ります。少しした後、DB2 はマシン上で開始され、 DB2 への接続が正常に確立されます。この時点で、データベースが不整合になり、クライアントはそれが回復するのを待たなければならない場合があります。

HA 環境用のアプリケーションを設計する際、データベース接続がハングする場合のために特別なコードを書く必要はありません。接続は、 Sun Cluster ソフトウェアが論理 IP アドレスを移動させる少しの間だけハングします。 Sun Cluster 上で実行されるどのデータ・サービスにも、この段階で同様のハング接続があります。どのようにデータベースがダウンするかに関係なく、クライアントはエラー・メッセージを受け取り、正常に行われるまで再接続を試行しなければなりません。クライアントから見ると、 HA インスタンスがダウンしてから同じマシンに戻されたように見えます。制御されるフェールオーバーでは、クライアントには、強制的に切断されて、後に同じマシンでデータベースへの再接続が可能であるように見えます。制御されないフェールオーバーでは、クライアントには、データベース・サーバーがクラッシュしてから、すぐに同じマシンに戻されたように見えます。

EE および EEE インスタンスのディスクのレイアウト

DB2 は、必要とするディスク装置またはファイル・システムが、クラスター内の各マシン上で同じように見えることを予期します。これを確実にするためには、必要なディスクまたはファイル・システムを、 HA インスタンスに関連した論理ホストに従い、クラスター内の各マシンで同じパス名になるように構成する必要があります。

DMS および SMS 表スペースの両方とも、 HA 環境でサポートされています。 DMS 表スペースの装置コンテナーでは、ボリューム・マネージャーにより作成された生装置を使用する必要があります。これらの装置は、ミラーリングされているか、または RAID 構成になっています。 `/dev/rdsk/c20t0d0s0` のような正規のディスク装置は、以下の理由から、使用すべきではありません。

- 装置に、同時に複数のマシンから書き込まれる可能性が高くなる。
- コントローラー番号が別のマシンで異なる可能性がある。

DB2 がこの状況でフェールオーバーされると、必要なディスク装置は別のマシンの場合と同じように見えず、DB2 は始動しません。 DMS 表スペースのファイル・コンテナー、および SMS 表スペースのファイル・コンテナーは、マウントされたファイル・システムに常駐する必要があります。論理ホストのファイル・システムは、論理ホストの `vfstab` ファイルに組み込まれていると、自動的にマウントされます。

論理ホストの `vfstab` ファイルは以下のパスにあります。

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logical_host>
```

logical_host は、 `vfstab` ファイルに関連した論理ホストの名前です。

各論理ホストには独自の vfstab ファイルがあり、このファイルには、論理ホストのディスク・グループが現行のマシンに移された後で、 HA サービスが開始される前にマウントされるファイル・システムが含まれます。 Sun Cluster ソフトウェアは、ファイル・システムの正常性を保証するために、 **fsck** (ファイル・システム検査) を実行した後で適切に定義されたファイル・システムをマウントしようと試みます。 **fsck** が失敗すると、ファイル・システムはマウントされず、エラー・メッセージがログに記録されます。

注: プロセスにオープン・ファイルが含まれる場合、または現行の作業ディレクトリーがマウント・ポイントの下にある場合、マウントは失敗します。これを防ぐには、論理ホストの vfstab ファイルに含まれるマウント・ポイントの下に、プロセスが残されていないことを確認してください。

SMS 表スペースを使用するときは、任意の規則を EEE インスタンスのファイル・システムのレイアウトで使用することができます。以下に示すのは、hadb2_setup ユーティリティーで使用される規則です。

```
scadmin@crackle(190)# pwd
/export/ha_home/db2eee/db2eee
scadmin@crackle(191)# ls -l
total 18
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0000 -> /log0/disks/db2eee/NODE0000
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0001 -> /log0/disks/db2eee/NODE0001
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0002 -> /log0/disks/db2eee/NODE0002
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0003 -> /log0/disks/db2eee/NODE0003
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0004 -> /log0/disks/db2eee/NODE0004
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0005 -> /log1/disks/db2eee/NODE0005
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0006 -> /log1/disks/db2eee/NODE0006
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0007 -> /log1/disks/db2eee/NODE0007
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0008 -> /log1/disks/db2eee/NODE0008
scadmin@crackle(192)#
```

インスタンス所有者は db2eee で、 db2eee インスタンスのデフォルト・データベース・ディレクトリーは /export/ha_home/db2eee です。論理ホスト log0 は、データベース区画 0、1、2、および 3 をホストしており、論理ホスト log1 は、データベース区画 4、5、6、7、および 8 をホストしています。

各データベース区画ごとに、対応する NODE*xxxx* ディレクトリーがあります。データベース区画のノード・ディレクトリーは、関連する論理ホストのファイル・システムの下にあるディレクトリーを指しています。

パスの表記規則を選択するときは、以下のことを確かめてください。

1. ファイル・システムのディスクが、それを必要とするデータベース区画を担当する論理ホストのディスク・グループにある。
2. コンテナーを保持するファイル・システムが、論理ホストの vfstab ファイルによってマウントされている。

EE および EEE インスタンスのホーム・ディレクトリーのレイアウト

EE インスタンスの場合、ホーム・ディレクトリーは、論理ホストの vfstab ファイルで定義されたファイル・システムである必要があります。このディレクトリーは DB2 が開始される前に使用可能になり、クラスター内で論理ホストが移動される場所に DB2 とともに移されます。各マシンには vfstab ファイルの独自コピーがあるため、各マシンで同じ内容になるように注意を払う必要があります。以下に示すのは EE インスタンスのホーム・ディレクトリーの例です。

```
/log0/home/db2ee
```

/log0 は論理ホスト log0 の論理ホスト・ファイル・システムで、 db2ee は DB2 インスタンスの名前です。このホーム・ディレクトリーのパスは、「db2ee」インスタンスをホストするクラスター内の各マシンにある /etc/passwd ファイルに含める必要があります。

EEE インスタンスの場合、ホーム・ディレクトリーをセットアップする方法は 2 つあります。ホット・スタンドバイ構成の場合、ホーム・ディレクトリーは EE インスタンスと同じ方法でセットアップすることができます。相互引き受け構成の場合は、 HA-NFS をホーム・ディレクトリーで使用し、 EEE インスタンスをセットアップする*前に* 適切に構成しなければなりません。

クラスター内のマシンのうちの 1 つは、選択した論理ホストの dfstab ファイルを使用して、 EEE インスタンスのファイル・システムをエクスポートする必要があります。 dfstab ファイルには、マシンが論理ホストをホストしているときに NFS を介してエクスポートされるファイル・システムが含まれています。各マシンには dfstab ファイルの独自コピーがあるため、各マシンで同じ内容になるように注意を払う必要があります。

HA-NFS ファイル・システムに関する情報は、 (hadb2_setup プログラムによって) hadb2tab ファイルに置かれています。 HA エージェントがインスタンスに関する情報を読み取ると、インスタンスの HA-NFS ファイル・システムが自動的にマウントされます (342ページの『hadb2tab ファイル』を参照してください)。

HA-NFS ファイル・システムのマウント・ポイントは、通常、/export/ha_home です。クラスター内の各マシンで、このマウント・ポイントは HA-NFS ディレクトリーをエクスポートする論理ホストから NFS マウントされます。 EEE インスタンス所有者のホーム・ディレクトリーは、以下のディレクトリーに置かれており、呼び出されます。

```
/export/ha_home/<instance>
```

instance は、インスタンス所有者の名前です。

ホーム・ディレクトリーをマウントしたりアンマウントしないですむように、各マシン上のインスタンスごとにホーム・ディレクトリーを持つこともできます。これを行うには、確実にホーム・ディレクトリーが各マシン上で同一になるように、追加の管理オーバーヘッドが必要になります。これが失敗すると、DB2 は正常に始動しないか、また別の構成で DB2 が始動されることになります。これは、サポートされている構成では*ありません*。

論理ホストと DB2 UDB EEE

論理ホストは通常、1 つまたは複数のデータベース区画をホストしたり、HA-NFS ファイル・システムをエクスポートしたりするために選択されます。たとえば、クラスター内に 4 つのデータベース区画と 2 つのマシンがある場合、各マシンごとに 1 つの論理ホストが存在しなければなりません (339ページの図61)。一方の論理ホストで 2 つのデータベース区画のホストと HA-NFS ファイル・システムのエクスポートを行い、他方の論理ホストで残りの 2 つのデータベース区画のホストを行うことができます。

デフォルトでは、 DB2 UDB EEE インスタンスは、そのインスタンス用に活動状態にあるデータベース区画がすでに 1 つ以上存在しているマシンに、最大で 2 つのデータベース区画を正常に追加するのに十分のリソースを割り当てます。たとえば、クラスター上の単一インスタンス用に 4 つのデータベース区画がある場合は、論理ホストごとに 1 つのデータベース区画があるか、または 1 つの論理ホストが 3 つのデータベース区画をホストしている場合にのみ、このことが問題となります。どちらの場合も、同じインスタンスのデータベース区画をすでにホストしているマシンに対して、 3 つのデータベース区画をフェールオーバーすることが可能です。

DB2_NUM_FAILOVER_NODES レジストリー変数を使用すれば、フェールオーバーされるデータベース区画のために予約されたリソースの量を増やすことができます。

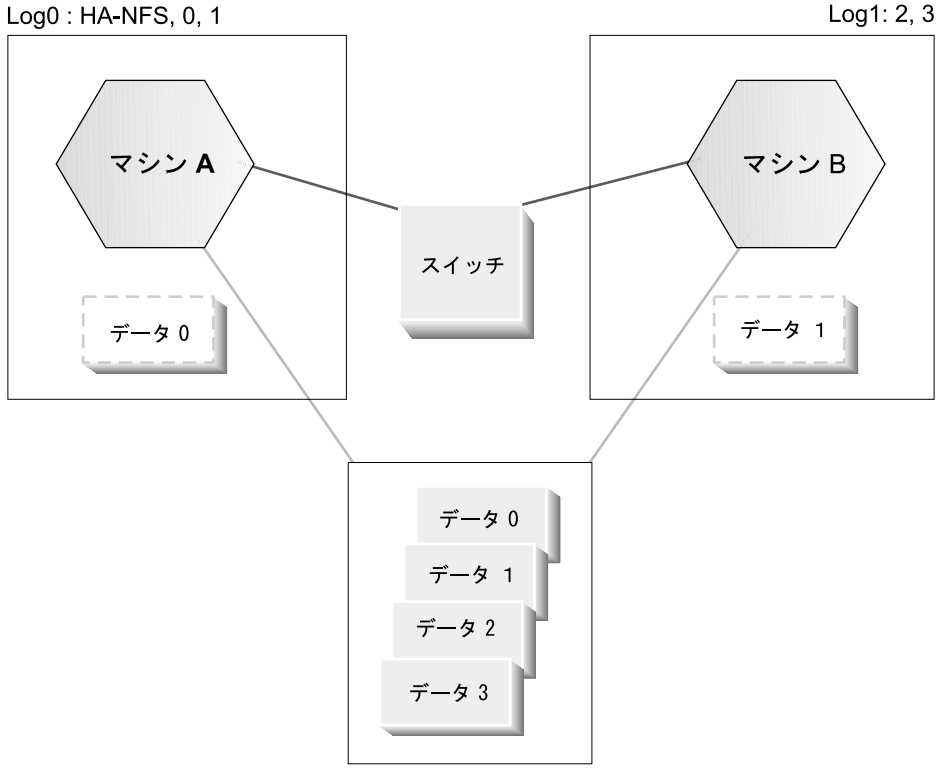


図 61. 各マシンごとに 1 つの論理ホストがある

DB2 のインストール場所およびオプション

DB2 がインストールされるファイル・システムは、ミラーリングされているか、または少なくとも RAID 構成にある必要があります。 DB2 が通常のディスクにインストールされていると、ディスク障害は起こりやすくなります。その結果起こるフェールオーバーは防止可能なものとみなされるため、クラスターの安定度は減少します。

DB2 は、論理ホストのディスク・グループにあるディスクにはインストールできません。これは、HA エージェントが常に、DB2 ライブラリーへのアクセスを必要とするからです。 DB2 ライブラリーにアクセスできないと、HA エージェントには障害が起こります。 DB2 は通常、クラスター内の各マシンでインストールしなければなりません。

データベースおよびデータベース・マネージャー構成パラメーター

データベース・マネージャー構成パラメーターは、フェールオーバーの後、DB2 が始動される前に、 pre_db2start スクリプトを使用して変更することができます (344ページの『ユーザー・スクリプト』を参照)。この実行可能スクリプトは、 (存在する場合に) インスタンス所有者のホーム・ディレクトリーの sqllib/ha ディレクトリーの下で実行されます。名前から分かるように、これは **db2start** の直前に実行されます。インスタンスが EEE インスタンスでない限り、 pre_db2start スクリプトには、制御メソッドに渡されたのと同じ引き数が渡されます。 EEE インスタンスの場合は、 pre_db2start スクリプトにも、 **db2start** コマンドのノード番号が渡されます。

破損回復

HA 環境での破損回復は、通常の環境での破損回復と同じです。 HA インスタンスが、クラッシュしたマシンとは別のマシンで再開されたとしても、インスタンスのファイルおよびディスク装置は同じように見え、データベースを回復するのに必要なアクションも異なりません。破損回復や他の形式のデータベース回復についての詳細は、 *管理の手引き: インプリメンテーション* の『データベースの回復』を参照してください。

データベースは手作業で (または、ユーザー・スクリプトの 1 つによって) 再始動できますが、特に EEE インスタンスの場合には、 *autorestart* データベース構成パラメーターを ON に設定するようお勧めします。これにより、データベースが不整合の状態になる時間を最小限にとどめることができます。

データ複製による高可用性

データの可用性は、複製によっても強化することができます。 2 つのサーバー間でデータを複写することにより、 1 つの形式の高可用性が実現します。サーバーの 1 つがダウンすると、別のサーバーがそれを引き継ぎ、データ・サービスを提供し続けることができます。

ただし、複製は非同期に行われるので、サーバーがダウンしたときに一部の変更内容が別のサーバーに伝えられない場合もあります。

DB2 高可用性エージェント

DB2 高可用性エージェントは、 DB2 と SC2.x の間の仲介者として機能します。このエージェントにより、DB2 についての詳しい知識がなくても、 Sun Cluster 2.2 ソフトウェアはクラスター環境で DB2 を制御することができます。 EE および EEE 両インスタンスのための 1 つのエージェントがあります。このエージェントは管理インスタンスとデータベース・インスタンスの両方をサポートします。

hadb2 サービスの登録

SC2.2 を使用して作業するには、 DB2 HA エージェントが登録されていなければなりません。データ・サービスを登録すると、使用可能な制御メソッドとそれらが常駐するディレクトリーが SC2.2 に知らされます。 HA エージェントに付属している特殊なスクリプト hadb2_reg は、 hadb2 サービスを EE および EEE インスタンスの両方に対して登録できます。 hadb2_reg スクリプトは、クラスター全体に対して 1 度だけ実行する必要があります。

DB2 HA エージェントの制御メソッドのセットは 1 つだけですが、それらが登録される方法は、 EEE インスタンスが相互引き受け構成で使用されるかどうかによって異なります。ホット・スタンドバイ構成の EE インスタンスまたは EEE インスタンスの場合、 HA-NFS は使用されません。したがって、 hadb2 サービスが HA-NFS に依存することを SC2.2 ソフトウェアに知らせる -d nfs スイッチは必要ありません。

DB2 V7.1 制御メソッドを EEE インスタンス用に登録するために hadb2_reg が使用する実際のコマンドは以下の通りです。

```
hareg -r hadb2 -b /opt/IBMdb2/V7.1/ha -m
START=hadb2_start,START_NET=hadb2_startnet,STOP_NET=hadb2_stopnet,
FM_START=hadb2_fmstart,FM_STOP=hadb2_fmstop
-t START_NET=$TIMEOUT,STOP_NET=$TIMEOUT -d nfs
```

-b スイッチは、すべての制御メソッドを opt/IBMdb2/V7.1/ha ディレクトリーで探すように SC2.x に指示します。 -m スイッチは、 hadb2 サービス用の実際の制御メソッドを定義します。 -t スイッチは、 START_NET および STOP_NET 制御メソッドのタイムアウトを定義します。各制御メソッドについての詳細は、 Sun Cluster の資料を参照してください。

hadb2_unreg スクリプトを使用すると、 hadb2 サービスを登録解除することができます。これは、hadb2_reg と同様、クラスターで 1 度だけ実行する必要があります。

hadb2tab ファイル

hadb2tab ファイルは、 DB2 HA エージェント用の主な構成ファイルです。各制御メソッドはこのファイルを調べて、可用性の高いインスタンスを見つけ出します。 hadb2tab ファイルは、 DB2 UDB バージョン 7.1 の /var/db2/v71/ ディレクトリーの下にあります。このファイルは複数インスタンスをサポートしており、それぞれの非コメント行は、それぞれの HA インスタンスを表します。以下は、hadb2tab ファイルの例です。

```
<scadmin@thrash(203)# cat hadb2tab
EEE DATA db2eee jolt ON /export/ha_home /log0/home #Added by DB2 HA software
EE ADMIN db2ee log1 ON  -               -          #Added by DB2 HA software
```

最初のフィールドは、インスタンスが EE インスタンスであるか、または EEE インスタンスであるかを DB2 HA エージェントに示します。 2 番目のフィールドは、インスタンスがデータ・インスタンスであるか、または管理インスタンスであるかを示します。 3 番目のフィールドには、 HA インスタンスのユーザー名が含まれています。 4 番目のフィールドは、インスタンスの論理ホストまたは HA-NFS ホストです。これは、インスタンスが EE インスタンスと EEE インスタンスのいずれであるかに依存します。 5 番目のフィールドは、インスタンスの障害モニターのオン / オフを示します。最後の 2 つのフィールドは、それぞれ、ローカル・マウント・ポイントとリモート HA-NFS ディレクトリーです。これらのフィールドは、使用されない場合は - (ハイフン) に設定され、 EEE 相互引き受け構成でのみ使用されます。行で「#」マーカーより前にある情報が、長さ 0 であるか、またはインスタンスの有効な定義であれば、 hadb2tab ファイルでコメントが使用できます。

制御メソッド

SC2.2 エージェントの制御メソッドは、一連のスクリプトまたはプログラムです。 DB2 (Solaris 版) のエージェントは、以下のメソッドを含む一連のプログラムです。

START_NET

hadb2_startnet が、DB2 を始動するのに使用されます。

STOP_NET

hadb2_stopnet が、DB2 を停止するのに使用されます。

FM_START

hadb2_fmstart が、 DB2 の障害モニターを始動するのに使用されます。

FM_STOP

hadb2_fmstop が、 DB2 障害モニターを停止するのに使用されます。

これらの制御メソッドについての詳細は、 Sun Cluster の資料を参照してください。

EE インスタンスの場合、インスタンスに関連する論理ホストは、 hadb2tab ファイルで定義されます。ただし、EEE インスタンスの場合、制御メソッドは以下のファイルも参照しなければなりません。

```
~<instance>/sqllib/ha/hadb2-eee.cfg
```

~<instance> は、インスタンス所有者のホーム・ディレクトリーです。このファイルには各データベース区画ごとに 1 つの行が含まれ、データベース区画と論理ホストを関連付けるために使用されます。以下に示すのは、有効な hadb2-eee.cfg ファイルの例です。

```
crackle % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

インスタンスまたはデータベース区画は、クラスター内で対応する論理ホストについていきます。論理ホストは、基礎となるハードウェアおよび SC2.2 によってサポートされるクラスター内のマシンならどれにでも移動できます。構成が適切にセットアップされていれば、 DB2 は SC2.2 ソフトウェアによってサポートされるどのトポロジーもサポートします。

インスタンスの全情報を読み取った後、制御メソッドは、インスタンスと関連する論理ホストを認識しています。コマンド行引き数を解析した後、制御メソッドは、現行のマシンによりホストされる論理ホストとホストされない論理ホストを認識しています。

以下の表は、実行されている制御メソッドと、データベース区画またはインスタンスと関連した論理ホストが現行のマシンでホストされているかどうかによって行われるアクションを示しています。

制御メソッド	関連した論理ホストがホストされている場合	関連した論理ホストがホストされていない場合
START_NET	DB2 インスタンスまたはデータベース区画を開始する	処置なし
STOP_NET	処置なし	DB2 インスタンスまたはデータベース区画を停止する
FM_START	インスタンスの障害モニターを開始する	処置なし
FM_STOP	処置なし	インスタンスの障害モニターを停止する

開始アクションを実行する制御メソッドは、現在ホストされている論理ホストだけを処理し、停止アクションを実行する制御メソッドは、現在ホストされていない論理ホストだけを処理します。

HA-NFS が使用されている場合、制御メソッドは、特別な方法で HA-NFS ディレクトリーをマウントする必要もあります。 HA-NFS のローカル・マウント・ポイントおよびディレクトリーが - (ハイフン) として定義されていない場合、制御メソッドはローカル・マウント・ポイントで statvfs(2) を実行します。ローカル・マウント・ポイントのファイル・システムのタイプが nfs でない場合、エージェントは、 hadb2tab 行の情報を使用してファイル・システムのマウントを試みます。 HA-NFS のマウント・ポイントおよびディレクトリーが - (ハイフン) として定義されていない場合、対応する論理ホストの vfstab ファイルが、インスタンスのホーム・ディレクトリーを含むファイル・システムをマウントするために必要となります。 HA-NFS のローカル・マウント・ポイントおよびリモート・ディレクトリーは、 EE および EEE のホット・スタンドバイ構成の場合、 - (ハイフン) としてのみ定義できます。

ユーザー・スクリプト

このスクリプトは制御メソッドから実行され、機能を追加します。このスクリプトには制御メソッドと同じコマンド行引き数が渡され、システム管理者またはデータベース管理者によって書き込まれます。

バックグラウンドで実行されないスクリプト内からプログラムを実行しなければならない場合、 nohup(1) を使用してそのプログラムをバックグラウンドで実行することを考慮してください。 nohup プログラムは、実行されているプログラムを、 SIGHUP (またはハングアップ) シグナルから保護します。 nohup

を使用しないと、バックグラウンドでスクリプトから実行されているプログラムは、スクリプトの終了時に SIGHUP シグナルの結果として強制終了される場合があります。

制御メソッドは以下のスクリプトを実行します。

- /var/db2/v61/failover
- ~<instance>/sqllib/ha/pre_db2start
- ~<instance>/sqllib/ha/post_db2start
- ~<instance>%s/sqllib/ha/post_failover
- ~<instance>/sqllib/ha/pre_db2stop
- ~<instance>/sqllib/ha/fm_warning

~*instance* は、 HA インスタンスのホーム・ディレクトリーです。

fm_warning スクリプトを除き、各ユーザー・スクリプトは、そのスクリプトを呼び出した制御メソッドと同じ引き数を使用して実行されます。 EEE インスタンスを使用している場合、データベース区画番号も (最後の引き数として) ユーザー・スクリプトに渡されます。

/var/db2/v71/failover スクリプトは、START_NET メソッドの最初に呼び出され、バックグラウンドで実行されます。このスクリプトを使用すると、たとえばフェールオーバーが起きたときにサポート・スタッフに電子メールを出すことができます。以下に、フェールオーバー・スクリプトの例を示します。

```
#!/bin/ksh
# E-mail or page support staff to notify them that a failover has occurred.
echo "Failover occurred on machine 'hostname':Running $0!" |/bin/mail admin@sphere.torolab.ibm.com
```

スクリプトから正常に電子メールを出すためには、 sendmail(1m) が適切にシステムで構成されていなければなりません。

名前から分かるように、 pre_db2start スクリプトは、 **db2start** が呼び出される直前に実行されます。このスクリプトは、データベース・マネージャー構成パラメーターの変更などのタスクで使用することができます。完了までに最大で 20 秒が与えられています。 EEE インスタンスの場合、このスクリプトは **db2start** が各データベース区画で呼び出される前に実行されます。このスクリプトは、データ・インスタンスにのみ適用でき、管理インスタンスには適用できません。

同様に、post_db2start スクリプトは、 **db2start** が呼び出された*直後に* 実行されます。このスクリプトは、データベースの再始動のようなタスクで使用す

ることができます。これは、バックグラウンドで実行されるので、その実行時間が他のインスタンスに支障をきたすことはありません。このスクリプトは、データ・インスタンスにのみ適用でき、管理インスタンスには適用できません。

インスタンス所有者のホーム・ディレクトリーの下にある post_failover スクリプトは、インスタンスの処理の後で実行されます。このスクリプトを使用すると、 DB2 が機能していることをクライアント・アプリケーションに知らせたり、データベースを活動化したり、状況ファイルを管理者に送信したりすることができます。これは、バックグラウンドで実行されるので、その実行時に別の HA インスタンスに対するアクションが遅れることはありません。以下に、フェールオーバー後処理スクリプトの例を示します。

```
#!/bin/ksh
#

# Send the status file to the administrato-r.
mail admin@sphere.torolab.ibm.com </tmp/HA.info.db2eee
```

DB2 HA エージェントの START_NET および STOP_NET メソッドは両方とも、各インスタンスの処理の後に状況ファイルを作成します。状況ファイルの名前は以下の通りです。

```
/tmp/HA.info.<instance>
```

instance は、インスタンス所有者のユーザー名です。状況ファイルには、インスタンスの開始および停止レポートと、制御メソッドを実行するのにかかった時間が含まれます。以下に、状況ファイルの例を示します。

```
scadmin@crackle(173)# cat /tmp/HA.info.db2eee
-----  Elapsed Time: 00:00:18           -----
-----  Elapsed Time: 00:00:00  (HA-NFS) -----

NODE     ACTION     RESULT     TRIES     RC
----     ------     ------     -----     --
   4     stop       success        3    1064
   5     stop       success        1    1064
   6     stop       success        2    1064
   7     stop       success        2    1064
   8     stop       success        1    1064
-----------------------------------------------
```

pre_db2stop スクリプトは、 **db2stop** が呼び出される直前に実行されます。このスクリプトを使用すると、 DB2 が停止しようとしていることをクライアント・アプリケーションに通知することができます。完了までに最大で 20 秒が与えられます。このスクリプトは、データ・インスタンスにのみ適用でき、管理インスタンスには適用できません。

障害モニターも、予期せぬシャットダウンのために DB2 が再始動されたときに、ユーザー・スクリプトを実行します。以下のスクリプトが呼び出されます。

```
~<instance>/sqllib/ha/fm_warning
```

fm_warning スクリプトを使用すると、 DB2 が障害モニターによって再始動されたことをシステム管理者に通知することができます。システム管理者は、DB2 が突然シャットダウンした理由を突き止め、これが再び起きないように適切な処置を取る必要があります。 fm_warning スクリプトはバックグラウンドで実行されます。

他の考慮事項

HA データ・サービスがオフにされると、フェールオーバーまたはクラスターの再構成時には停止メソッドだけが実行されます。他のメソッドは、HA データ・サービスが適切に登録され、オンにされた場合にのみ実行されます。

クラスター内の各マシンに、そのマシンが担当できるすべてのデータ・サービスを実行するのに十分なリソースがあることを確認してください。 CPU のロード、メモリー、スワップ、およびカーネル・パラメーターなどのリソースは、クラスターが実働する前に考慮しなければなりません。たとえば、クラスター内のマシンが 2 つの DB2 インスタンスを実行する必要がある場合、そのマシンのカーネル・パラメーターは、各インスタンスで必要とされるものの合計でなければなりません。

障害モニター

障害モニターがオンにされると、障害モニターはクラスターの再構成時またはフェールオーバー時に始動されます。 DB2 が START_NET スクリプトによって始動されていない場合、障害モニター自身が DB2 を始動します。障害モニターは、 DB2 が始動しなかったかどうか、または不明な理由でシャットダウンされたかどうかを検出できます。このため、障害モニターがオンになっているときは、DB2 を手作業でシャットダウンしないことが重要です。障害モニターはこれを予期しないシャットダウンと見なし、 DB2 を再始動します。これがあまりにも多く起きる場合、適切な論理ホストがフェールオーバーされます。

障害モニターがインスタンスで使用可能になっているとき、インスタンスを手作業で開始または停止する正しい方法は、最初に障害モニターまたは hadb2 サービスをオフにすることです。これらのアクションの両方とも、 -f および -s スイッチを使用した **hadb2_setup** コマンドによって開始できます (353ページの『hadb2_setup コマンド』を参照)。

注: 同じ論理ホストについて複数のインスタンスを使用しないでください。複数のインスタンスが 1 つの論理ホストと関連付けられていると、良好な状態にあるインスタンスが、不良の状態にあるインスタンスとともにフェールオーバーされる可能性があります。

EEE に関する考慮事項

論理ホストと関連させるデータベース区画を決定するときは、それらのデータベース区画がどのようにフェールオーバーするかを考慮することが重要です。2 つのマシン間にある 4 つのデータベース区画で使用される 2 マシン・クラスターを考慮してみましょう。これは、図62 に示されています。

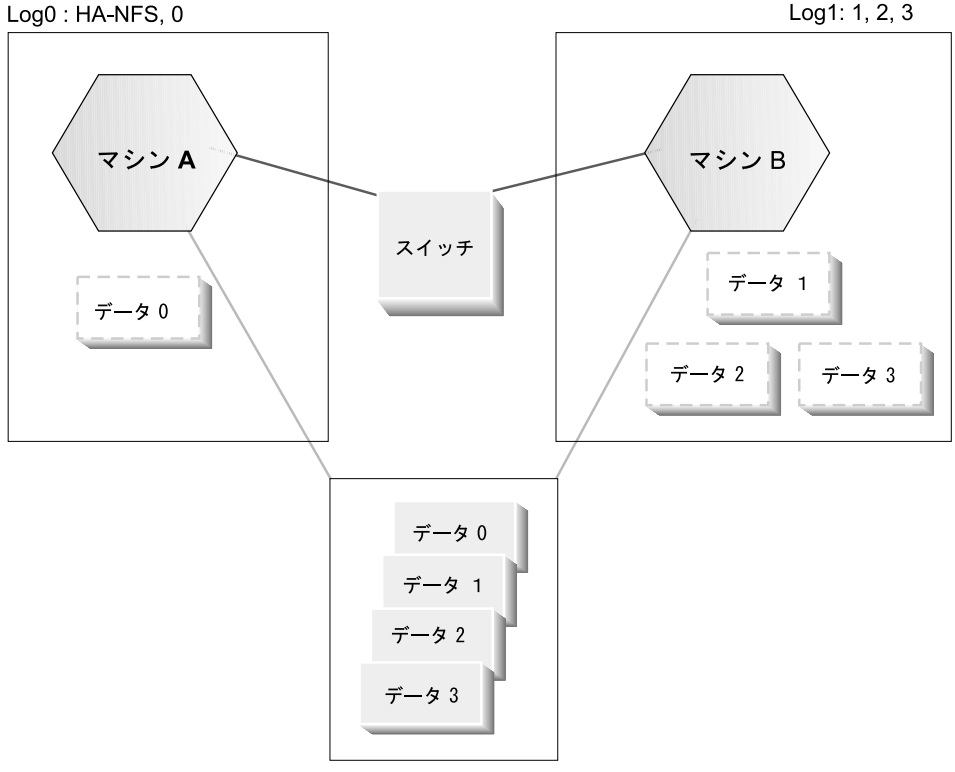


図 62. 4 つのデータベース区画を持つ 2 マシン・クラスター

一方の論理ホストを各データベース区画と関連付け、他方を HA-NFS に関連付けます。この場合、すべての論理ホストが 1 つのシステムによってホストされていると、問題が生じる可能性があります。システムに障害が起きる場合、すべての論理ホストを同時にシステムから移動しなければなりません。不運にも、 Sun Cluster は予測可能な順序で論理ホストを移動しないので、データベ

ース区画が関連付けられている論理ホストが HA-NFS の論理ホストの前に移動する可能性があります。単一のシステムでホストされるものごとにデータベース区画をグループ分けするのは良いアイデアです。このとき、通常 1 つのマシンでホストされる 2 つのデータベース区画が、単一の論理ホストと関連付けられています。

EEE インスタンスによって使用される db2nodes.cfg ファイルは、データベース区画が常駐するマシンを示すように更新されます。たとえば、すべてのデータベース区画が「crackle」というマシンにある場合、 db2nodes.cfg ファイルは以下のようになります。

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 crackle 4 204.152.65.33
5 crackle 5 204.152.65.33
6 crackle 6 204.152.65.33
7 crackle 7 204.152.65.33
8 crackle 8 204.152.65.33
```

これらのデータベース区画の一部が「thrash」というマシンに移動すると、db2nodes.cfg ファイルは以下のように更新されます。

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 thrash 0 204.152.65.34
5 thrash 1 204.152.65.34
6 thrash 2 204.152.65.34
7 thrash 3 204.152.65.34
8 thrash 4 204.152.65.34
```

マシン名「thrash」を反映するようにホスト名とスイッチ名の両方が変更されることと、ポート番号も異なることに注意してください。

HA.config ファイル

存在する場合、 /etc/HA.config ファイルには、いくつかの構成オプションが含まれます。これには、以下のものが含まれます。

```
scadmin@thrash(204)# cat /etc/HA.config
SYSLOG_FACILITY=LOG_LOCAL3
SYSLOG_LPRIORITY=LOG_INFO
SYSLOG_EPRIORITY=LOG_ERR
USE_INTERCONNECT=auto
SWITCH_NAME=204.152.65.18
DEBUG_LEVEL=2
```

```
FAILS_PER_HOUR=2
FAILS_PER_DAY=4
FAILS_PER_WEEK=10
FM_FAIL_SEV=soft
DB2START_TIMEOUT=60
DB2STOP_TIMEOUT=500
SCRIPT_USER=bin
```

注: HA.config ファイルが存在しない場合、デフォルト値が使用されます。

SYSLOG_FACILITY 変数は、メッセージおよびエラーの両方をログに記録するための SYSLOG 機能を設定します。 SYSLOG_LPRIORITY および SYSLOG_EPRIORITY 変数は、情報メッセージおよびエラー・メッセージのそれぞれをログに記録するための SYSLOG 優先順位を設定します。

SYSLOG デーモンが DB2 HA エージェントからの情報をログに記録できるようにするために、いくらかの変更が必要となる場合もあります。たとえば、/etc/syslog.conf ファイルに追加された以下の 2 行のうちの 1 つは、SYSLOG デーモンに、情報をログ・ファイルに書き込むよう指示します。

```
*.notice                                        /var/adm/SC.x
local3.info                                     /var/adm/SC.LOG_LOCAL3
```

Sun Cluster は通常、高速内部接続を行います。 DB2 との間で高速内部接続を使用するには、 USE_INTERCONNECT を auto または override に設定します。 auto 設定 (デフォルト) では、 Sun 内部論理ネットワーク・インターフェースを使用します。最初のインターフェースに障害が起きると、このインターフェースは別の物理インターフェースに移されます。 USE_INTERCONNECT が override に設定されている場合、スイッチ名は SWITCH_NAME 変数からとられます。あるいは、 USE_INTERCONNECT を no に設定することができます。これは高速内部接続を使用しないことを指定します。

DEBUG_LEVEL は、フェールオーバー時にログに記録される情報の量を指定します。これは 0 〜 10 の数字であり、10 が最高のデバッグ・レベルです。情報は、指定した SYSLOG 優先順位および機能でログに記録されます。問題が生じた場合は、デバッグ・レベルを最大レベルに設定し、 SYSLOG が HA エージェントからの出力をログを記録するように構成してから、 SYSLOG 出力を IBM サービスに送ってください。

変数 FAILS_PER_HOUR、FAILS_PER_DAY、および FAILS_PER_WEEK は、DB2 障害モニターが論理ホストをフェールオーバーするときを決定するのに役立ちます。各 HA 環境は異なっています。許容できる DB2 障害の数を決めなければなりません。「許容できる」障害が起きるたびに、 DB2 は同じマシン

上で再始動されます。これら 3 つの障害しきい値のうちの 1 つを超えると、インスタンスまたはデータベース区画に関連した論理ホストがフェールオーバーされます。

FM_FAIL_SEV 変数は、フェールオーバーが「ソフト」であるか「ハード」であるかを指定します。詳細については、 Sun Cluster の資料で `hactl(1m)` を参照してください。

DB2START_TIMEOUT および DB2STOP_TIMEOUT 変数は、 **db2start** および **db2stop** が実行可能な最大秒数を指定します。指定した間隔が過ぎると、HA エージェントは操作が失敗したとみなし、インスタンスの再始動を試みます。

特定のインスタンスと関連していないユーザー・スクリプトがあります。通常、これらのスクリプトはルートとして実行されますが、これは、SCRIPT_USER 変数によりオーバーライド可能です。 SCRIPT_USER 変数を設定すれば、これらのスクリプトを実行できるユーザー ID を指定できます。

制御メソッドが DB2 コマンドを実行する方法

DB2 HA エージェントは、 **su** コマンドを使用して、インスタンス所有者としてコマンドを実行します。実際のコマンドは以下のようなものです。

```
su - <instance> -c "db2stop"
```

instance は、インスタンスのユーザー名です。

インスタンス所有者の `.profile` ファイルが **su** に適したものであるようにすることは重要です。そうでないと、 **su** コマンドは正常に機能しない可能性があります。 **su** コマンドを手作業で、またはスクリプトから呼び出して、コマンドを正常に実行できることを確認してください。

セットアップ

このセクションを読む前に、必ず SC2.2 ソフトウェアに精通しておいてください。このセクションでは、読者が SC2.2 と HA-NFS のセットアップ方法を理解しており、ボリューム・マネージャーの使用方法を理解しているものと想定しています。 DB2 で必要な他のパッチに加えて、 HA エージェントでは以下のパッチが必要です。

```
Solaris 2.6:
   105210-17 (またはそれ以降)
   105786-05 (またはそれ以降)
```

注: Solaris 7 (Solaris 2.7) ではパッチは必要ありません。

一般的なインストール・ステップ

1. SC2.2 をクラスター内の全マシンにインストールする。インストール時に、SC2.2 はインストールするエージェントを尋ねてきます。 DB2 は SC2.2 に付属していないので、これはエージェントのリストにはありません。DB2 のエージェントは DB2 とともにインストールされ、 **hadb2_reg** コマンドによって登録されます。
2. 論理ホストをディスク・グループおよび論理 IP アドレスと共に構成する。

DB2 UDB エンタープライズ・エディションでのセットアップ

1. 論理ホストの論理ホスト・ファイル・システムの下に、インスタンスのホーム・ディレクトリーを作成する。
2. DB2 をクラスター内の全マシンにインストールする。
3. インスタンスのホーム・ディレクトリーが現在あるクラスター内のマシンに、インスタンスを作成する。
4. クラスター内の別のマシンにインスタンスのユーザーを追加し、数値ユーザー ID が同じであることを確認する。
5. **hadb2_reg** コマンドを使用して、 `hadb2` サービスを登録する。
6. **hadb2_setup** コマンドを実行して、インスタンスの HA をセットアップする。

DB2 UDB エンタープライズ拡張エディションでのセットアップ

1. HA インスタンス所有者のホーム・ディレクトリーを作成する。
   a. ホット・スタンドバイの場合、論理ホストの論理ホスト・ファイル・システムの下に、インスタンスのホーム・ディレクトリーを作成する。
   b. 相互引き受けの場合、 HA-NFS を構成し、論理ホストの 1 つからホーム・ディレクトリーをエクスポートする。マシンの 1 つで、選択したマウント・ポイントの下に HA-NFS ディレクトリーをマウントする。
2. DB2 をクラスター内の全マシンにインストールする。
3. HA-NFS ファイル・システムをマウントしたマシンに、インスタンスを作成する。
4. クラスター内の別のマシンにインスタンスのユーザーを追加し、数値ユーザー ID が同じであることを確認する。
5. **hadb2_reg** コマンドを使用して、 `hadb2` サービスを登録する。
6. **hadb2_setup** コマンドを実行して、インスタンスの HA をセットアップする。

注: NIS を使用して HA インスタンスの情報を定義することはお勧めしません。これは、NIS によって単一の障害ポイントが生じる可能性があるためです。

hadb2_setup コマンド

hadb2_setup コマンドは、 DB2 HA エージェントに付属するプログラムの中心をなすものです。これを使用すると、インスタンスのセットアップ、修正、または削除を行えます。また、 hadb2_setup サービスのオン / オフも行えます。このコマンドを使用すれば、 hadb2tab ファイルを手作業で編集する必要はありません。

注: **hadb2_setup** コマンドは、実行されるマシン上でのみ、アクションを実行します。 1 つのマシンに対して与えられる変更は、クラスター内の別のマシンにも行われます。

以下の引き数がサポートされています。

```
EE インスタンスを追加する場合:
---------------------
   hadb2_setup -a -i <instance> -f [on|off] -h <logical_host> -p [DATA|ADMIN] -t EE

   たとえば、以下のようになります。
      hadb2_setup -a -i db2ee -f off -h log1 -p DATA -t EE


EEE インスタンスを追加する場合:
----------------------
   hadb2_setup -a -i <instance> -f [on|off] -h <nfs_host> -l <mount_point> ¥
      -r <ha-nfs_dir> -p [DATA|ADMIN] -t EEE -n "<node_info>"

   たとえば、以下のようになります。
      hadb2_setup -a -i db2eee -f off -h ha-sun1 -l /export/ha_home ¥
         -r /log0/home -p DATA -t EEE -n "log0[0,10,20],log1[30,40,50]"


インスタンスを削除する場合:
---------------------
   hadb2_setup -d -i <instance>

インスタンスを修正する場合:
---------------------
   hadb2_setup -m -i <instance> [-f [on|off] | -l <mount_point> | ¥
      -h <host> | -p [DATA|ADMIN] | -r <ha-nfs_dir> | -t [EE|EEE] ]


他のオプション:
-------------
   -s   <on|off>            (すべての HA インスタンスで) hadb2 のオン / オフを行う
   -y                       安全チェックで「はい (yes)」を想定する
```

hadb2 サービスをオンまたはオフにするには、 -s スイッチを指定してください。これは、-n および -y スイッチを指定して **hareg** コマンドを実行し、

hadb2 サービスを指定することに相当します。 **hareg(1m)** コマンドについての詳細は、 Sun Cluster の資料を参照してください。

インスタンスの障害モニターは、 -f スイッチを使用してオフにすることができます。これは、ローカル・マシン上にあるインスタンスの障害モニターを停止させる効果と、 hadb2tab ファイルを修正して障害モニターが停止したことを反映させる効果があります。

EE インスタンスの場合、インスタンスがフェールオーバーされるときに全マシンの障害モニターをオフにすることをお勧めします。 EEE インスタンスの場合は、手作業でシャットダウンする前に、インスタンスのデータベース区画をホストしている全マシン上で障害モニターをオフにしなければなりません。

インスタンスを削除するには、 -d スイッチを使用します。これは、 hadb2tab ファイルからインスタンスを除去するだけです。他のファイルやディレクトリーを除去したり修正したりすることはありません。 hadb2tab ファイルは HA-DB2 エージェントの主な構成ファイルなので、このファイルからインスタンスを除去すると、制御メソッドはその存在を認識しなくなります。

インスタンスを修正するには、 -m スイッチを使用します。これは、 hadb2tab ファイルの情報を変更するだけです。他のファイルやディレクトリーを除去したり修正したりすることはありません。 -m スイッチは、 hadb2tab ファイルにある情報に関する任意のスイッチと共に使用することができます。 db2nodes.cfg ファイルおよび hadb2-eee.cfg ファイルは、最初のセットアップの後に手作業で変更しなければなりません。これは、**hadb2_setup** コマンドがこれらのファイルの修正をサポートしていないためです。

インスタンスの追加には、さらに多くのことが関係しています。

EE インスタンスの場合、以下の引き数が必要です。

```
hadb2_setup -a -i <instance> -f <fm> -h <logical_host> -t <EEE_or_EE> -p <purpose>
```

instance は追加されるインスタンスの名前です。 *fm* は障害モニターを最初にオンにするかオフにするかを指定します。 *logical_host* は関連する論理ホストです。 *EEE_or_EE* は EE に設定されます。 *purpose* は DATA または ADMIN のいずれかになります。

EEE インスタンスの場合、以下の引き数が必要です。

```
hadb2_setup -a -i <instance> -f <fm> -h <nfs_host> -t <EEE_or_EE> -p
   <purpose> -l <mount_point> -r <HA-NFS_directory> -n <node_info>
```

instance は追加されるインスタンスの名前です。 *fm* は障害モニターを最初にオンにするかオフにするかを指定します。 *nfs_host* は HA-NFS ファイル・システムをエクスポートする論理ホストのホスト名です。 *EEE_or_EE* は EEE に設定されます。 *purpose* は DATA または ADMIN のいずれかになります。 *mount_point* は HA-NFS ディレクトリーのローカル・マウント・ポイントです。 *HA-NFS_directory* は HA-NFS ディレクトリーです。 *node_info* はデータベース区画と論理ホストを関連させる情報です。たとえば、以下のようになります。

```
hadb2_setup -a -i db2eee -f on -h jolt -l /export/ha_home -p DATA -t EEE -r
   /log1/home -n "log0[0,1],log1[2,3]"
```

EEE インスタンスを追加するとき、ノード情報は引用符で閉じる必要があります。この例では、インスタンス「db2eee」は 2 つの論理ホスト「log0」および「log1」と関連付けられます。「db2eee」インスタンスのデータベース区画「0」および「1」は論理ホスト「log0」と関連付けられ、データベース区画「2」および「3」は論理ホスト「log1」と関連付けられます。

hadb2_setup コマンドを使用すると、インスタンスをクラスター内の全マシンに追加できます。インスタンスはその後、強制的にクラスターの再構成を行うか、または hadb2 サービスをオフにしてからオンにすることによって始動させることができます。これを行うには、**hareg** コマンドを実行するか、 -s スイッチを指定して **hadb2_setup** コマンドを実行します。インスタンスが始動しない場合は、 359ページの『トラブルシューティング』を参照してください。

hadb2_setup コマンドで EEE インスタンスを追加すると、以下のアクションが透過的に実行されます。

- 指定した情報の検査。これには、ユーザーがシステムに存在することと、HA-NFS が実行されていることを確認することが含まれます。
- db2nodes.cfg ファイルの作成。
- hadb2-eee.cfg ファイルの作成。
- EEE インスタンスの .rhosts ファイルの作成。
- デフォルトのデータベース・パスから関連する論理ホストのデータ・ディレクトリーへのシンボリック・パスの作成。
- hadb2tab ファイルへの行の追加。

構成エラーを防止し、 HA インスタンスを **hadb2_setup** コマンドの実行後に始動できるようにするために、コマンドは新規インスタンスが追加される前にかなりの量のテストを実行します。

db2nodes.cfg ファイルが作成され、現行のクラスターの状態に対応する情報が入れられます。たとえば、論理ホスト「log0」がマシン「crackle」によってホストされている場合、「log0」に関連したデータベース区画の項目には、マシン名「crackle」および「crackle」の高速内部接続が含まれます。

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 thrash 0 204.152.65.34
3 thrash 1 204.152.65.34
```

hadb2-eee.cfg ファイルは、コマンドで指定したノード情報のみに基づいて作成されます。データベース区画ごとに 1 つの行があります。

```
sphere % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

.rhost ファイルは DB2 UDB EEE で必要とされ、このファイルにはクラスター内の各マシンの全ホスト名 (または IP アドレス) が含まれていなければなりません。たとえば、以下の通りです。

```
crackle db2eee
204.152.65.1 db2eee
204.152.65.17 db2eee
thrash db2eee
204.152.65.2 db2eee
204.152.65.18 db2eee
crackle db2eee
jolt db2eee
bump db2eee
thrash.torolab.ibm.com db2eee
crackle.torolab.ibm.com db2eee
```

SMS 表スペースのファイル・システム・レイアウトに従い、 **hadb2_setup** コマンドは複数のディレクトリーやシンボリック・リンクをセットアップします。これには以下のものが含まれます。

- 各論理ホストの論理ホスト・ファイル・システムの下の「data」ディレクトリー。
- 論理ホストに関連した各データベース区画の (「data」ディレクトリーの下にある) ノード・ディレクトリー。
- デフォルトのデータベース・パスのシンボリック・リンクは、 ~<instance> の下にあります。 ~*instance* はインスタンスのホーム・ディレクトリーです。各データベース区画ごとに、対応するノード・ディレクトリーを指す 1

つのシンボリック・リンクがあります。詳しくは、335ページの『EE および EEE インスタンスのディスクのレイアウト』を参照してください。

フェールオーバー時間

フェールオーバー時間は、データが最初に使用不可になったときから再び使用可能になったときまでの時間により計測されます。フェールオーバー中に以下のような複数の事象が起きると、フェールオーバー時間に大きく影響する可能性があります。

- ディスクのデポートとインポート。

  ディスクのデポートとインポートには、通常、他の事象に比べて非常に長い時間がかかります。ただし、これは全体のダウン時間には影響しません。フェールオーバー時に 1 つのマシンから別のマシンに移動する必要のあるディスクが多いほど、プロセスにかかる時間は長くなります。欠陥のあるディスクがある場合、プロセスはさらに長くなる可能性があります。

- 論理ホスト用にマウントされているファイル・システムの Fsck。

  論理ホストのファイル・システムをマウントする前に、 `fsck` を実行してファイル・システムの正常性を確認しなければなりません。ファイル・システムが大きいほど、このプロセスには時間がかかります。ジャーナル・ファイル・システムを使用すると、この時間を大幅に減少させることができます。ジャーナル・ファイル・システムは通常、HA 環境で使用されるので、 `fsck` 時間は問題になりません。

- HA エージェントから呼び出されるユーザー・スクリプト。

  HA エージェントは、存在して実行可能なユーザー・スクリプトを呼び出します。これらのスクリプトのいくつかは同期的に実行され、 HA インスタンスを立ち上げるのにかかる時間を増す可能性があります。これらのスクリプトができるだけ素早く実行されるようにしてください。これらのスクリプトにより呼び出される外部プログラムをバックグラウンドで実行することを考慮できます。

- HA-NFS。

  単一の EEE インスタンスが相互引き受け構成にある場合、 HA-NFS をインスタンス所有者のホーム・ディレクトリーで使用しなければなりません。HA-NFS は、 *lockd* (HA-NFS の HA エージェントで定義される) の猶予期間のために、フェールオーバー時間を増します。 HA-NFS の実行時におけるこの時間は 90 秒です。フェールオーバー後に HA-NFS ファイル・システム上でファイルをロックするプロセスが、猶予期間の終了を待たなければならないため、これはフェールオーバー時間に影響を与えます。 DB2 の HA エージェントは、フェールオーバー後にインスタンス所有者のホーム・

ディレクトリーの下でファイルをロックする最初のプロセスであり、最初のロックを取得するのにかかる時間を記録します。この時間は、フェールオーバー後の状況レポートに表示されます。

- DB2 の開始。

  DB2 の開始は、少ししかフェールオーバー時間に影響しません。 EE インスタンスの場合、平均で約 5 ～ 15 秒影響します。 EEE インスタンスの場合、約 10 秒に加え、フェールオーバーされるデータベース区画ごとに約 5 秒影響します。たとえば、3 つのデータベース区画がフェールオーバーされている場合、これら 3 つのデータベース区画の開始によって影響するフェールオーバー時間は、約 25 秒です。これには、インスタンスのデータベースの破損回復は含まれません。

- データベースの破損回復。

  破損回復はしばしば、フェールオーバーに関連するダウン時間の大部分に影響します。データベースの回復にかかる時間は、以下のいくつかの要因に依存します。

  – クライアントの作業負荷。データベースの変更だけがトランザクション・ログに記録されます。クライアントの作業負荷のほとんどが読み取り専用の操作である場合、破損回復時にデータベースに適用しなければならないトランザクションは比較的少なくてすみます。

  – ディスクとマシンの速度。ディスクおよび HA インスタンスをホストしているマシンの速度も、データベースを回復するためにかかる時間に貢献します。システムが速いほど、破損回復時間は短くなります。

  – *softmax* データベース構成パラメーターの値。 *softmax* の値は、ログ・ファイル・サイズのうち、ソフト・チェックポイントが取られ、ログ制御ファイルが書き込まれる部分のパーセンテージです。ログ制御ファイルは、破損回復時に使用され、データベースを一貫した状態に復元するために実際に必要なログ・レコードを判別します。この値を小さくすると、データベース・マネージャーがページ・クリーナーを起動する頻度が上がり、ソフト・チェックポイントを作成する頻度も上がります。パフォーマンスは落ちますが、データベースの回復は速くなります。

  – インスタンスが EE と EEE のどちらであるか。インスタンスが EEE インスタンスである場合、データベースの再始動操作は、並列に実行されます。各データベース区画は、データベースの各部分の再始動を担当します。データベースに 50 GB のデータがある場合、 4 つのデータベース区画を持つインスタンスは、 1 つの EE インスタンスよりも約 4 倍速くデータベースを回復させることができます。

トラブルシューティング

以下の表には、起こり得る問題、その推定原因、および解決するために取ることのできるアクションが示されています。

表 29. Sun Cluster 2.2 での高可用性のトラブルシューティング

症状	考えられる原因	アクション
論理ホストのファイル・システムをマウントできない	論理ホストのファイル・システムは通常、論理ホストのフェールオーバー時にマウントおよびアンマウントされます。フェールオーバーのときに、論理ホストのファイル・システムの下に活動状態のプロセスまたはオープン・ファイルがあってはなりません。まれに、強制終了できないプロセスが、その現行作業ディレクトリーを論理ホストのファイル・システムの下に保持していることがあります。プロセスがマウント・ポイントの下にあるかどうかを知るには、 `fuser(1m)` か、 GNU ユーティリティー `lsof` を使用します。論理ホストのファイル・システムをマウントできない場合、エラー・メッセージが出されます。[a]	システムをリブートするか、または論理ホストのファイル・システムを別の名前にして再作成してください。これを行うことにより、凍結したプロセスをディレクトリーの下に残すことができ (強制終了ができないため)、マウントができるようになります。 [b]
db2start または db2stop のタイムアウトが機能しない	SIGALRM シグナルがブロック化システム呼び出しから出されない場合があります。代わりに、 SA_RESTART フラグが `sigaction()` を使用して設定されたかのように、このシステム呼び出しが再始動されます。これにより、DB2 HA エージェントのタイムアウトは無視され、エージェント・メソッドはハングし、ハングした **db2start** または **db2stop** コマンドから回復されません。	Solaris 2.6 用の必要なパッチ 105210-17 (またはそれ以降) を適用してください。

表 29. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
インスタンスにログインするとハングする	これが起きる理由はたくさんありますが、最も一般的な理由には、 NFS の問題および /usr/sbin/quota プログラムが関係します。	NFS マウントを検査してそれが良好な状態であることを確認してください。また、インスタンス所有者が所有する割り当て量のプロセスも確認してください。システム管理者の判断により、割り当て量プログラムを /bin/true へのシンボリック・リンクに変更すると、この問題を解決できる可能性があります。ただし、これは推奨されている解決方法ではありません。
EEE インスタンスをセットアップしたが、始動しない	**hadb2_setup** コマンドが、ポートを /etc/services ファイルに追加しません。管理者が手作業で追加すると予期されています。エラー・メッセージが戻されます。[c]	/etc/services ファイルで適切なポートを指定したことを確認してください。

表 29. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
START_NET メソッドが DB2 を開始できない		障害モニターをオフにして、インスタンスがフェールオーバーされていないことを確認してください。インスタンス所有者としてログインして、 DB2 を手作業で開始してください。 1. hadb2tab 構成ファイルで適切なインスタンス・タイプが指定されていることを確認します。たとえば、 EE 管理インスタンス用の db2nodes.cfg ファイルがあると、問題が起こります。 HA エージェント・メソッドは、この問題から回復することができません。 2. .rhosts ファイルが存在し、その中に有効な項目があることを確認します。 3. HA-NFS ファイル・システムが、クラスター内の全マシンのルート許可によって共用されていることを確認します。 4. カーネル・パラメーターを検査し、正しいことを確認します。 5. /etc/services ファイルにインスタンスの項目が含まれることを確認します。

表 29. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
インスタンスが 1 つのマシンでしか機能しない	• インスタンスの *uid* 数値が、クラスター内の各マシンで同じでない可能性があります。 • カーネル・パラメーターが、クラスター内の各マシンで有効でない可能性があります。 • `hadb2tab` ファイルが、クラスター内の各マシンで同じでない可能性があります。 • 論理ホストの `vfstab` ファイルのような別の構成ファイルが、クラスター内の各マシンで同じでない可能性があります。	これらの原因のどれも当てはまらないと思われる場合、インスタンス所有者としてログインを試みて、 DB2 を手作業で開始してください。 EE インスタンスの場合、インスタンスをホストしている論理ホストが現行のマシンによってホストされていれば、これはうまくいくはずです。 EEE インスタンスの場合、データベース区画をホストできるクラスター内のどのマシンからも、これはうまくいくはずです。
su - <instance> -c ″db2start″ が機能しない	• インスタンスの `.profile` が、**su** に適していない可能性があります。 • Bourne シェル (`/bin/sh`) に関する既知の問題があります。このシェルでは、**su** コマンドは手作業では機能しますが、 HA エージェントを介しては機能しません。	• ルートとして、このコマンドを手作業で実行して機能することを確かめてから、 HA エージェントを介して再試行してください。 • 必要であれば、 Korn シェル (`/bin/ksh`) に切り替えてください。
EEE インスタンスが開始しないのに、ホーム・ディレクトリーはマウントされている	HA-NFS ディレクトリーが「ルート」許可と一緒にクラスター内のマシンにエクスポートされなかった可能性があります。 DB2 と HA エージェントの両方で、適切に実行されるにはこれが必要です。	これをテストするには、インスタンス所有者のホーム・ディレクトリーの下に、 (ルートとして) ファイルを作成してください。

表 29. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
EEE インスタンスのディレクトリーにアクセスしようとすると、「不良な NFS ファイル・ハンドル」エラーが戻される	インスタンス所有者のホーム・ディレクトリーの下にプロセスがまだ残っている可能性があります。	インスタンス所有者のホーム・ディレクトリーをアンマウントし、 HA エージェントがそれを再マウントできるようにしてください。 HA エージェントが再マウントするのは、hadb2 サービスがオフになり、再びオンになった場合です (353ページの『hadb2_setup コマンド』の **hadb2_setup** コマンドにある -s スイッチの説明を参照してください)。

表 29. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
制御メソッドが SC2.2 を介して正常に実行されない	hadb2 サービスが Sun Cluster に登録されていないか、それがオンになっていない可能性があります。	制御メソッドがコマンド行から通常どおり実行されているように見える場合は、 SYSLOG ファイルを検査して、問題を説明するのに役立つエラー・メッセージを探してください。 hadb2 サービスが Sun Cluster ソフトウェアに登録されており、それがオンになっていることを確認してください。 手動によるメソッドの実行は、問題のデバッグに役立ちます。[d] 制御メソッドは、ルートとして実行し、適切なコマンド行引き数を渡す必要があります。論理ホストのリストがヌルの場合、引き数は "" として渡されます。ブランク・スペースの区切り文字がない二重引用符は、ブランクの引き数を表します。以下はその例です。 `hadb2_startnet log0,log1 "" 600` 最初の引き数 log0,log1 は、論理ホスト log0 および log1 が現行のマシンによってホストされていることを、 hadb2_startnet メソッドに知らせます。 2 つ目の引き数はヌルで、これはクラスター内のほかのマシンでホストされている論理ホストが存在しない (すべてが現行のマシンにある) ことを、 hadb2_startnet メソッドに知らせます。 3 つめの引き数は、 SC2.2 が 600 秒後にタイムアウトになることをメソッドに知らせます。

表 29. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション
ユーザー・スクリプトが実行されない	ユーザー・スクリプトは、それが適切なディレクトリーにあり、実行可能である場合にのみ実行できます。	所有権と属性を検査してください。それでもスクリプトが実行できない場合は、 IBM サービスに連絡してください。実行できないスクリプトのディレクトリー・リストと、スクリプトを実行できなければならなかったフェールオーバーまたはクラスター構成の SYSLOG 出力を転送してください。
/etc/syslog.conf で指定したファイルに情報を記録できない		`touch(1)` を使用して、`/etc/syslog.conf` ファイルで指定したファイルを作成してから、SYSLOG デーモンを再始動してください。

表 29. Sun Cluster 2.2 での高可用性のトラブルシューティング (続き)

症状	考えられる原因	アクション

[a] 論理ホストのファイル・システムをマウントできないときに出されるエラー・メッセージは、以下のようなものです。

```
Aug 17 11:14:01 rash ID[SUNWcluster.loghost.1170]: importing data1
Aug 17 11:14:06 rash ID[SUNWcluster.scnfs.3040]: mount -F ufs -o ""
     /dev/vx/dsk/data1/data1-stat /log1 failed.
Aug 17 11:14:07 rash ID[SUNWcluster.ccd.ccdd.5304]: error freeze cmd =
     /opt/SUNWcluster/bin/loghost_sync
CCDSYNC_POST_ADDU LOGHOST_CM:log1:rash /etc/opt/SUNWcluster/conf/ccd.database
     2 "0 1" 1 error code = 1
```

[b] たとえば、以下の通りです。

```
scadmin@rash(218)# ps -fe | egrep db2
db2ee 1984 1 0 0:01 <defunct>

Solution:

   scadmin@rash(229)# cd /
   scadmin@rash(230)# mv /log1 /log1.bkp
   scadmin@rash(231)# mkdir /log1
```

[c] エラー・メッセージは以下のようなものです。

```
SQL6030N START or STOP DATABASE MANAGER failed. Reason code "13".
```

[d] たとえば、 `hadb2_startnet` メソッドが `libdb2.so.1` を見付けられないが、 Sun Cluster を介して通常どおり実行される場合、エラーは報告されません。このメソッドを手作業で実行すると、以下のような結果が出されます。

```
scadmin@crackle(213)# hadb2_startnet '''log0,log1' 600
ld.so.1: hadb2_startnet: fatal: libdb2.so.1: open failed:
   No such file or directory
Killed
```

第5部 付録および後付け

付録A. 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、385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、 381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。

- 384ページの『オンライン情報の表示』
- 389ページの『オンライン情報の検索』
- 381ページの『印刷資料の注文方法』
- 381ページの『PDF 資料の印刷』

表 30. DB2 情報

資料名	説明	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。	第 1 巻 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、および回復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。	第 2 巻 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。	第 3 巻 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
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 30. DB2 情報 (続き)

資料名	説明	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 30. DB2 情報 (続き)

資料名	説明	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 およびその構成要素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の手引き	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 30. DB2 情報 (続き)

資料名	説明	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 構成要素の使用方法を説明します。	SC27-0782 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 Setup and User's Guide	OLAP Starter Kit の構成およびセットアップに関する情報を提供します。	SC27-0702 db2ipx70	db2ip

表 30. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
OLAP Spreadsheet Add-in User's Guide 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。	SC27-0786 db2epx70	db2ep
OLAP Spreadsheet Add-in User's Guide 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。	SC27-0785 db2tpx70	db2tp
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 30. DB2 情報 (続き)

資料名	説明	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 30. DB2 情報 (続き)

資料名	説明	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 30. DB2 情報 (続き)

資料名	説明	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 30. DB2 情報 (続き)

資料名	説明	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、*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、371ページの表30 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.com 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。

オンライン・ヘルプへのアクセス

すべての DB2 構成要素で、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、複数サイト更新の構成、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。

ウィザード	内容	利用方法
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。
複数サイト更新の構成 (Configure Multisite Update)	複数サイト更新、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、**「複数サイト更新 (Multisite Update)」**を選択します。
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。

ウィザード	内容	利用方法
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。
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資料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、385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B. 命名規則

次のデータベースおよびデータベース・オブジェクトに名前を付けるときには、下記の命名規則を使用してください。

- データベース名
- データベース名およびデータベース別名
- ユーザー ID およびパスワード
- スキーマ名
- グループ名およびユーザー名
- オブジェクト名

IBM SQL または ISO/ANSI SQL92 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。これらの語のリストについては、*SQL 解説書* を参照してください。

許可 ID (ユーザー名とグループ名を含む) およびワークステーションに関する命名規則、および追加のプラットフォーム制約事項については、*概説およびインストール* を参照してください。

データベース名

新しいデータベースが作成されるたびに、データベース・マネージャーは、そのデータベースのための制御ファイルとデータ・ファイルを格納するためのディレクトリーを別個に作成します。

これらのディレクトリーの命名スキーマは SQL00001 ～ SQL*nnnnn* で、SQL00001 には最初に作成されたデータベースに関連した制御ファイルが含まれ、 SQL00002 には、2 番目に作成されたデータベースの制御ファイルが含まれ、それ以降も同様です。

これらのディレクトリーは、自動的に保守されます。ディレクトリー命名に関する問題が発生しないように、データベース・マネージャーが使用するのと同じ命名スキーマでユーザー自身のディレクトリーを作成しないでください。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。

データベース名およびデータベース別名

データベース名 は、管理者およびそのユーザーが CREATE DATABASE コマンドまたは API の一部として指定する識別用の名前です。これらの名前は、カタログされている場所で固有である必要があります。たとえば、DB2 を UNIX ベースで実装している場合はこの場所はディレクトリー・パスですが、OS/2 で実装している場合はドライブ名です。

データベース別名 は、ローカルまたはリモート・データベースに与えられたローカル同義語です。この名前は、システム・データベース・ディレクトリー (データベース・マネージャーの個々のインスタンスの別名すべてを保管している) 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 (省略時値) 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。

データベース名またはデータベース別名を命名するときには、次のように指定してください。

- 1 ～ 8 文字の長さです。
- 先頭は次のいずれかでなければなりません。
  - A ～ Z (小文字は大文字に変換する)
  - @、#、または $
- 2 番目以降の文字には次のものを使用できます。
  - A ～ Z (小文字は大文字に変換する)
  - 0 ～ 9
  - @、#、$、および _ (下線)

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、別の国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。最後に、Windows NT システムでは、どのインスタンス名もサービス名と同じであってはなりません。

ユーザー ID およびパスワード

ユーザー ID またはパスワードを作成するとき、作成する名前は次のようなものです。

- 以下の語は使用できません。

  - USERS、ADMINS、GUESTS、PUBLIC、 LOCAL または *SQL 解説書* にリストされている SQL 予約語
- 以下の文字で始まってはなりません。
  - SQL、SYS、または IBM
- 次の文字を使用できます。
  - A ～ Z

    注: 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。
  - 0 ～ 9
  - @、#、または $
- ユーザー ID は 30 文字を超過することはできません。

注: 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、この作業を続け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> VERIFY <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。

スキーマ名

次のスキーマ名は予約語なので、使用してはなりません。

- SYSCAT
- SYSFUN
- SYSIBM

- SYSSTAT

今後の潜在的な移行の問題を避けるために、SYS で始まるスキーマ名を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。

また、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。

グループ名およびユーザー名

UNIX 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。

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。

Windows NT では、ローカル・グループ名、グローバル・グループ名、およびユーザー ID を同じにすることはできません。

グループ名は 8 文字を超過することはできません。

オブジェクト名

データベース・オブジェクトには、次のものがあります。

- スキーマ
- 表
- 視点
- 列
- 索引
- ユーザー定義関数 (UDF)
- ユーザー定義タイプ (UDT)
- トリガー
- 別名

- 表スペース
- ストアード・プロシージャー
- メソッド
- ノードグループ
- バッファー・プール
- イベント・モニター

データベース・オブジェクトを命名するときには、次のように指定してください。

- 1 ～ 18 文字 (バイト) の長さです。

  注: 以下の例外があります。
  - スキーマおよび列では 1 ～ 30 文字が許可されている
  - 表、視点、相関名、および別名では 1 ～ 128 文字が許可されている
- 先頭は次のいずれかでなければなりません。
  - A ～ Z (小文字は大文字に変換する)
  - 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)
- 2 番目以降の文字には次のものを使用できます。
  - A ～ Z (小文字は大文字に変換する)
  - 有効なアクセント付き文字 (ö など)
  - 0 ～ 9
  - @、#、$、および _ (下線)
  - 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)

キーワードを使用することができます。キーワードが SQL キーワードとして解釈される可能性もある文脈で使用される場合、区切り識別名として指定する必要があります。区切り識別子については、*SQL 解説書* を参照してください。

移植性を最大にするには、IBM SQL および ISO/ANSI SQL92 予約語を使ってください。これらの語のリストについては、*SQL 解説書* を参照してください。

注:

1. 区切り識別子を使用する際に、上記の命名規則に違反するオブジェクトを作成することもできます。しかしながら、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。ご使用のデータベースの使用と操作に関する潜在的な問題を避けるには、上記の規則を**破らないでください**。

たとえば、名前に + または – 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。

連合データベースのオブジェクト名

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) のオブジェクトには、以下のものが含まれます。

- 索引仕様
- ニックネーム
- サーバー
- ラッパー
- 機能マッピング
- タイプ・マッピング
- ユーザー・マッピング

連合データベースのオブジェクトを命名するときには、制限が適用されます。オブジェクト名とそれに関連した識別子の制限および要件についての完全なリストは、 *SQL 解説書* に載せられています。要約すると、オブジェクト名には以下の制限および要件があります。

- 制限があります。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。
- 先頭は次のいずれかでなければなりません。
  - A ～ Z (引用符のない名前は英大文字に変換されます)
  - 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)
- 内部の命名規則に従う必要があります。先頭以外の文字には次のものを使用できます。
  - A ～ Z
  - 有効なアクセント付き文字 (ö など)
  - 0 ～ 9
  - @、#、$、および _ (下線)
  - 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)

キーワードを使用することができます。キーワードが SQL キーワードとして解釈される可能性もある文脈で使用される場合、区切り識別名として指定する必要があります。区切り識別子については、*SQL 解説書* を参照してください。

移植性を最大にするには、IBM SQL および ISO/ANSI SQL92 予約語を使ってください。これらの語のリストについては、*SQL 解説書* を参照してください。

オプション (サーバー、ニックネーム) 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。

- 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』) にセットします。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″ にセットします。

  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、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』) にセットできます。データ・ソースで大文字のユーザー ID を必要とする場合、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』) にセットできます。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・オプションについての詳細は、 *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 ファミリーのデータ・ソースにある多くの区切り識別子は、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 対話機能モードのコマンド・プロンプト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、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C. データベース移行の計画

このセクションでは、移行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 に移行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* のデータベースの移行に関する詳細は、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。

データベースの移行時には、次のようになります。

- 次のデータベース・エンティティーが移行されます。
  - 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移行され、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、データベース移行時に再作成されます。
- データベース活動記録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。

移行に関する考慮事項

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移行するには、次のことを考慮する必要があります。

- 『移行の制約事項』
- 401ページの『機密保護および権限』
- 401ページの『記憶域要件』
- 401ページの『リリース間の非互換性』

移行の制約事項

データベースをバージョン 7 に移行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。

- 移行が行えるのは、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移行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 に移行してから V7 に移行します。
- データベースを V7 サーバーに移行するために、 V7 クライアントの移行コマンドを出すことができます。ただし、古い DB2 クライアントの移行コマンドでは、データベースを V7 サーバーに移行することはできません。
- 異なるプラットフォーム間での移行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。
- データベースを移行する前に、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移行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。

機密保護および権限

データベースを移行するには、SYSADM 権限が必要です。

記憶域要件

移行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、移行は失敗します。

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

データベースの移行を計画するときは、 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、データベースを移行した後で、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。移行前と移行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。)

データベースの移行

次に、データベースを移行する場合に行う必要のあるステップを示します。移行を開始するには、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。

移行の前に

注: 移行前ステップは、以前のリリース (すなわち、新しいリリースに移行する前の、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. 400ページの『移行の制約事項』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、移行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 移行前ユーティリティーを使用して、データベースが移行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移行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。

   移行は回復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、移行が失敗すると、 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、回復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移行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回復では使用できません。

移行

5. 次のいずれか 1 つを使用して、データベースを移行します。
   - 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 - データベース移行 API

OS/2 の場合: DB2CIDMG 移行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操作員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 移行を使用するには、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移行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。

移行の後で

6. 任意選択で、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移行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 に移行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. 任意選択で、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。古い統計は移行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. 任意選択で、 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、任意選択で Explain 表を移行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D. 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分類されます。

- 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
- データベースの機密保護と調整
- 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移行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コード化されています。

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コード化されています。

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コード化されています。

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なおす必要があります。

スクラッチパッドの長さフィールドの変更

WIN	UNIX	OS/2

変更点: ユーザー定義関数 (UDF) で、渡されるスクラッチパッドの長さフィールドを変更すると、SQLCODE -450 が戻されるようになりました。

症状: スクラッチパッドの長さフィールドを変更する UDF は失敗します。呼び出しステートメントは SQLCODE -450 を受け取り、ここにスキーマおよび関数の固有の名前が示されます。

解決方法: スクラッチパッドの長さフィールドを変更しないように UDF 本文を書き直します。

SQL

スキーマ SESSION によって修飾される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、正規基本表用のスキーマとして使用できます。このため、アプリケーションで、実表 SESSION.T1 と宣言済み一時表 SESSION.T1 とが同時に存在することが検出される可能性があります。パッケージのバインド時に、 (明示的または暗黙的に) ″SESSION″ で修飾されている表参照を含む静的ステートメントが見つかると、このステートメントのセクションや従属関係はカタログに保管されません。代わりに、このセクションを実行時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。実行時に、表名が一致する宣言済み一時表が存在する場合、たとえ同じ名前の永続基本表が存在しても、宣言済み一時表が使用されます。

症状: バージョン 6 以前では、 SESSION によって修飾される表を含む静的ステートメントのパッケージは、常に永続基本表を参照していました。パッケ

ージをバインドする際には、セクションおよびそのステートメントに関連した従属関係レコードがカタログに保管されました。バージョン 7 では、これらのステートメントはバインド時にはバインドされず、実行時に同じ名前の宣言済み一時表に解決されます。したがって、次のような状況が生じる可能性があります。

- バージョン 5 からの移行。バージョン 5 でこのようなパッケージが存在する場合、これはバージョン 6 で再びバインドされ、静的ステートメントは増分的にバインドされます。このことは、パフォーマンスに影響を与えます。これは、増分的にバインドされたセクションが、キャッシュに入った動的 SQL のように動作するためです。ただし、キャッシュに入った動的セクションは、他のアプリケーション (同じアプリケーション実行可能プログラムの異なるインスタンスも含む) との間では共用できません。
- バージョン 6 からバージョン 7 への移行。バージョン 6 でこのようなパッケージが存在する場合、バージョン 7 で再びバインドする必要はありません。その代わり、ステートメントは依然として正規の静的 SQL として実行され、最初のバインド時にカタログに保管されたセクションを使用します。しかし、このパッケージが (明示的または暗黙的に) 再バインドされる場合、 SESSION 修飾表の参照を含むパッケージ内のステートメントはもはや保管されず、増分的バインドが必要となります。これにより、パフォーマンスが低下する可能性があります。

要約すると、バージョン 7 でバインドされたパッケージのうち、 SESSION 修飾表を参照する静的ステートメントを持つものは、増分的バインドが必要であるため、もはや静的 SQL のようには動作しません。実際、既存の SESSION 修飾表、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みなされます。

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 に移行する場合にのみ関係があります)。

```
select pkgschema, pkgname, bschema, bname from syscat.packagedep
   where bschema = 'SESSION' and btype in ('T', 'V', 'I')
```

ユーティリティーとツール

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。

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 から移行する場合で、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、128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識別子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識別子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識別子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識別子には、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。次に例を示します。

```
#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、表またはスキーマ識別子の制限によって、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移行する際に、 `db2cli.ini` ファイルで一連の任意選択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移行が必要になります。

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移行済みのパッケージ、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なお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。

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。

データベースの機密保護と調整

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E. 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、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。

国別コードおよびコード・ページのサポート

432ページの表31 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。

- **コード・ページ**は、オペレーティング・システムのコード・セットからマップされる IBM 定義のコード・ページを示します。
- **グループ**は、コード・ページが単一バイト (「S」)、またはマルチバイト (「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。
- **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、2 文字のテリトリー識別子を表します。
- **国別コード**は、その国に特有のサポートを提供するためにデータベース・マネージャーが内部的に使用する国別コードを示します。
- **ロケール**は、データベース・マネージャーがサポートするロケール値を示します。
- **オペレーティング・システム**は、言語およびコード・セットをサポートするオペレーティング・システムを示します。
- **国名**は、国の名前です。

表 31. サポートされる言語およびコード・セット

コード・ページ	コード・グループ	コード・セット	Tr.	国別コード	ロケール	OS	国名
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31. 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	Tr.	国別コード	ロケール	OS	国名
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
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 31. 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	コード・グループ	コード・セット	Tr.	国別コード	ロケール	OS	国名
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
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
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

表 31. 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	コード・グループ	コード・セット	Tr.	国別コード	ロケール	OS	国名
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
922	S-10	IBM-922	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
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

表 31. 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	Tr.	国別コード	ロケール	OS	国名
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
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

表 31. 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	Tr.	国別コード	ロケール	OS	国名
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
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
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

表 31. 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	Tr.	国別コード	ロケール	OS	国名
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
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
932	D-1	IBM-932	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
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
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 31. 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	Tr.	国別コード	ロケール	OS	国名
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
921	S-10	IBM-921	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
921	S-10	IBM-921	LV	370	-	WIN	リトアニア
1112	S-10	IBM-1112	LV	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 31. 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	コード・グループ	コード・セット	Tr.	国別コード	ロケール	OS	国名
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 31. 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	Tr.	国別コード	ロケール	OS	国名
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 31. 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	Tr.	国別コード	ロケール	OS	国名
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 31. 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	コード・グループ	コード・セット	Tr.	国別コード	ロケール	OS	国名
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 31. 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	コード・グループ	コード・セット	Tr.	国別コード	ロケール	OS	国名
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 31. 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	コード・グループ	コード・セット	Tr.	国別コード	ロケール	OS	国名

以下は、アラブ諸国 (AA) にマップされます。

```
 ----------------------------------------
     /* Arabic (Saudi Arabia) */
     /* Arabic (Iraq) */
     /* Arabic (Egypt) */
     /* Arabic (Libya) */
     /* Arabic (Algeria) */
     /* Arabic (Morocco) */
     /* Arabic (Tunisia) */
     /* Arabic (Oman) */
     /* Arabic (Yemen) */
     /* Arabic (Syria) */
     /* Arabic (Jordan) */
     /* Arabic (Lebanon) */
     /* Arabic (Kuwait) */
     /* Arabic (United Arab Emirates) */
     /* Arabic (Bahrain) */
     /* Arabic (Qatar) */
```

以下は、英語 (US) にマップされます。

```
 --------------------------------
     /* English (Jamaica) */
     /* English (Carribean) */
```

以下は、ラテンアメリカ (Lat) にマップされます。

```
 ---------------------------------------
     /* Spanish (Mexican) */
     /* Spanish (Guatemala) */
     /* Spanish (Costa Rica) */
     /* Spanish (Panama) */
     /* Spanish (Dominican Republic) */
     /* Spanish (Venezuela) */
     /* Spanish (Colombia) */
     /* Spanish (Peru) */
     /* Spanish (Argentina) */
     /* Spanish (Ecuador) */
     /* Spanish (Chile) */
     /* Spanish (Uruguay) */
     /* Spanish (Paraguay) */
     /* Spanish (Bolivia) */
```

注: 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

コード範囲	説明	Sun Big-5	IBM Big-5
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

注: 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省略時値として、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省略時値であり、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。

コード・ページ値の入手

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。しかし、DB2CODEPAGE レジストリー変数を設定する必要はありません。 DB2 は、オペレーティング・システムから適切なコード・ページ値を判別します。 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。

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 (明示または省略時解釈により) 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
Windows の場合は、DB2CODEPAGE レジストリー変数が設定されていないと、 Windows のコード・ページは、国別 ID から得られます。これは、Windows `WIN.INI` ファイルの `[intl]` セクションにある `iCountry` 値で指定されているものです。

Windows 32 ビット・オペレーティング・システム
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、 432ページの表31 を参照してください。

文字セット

一般的に、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。

識別子用の文字セット

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。しかし、これらの特殊文字は、 NLS ホスト (EBCDIC) の不変の文字セットには含まれていないので、 NLS 環境で使用するときには注意してください。

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識別子については、 *SQL 解説書* を参照してください。

DBCS 識別子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、上記のように、拡張文字セット以外の文字を含めることはできません。

両方向 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両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省略時値が異なるため、 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、両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 for OS/390 から Windows 32 ビット オペレーティング・システム上の DB2 UDB に送られたデータは正しく表示されない可能性があります。

両方向特定の CCSID

DB2 は、特別な両方向コード化文字セット識別子 (CCSID) を介して、両方向データ属性をサポートします。以下の両方向 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。両方向 (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両方向サポートの実装

両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。省略時の設定では、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 環境で特定の両方向 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両方向 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両方向 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。

両方向の 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、両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、両方向レイアウト変換を実行するための任意選択機能を備えています。

サーバー・データベースへの発信データに対して、 DB2 コネクトで両方向レイアウト変換を実行するには、サーバー・データベースの両方向 (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、省略時の DB2 ホスト・データベース CCSID になります。

省略時のサーバー・データベース両方向 CCSID をオーバーライドするときに使う両方向 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両方向 CCSID とそのストリング・タイプのリストは、449ページの『両方向特定の 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 (両方向ストリング・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 (両方向ストリング・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 (両方向ストリング・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 での両方向ストリング・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、両方向ストリング・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、両方向レイアウト変換を実行します。この変換は、CCSID 62213 (両方向ストリング・タイプ 5) から CCSID 62221 (両方向ストリング・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、両方向レイアウト変換を実行します。この変換は、 CCSID 08616 (両方向ストリング・タイプ 6) から CCSID 62213 (両方向ストリング・タイプ 5) への変換です。

注: 場合によっては、両方向 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。

特定の両方向 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、同じものとして分類されます。

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 進数の分類順序を用いて分類されます。

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、データベースの分類済みデータを、異なる照合順序を用いて分類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分類済みデータを、別のデータベースの分類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分類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分類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会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。次に例を示します。

- 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みなされます。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。

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、クライアントの国別コードと関連づけられた日時値の省略時形式になりますが、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。各種の国別コードのストリング形式のリストは、 460ページの表34 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. 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33 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 33. 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

表 33. 時刻のストリング表示の形式 (続き)

形式名	省略形	時刻形式	例
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`) は任意選択です。
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。

MBCS 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別コードに関連する日時の値の省略時の形式です。この省略時の形式は、プログラムが事前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
  - 省略時解釈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
  - 省略時解釈の出力時刻形式は、ローカルの時刻形式になります。
- 入力の日付形式
  - 省略時解釈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34 で UK の項目を見てください。
- 出力の日付形式
  - 省略時の出力日付形式が、表34 に示されています。

    注: 表34 には、各種の国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 34. 国別コードによる日時の形式

国別コード	ローカル 日付形式	ローカル 時刻形式	省略時の 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

表 34. 国別コードによる日時の形式 (続き)

国別コード	ローカル 日付形式	ローカル 時刻形式	省略時の 出力の 日付形式	入力日付形式
055 ブラジル	dd.mm.yyyy	JIS	LOC	LOC、 EUR、 ISO
359 ブルガリア	dd.mm.yyyy	JIS	EUR	LOC、 USA、 EUR、 ISO
001 カナダ	mm-dd-yyyy	JIS	USA	LOC、 USA、 EUR、 ISO
002 カナダ (フランス語圏)	dd-mm-yyyy	ISO	ISO	LOC、 USA、 EUR、 ISO
385 クロアチア	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
042 チェコ共和国	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
045 デンマーク	dd-mm-yyyy	ISO	ISO	LOC、 USA、 EUR、 ISO
358 フィンランド	dd/mm/yyyy	ISO	EUR	LOC、 EUR、 ISO
389 FYR マケドニア	dd.mm.yyyy	JIS	EUR	LOC、 USA、 EUR、 ISO
033 フランス	dd/mm/yyyy	JIS	EUR	LOC、 EUR、 ISO
049 ドイツ	dd/mm/yyyy	ISO	ISO	LOC、 EUR、 ISO
030 ギリシャ	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
036 ハンガリー	yyyy-mm-dd	JIS	ISO	LOC、 USA、 EUR、 ISO
354 アイスランド	dd-mm-yyyy	JIS	LOC	LOC、 USA、 EUR、 ISO
091 インド	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
972 イスラエル	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO
039 イタリア	dd/mm/yyyy	JIS	LOC	LOC、 EUR、 ISO

表 34. 国別コードによる日時の形式 (続き)

国別コード	ローカル 日付形式	ローカル 時刻形式	省略時の 出力の 日付形式	入力日付形式
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
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

表 34. 国別コードによる日時の形式 (続き)

国別コード	ローカル 日付形式	ローカル 時刻形式	省略時の 出力の 日付形式	入力日付形式
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

注:

1. 省略時解釈の C ロケールを使用する国には、国別コード 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 Unicode/UCS-2 および UTF-8 サポート

ここでは、2 つの規格について説明します。

概要

Unicode 文字コード化規格は、固定長の文字コード化スキームです。これには、世界で実際に使われているほとんどすべての言語の文字が含まれています。 Unicode 文字は通常は 「U+*xxxx*」 のように示されます。この *xxxx* には文字の 16 進コードが入ります。

言語に関係なく、それぞれの文字は 16 ビット (2 バイト) の大きさです。世界の主要な言語で使われているほとんどの文字をコード化するのであれば、65000 のコード要素があれば十分ですが、Unicode 規格の拡張メカニズムでは、さらに 100 万文字をコード化することが可能です。この拡張機能は、特定の 32 ビット文字を 2 つの連続したコード要素としてコード化するために、ある範囲のコード値を予約します (U+D800 から U+D8FF (これらを「サロゲート」といいます))。

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、 2 バイト版 (UCS-2) と 4 バイト版 (UCS-4) があります。 2 バイト版の ISO 規格は、サロゲートのない Unicode と同じものです。 ISO 10646 ではさらに、特定の UCS-4 コードを UCS-2 コード化ストリングでコード化するための拡張機能も定義しています。この拡張機能を UTF-16 といい、サロゲートのある Unicode と同じです。

DB2 UDB は、UCS-2、すなわちサロゲートのない Unicode をサポートしています。

UTF-8 (コード・ページ 1208) クライアントから非 Unicode データベースへの接続はサポートされていません。

UTF-8

UCS-2 すなわち Unicode コード化では、ASCII および制御文字も 2 バイトの長さであり、先頭のバイトはゼロになります。たとえば NULL は U+0000 で、大文字の A は U+0041 になります。 UCS-2 ストリングでは異質な NULL がストリング内に現れることがあり、このため、ASCII ベースのアプリケーションや ASCII ファイル・システムで大きな問題になる場合があります。一定の ASCII コードに依存するプログラムの場合には、 UTF-8 という変換アルゴリズムを使用してこの問題を避けることができます。

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 UCS-4 文字を可変長のバイト・ストリングに変換します。 UTF-8 では、ASCII 文字はそれぞれ通常の 1 バイト・コードで表されますが、 UCS-2 での非 ASCII 文字は 2 または 3 バイトの長さになります。つまり、UTF-8 は UCS-2 文字をマルチバイトのコード・セットに変換しますが、 ASCII 文字は一定であるということです。 UTF-8 形式におけるそれぞれの UCS-2 文字のバイト数は、次の表を見て判断することができます。

UCS-2 (16 進)	UTF-8 (2 進)	説明
0000 ～ 007F	0xxxxxxx	ASCII
0080 ～ 07FF	110xxxxx 10xxxxxx	最大 U+07FF まで
0800 ～ FFFF	1110xxxx 10xxxxxx 10xxxxxx	その他の UCS-2

注: D800 ～ DFFF の範囲は、この表の 3 行目の扱いから除外されます。
その範囲の値は 0000 0800 ～ 0000 FFFF の範囲の UCS-4 を制御します。

上記のいずれの場合でも、一連の *x* は、文字の UCS ビット表記です。たとえば、U0080 は 11000010 10000000 に変換されます。

DB2 UDB での UCS-2/UTF-8 の実装

コード・ページ / CCSID 番号

IBM では、UCS-2 コード・ページはコード・ページ 1200 として登録されています。すべてのコード・ページは、増加していく文字セットとして定義されます。つまり、あるコード・ページに新しい文字が追加されるときに、そのコード・ページ番号が変わってしまうことはありません。コード・ページ 1200 は、常に現行バージョンの Unicode/UCS-2 を参照し、 DB2 UDB の UCS-2 サポートで使用されます。

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。 DB2 の UCS-2/UTF-8 サポートでは、マルチバイトのコード・ページ番号として 1208 が使用されます。

DB2 UDB は、新しいマルチバイトのコード・ページとして UCS-2 をサポートしています。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。データベースが UCS-2/UTF-8 で作成されていれば、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。これを簡単に UCS-2 データベースと呼んでいます。

UCS-2 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、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、作成されるデータベースは UCS-2 データベースになります。別の方法として、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 2 文字の地域コードを使うことができます。

たとえば、米国の地域コードを指定して UCS-2 データベースを作成するには、次のコマンドを発行します。

```
DB2 CREATE DATABASE dbname USING CODESET UTF-8 TERRITORY US
```

sqlecrea API を使用して UCS-2 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 データベースの省略時の照合順序は IDENTITY です。 UCS-2 コード・ポイントの順に処理されます。したがって、デフォルトでは、すべての UCS-2/UTF-8 文字が順番に並べられ、それぞれの UCS-2 コード・ポイントの順序に基づいて比較されます。

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。

UCS-2 データベースでは、DB2 UDB によってサポートされているそれぞれの単一バイトおよびマルチバイト・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、 464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 バイトとみなすことはできません。マルチバイト UTF-8 コード化では、各 ASCII 文字は 1 バイトですが、非 ASCII 文字はそれぞれ 2 バイトか 3 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*/3 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 コード化を使うことも、合計記憶域要件に影響します。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、記憶要件は 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 文字であり、 3 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)、各 UCS-2 文字はちょうど 2 バイトを必要とするため、 UTF-8 形式で UCS-2 文字に相当する文字ごとに 3 バイトを割り当てるよりも、 UCS-2 漢字ストリング形式を選択する方が良いかもしれません。

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、 USC-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。

識別子

UCS-2 データベースでは、識別子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、識別子として任意の 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識別子について詳しくは、 391ページの『付録B. 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 SBCS または MBCS 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、識別子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 データベースで識別子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 バイトまたは 3 バイトが必要。そのため、*n* バイトの識別子は、 ASCII 文字と非 ASCII 文字の比率に応じて、 *n*/3 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 である識別子の場合 (たとえば、日本語の場合)、使用できるのは *n*/3 文字だけです。
- 識別子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、識別子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、すべての識別子を実際に ASCII に限定する必要があります。

UCS-2 リテラル

UCS-2 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 UCS-2 に変換されます。
- UX'....' または GX'....' 形式を使用した UCS-2 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、 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 UCS-2 コード・ポイントを表します。

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 MBCS 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 MBCS 文字のストリングを表す。

突き合わせ式に漢字ストリング DBCS データが含まれる場合、その式には DBCS 文字だけが含まれます。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。

- DBCS 下線は 1 つの DBCS 文字を表す。
- DBCS パーセント記号は、ゼロ以上の DBCS 文字のストリングを表す。

UCS-2 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。 UCS-2 文字はそれぞれ 2 バイトを使います。 UTF-8 形式では、UCS-2 文字が「混合バイト」でコード化されますが、 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かかわらず UCS-2 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 UCS-2 文字を表し、パーセント記号はゼロ以上の UCS-2 文字のストリングを表します。

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・ページの省略時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 0x01 ～ 0x7F 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みなされ、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みなされます。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・ページであるとみなします。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・ページであるとみなされます。 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・ページであるとみなし、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・ページであるとみなします。

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F. 特記事項

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミングまたはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミングまたはサービスを、日本で発表する意図があることを必ずしも示すものではありません。本書で、IBM ライセンス・プログラムまたは他の IBM 製品に言及している部分があっても、このことは当該プログラムまたは製品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。これらのプログラムまたは製品に代えて、IBM の知的所有権を侵害することのない機能的に同等な他社のプログラム、製品またはサービスを使用することができます。ただし、IBM によって明示的に指定されたものを除き、これらのプログラムまたは製品に関連する稼働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 および他社は、本書で説明する主題に関する特許権 (特許出願を含む)、商標権、または著作権を所有している場合があります。本書は、これらの特許権、商標権、および著作権について、本書で明示されている場合を除き、実施権、使用権等を許諾することを意味するものではありません。実施権、使用権等の許諾については、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。

本書に含まれる情報には、技術的に不正確なもの、または誤植が含まれる場合があります。これらに対する変更は、定期的に行われます。これらの変更は、資料の改訂版に含まれます。IBM は、本書で説明している製品、プログラムに対して、予告なく改良、変更を加える場合があります。

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、お客様になんら義務も負わせない適切な方法で、使用もしくは配布することがあります。

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

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

本プログラムに関する上記の情報は、適切な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。

本書に含まれるパフォーマンス・データは、制御された環境下で決定されています。したがいまして、その他の稼働環境で得られる結果とは、かなり異なる可能性もあります。一部の測定値は、開発中のシステムを使用している場合があり、これらの測定値が一般的に提供可能なシステムで同様の数値になることを保証するものではありません。さらに、一部の測定値が推定されたものもあります。実測値と異なる場合があります。本書のユーザーは、使用される特定の環境での該当データを確認してください。

IBM 以外の製品については、当該製品の提供者から直接、出版されている資料または一般公開されている情報から入手しました。IBM は、これらの製品についてはテストを行っておらず、これらの IBM 以外の製品に関する性能、互換性またはその他の主張について確認することはできません。IBM 以外の製品の機能に対する質問は、それぞれの製品提供者にお問い合わせください。

IBM の将来の方向性または意図については、予告なしに変更または中止する場合があり、 IBM の目的および目標のみ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が、これは説明に具体性を与えるために記載されたものであり、それらの例には、個人、企業、ブランドの、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。それらの名前はすべて架空のものであり、また名称や住所が類似する企業が実在しても、それは偶然に過ぎません。

著作権：

本書に含まれる情報には、サンプル・アプリケーション・プログラムがソース言語の形式で含まれており、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技法を示しています。お客様は、これらのサンプル・プログラムが書かれているオペレーティング・プラットフォームでアプリケーション・プログラミング・インターフェースが実行可能となるためのアプリケーション・プログラムを開発、使用、販売または配布もしくは転送する目的のためにのみ、サンプル・プログラムを、IBM に対する別途料金を支払うことなく、複製、変更、配布または転送することができます。これらのサンプルは、すべての条件下で十分にテストを行っていません。したがいまして、IBM は、これらのプログラムの信頼性、実用性または機能について、いかなる保証も負いません。

サンプル・プログラムまたはその改変版の複製物には、全部複製か部分複製かを問わず、次の著作権表示を必ず行うものとします。

商標

以下は、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

次のものは、他社の商標または登録商標です。

Tivoli および NetView は、米国およびその他の国における 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等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］

E

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。

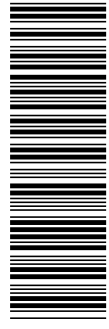
米国以外の国で IBM に連絡する方法については、 *IBM Software Support Handbook* の Appendix A を参照してください。この資料にアクセスするには、 Web ページ: http://www.ibm.com/support/ にアクセスし、ページの最下部にある「IBM Software Support Handbook」リンク・ボタンを選択します。

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。

部品番号: CT7XVJA

Printed in Japan

CT7XVJA



SC88-8513-00

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